The Building Blocks of a Successful API

Author: Kin Lane

August, 2012
The Building Blocks of a Successful API

The current API movement has been built on existing web technologies, using HTTP to transmit valuable business resources, via meaningful, descriptive URL endpoints. When anyone talks about APIs or Application Programming Interfaces, the conversation is immediately dominated by technical jargon, arguing about the finer points of HTTP or REST--while in reality there are many more building blocks that go into a successful API.

This white paper is the product of reviewing the 6000+ APIs available in the ProgrammableWeb API directory. After reviewing that many APIs, patterns begin to emerge of the building blocks that are used by the successful APIs.

This paper walks through the most common of these building blocks, used by 1500 of the most successful APIs. Let’s not waste anytime, and dive right in.

Landing Page

The first thing any consumer of an API will see is the landing page. Just like the home page of a website, the API landing page should speak to the target audience, telling them of the value the API possesses and providing single click access to every aspect of the API.

On the API landing page, the first thing a user should see is a short, descriptive paragraph explaining the value delivered by the API. There is no need talk about REST or JSON, API owners should explain what problems the API solve, in a language anyone will understand. Don’t assume anything about visitors. In all likelihood the user will know nothing about the company, products and history.

Each API landing page should speak to as wide an audience as possible, even non-developers, rising above the technological jargon and clearly explain the business value delivered by an API.
Technology

APIs are programmatic interfaces for applications to speak to each other, so at their heart they are very technical. Current trends of API are centered around a single approach using Representational State Transfer (REST), but an API platform is not limited to REST, and can employ several different technologies to empower developers to deploy modern mobile or web applications.

Pragmatic REST

REpresentational State Transfer (REST)
is a style of software architecture for distributed systems introduced and defined in 2000 by Roy Fielding in his doctoral dissertation, that has increasingly displaced other design models such as SOAP, due to its simplicity.

I label this building block as “Pragmatic REST” to avoid the common responses from the community about what is truly REST. I encourage you to read Roy Fielding’s dissertation, buy books like APIs: A Strategy Guide and participate in communities like API Craft--then form your own vision of what REST is, to meet your business objectives.

REST’s success lies in its simplicity, but is born out of very heated, ongoing and complex discussions around its design. When planning the design of your API, focus on the the simplicity, and what is important to your company. Don’t listen to the haters, there will always be someone who will criticize you for your design.

JavaScript Object Notation (JSON)

JSON or JavaScript Object Notation, is a text-based open standard designed for human-readable data interchange, derived from the JavaScript scripting language for representing simple data structures and associative arrays, called objects.

JSON has become the favored data format of API developers, with many APIs opting to only provide JSON response formats over XML--as JSON is much more lightweight than XML. I still consider XML as a viable option, but no modern API should deploy without JSON for data interchange.
Extensible Markup Language (XML)

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable, produced by the W3C. XML has been around since the 1990’s, and was the early format of web APIs—but is quickly being replaced by JSON as the preferred data format.

Depending on your APIs target audience you may want to still include XML as an option for an API, but make sure and consider the network overhead and impact on developers before implementation.

Basic HTTP Authentication

Basic HTTP Authentication is a way for a web browser or application to provide credentials in the form of a username and password. Since Basic Auth is integrated into HTTP protocol it is the easiest way for users to authenticate with a RESTful API, however if SSL is not used, the username and password are passed in plain text and can be easily intercepted on the open Internet.

OAuth is a much better choice for RESTful API authentication, but Basic Auth is perfectly suited for APIs that are intended for a wider audience and do not give access to sensitive information.

OAuth

OAuth (Open Authorization) is an open standard for authorization. It allows users to share their private resources stored on one site with another site without having to hand out their credentials, typically username and password.

OAuth allows users to hand out tokens instead of credentials to their data hosted by a given service provider. Each token grants access to a specific site for specific resources and for a defined duration. This allows a user to grant a third party site access to their information stored with another service provider, without sharing their access permissions or the full extent of their data.

OAuth is the preferred authentication method for APIs. There are plenty of resources available for deploying an OAuth server in your preferred language, for use with your API.
oEmbed

OEmbed is a format that allows users to embed content, such as photos or videos, into web sites. OEmbed APIs, bundled with simple, configurable JavaScript giving API users a simple way to embed and syndicate content to other web sites—even for non-developers.

If the resources you make available via your website and API are media rich, oEmbed is an essential technology allowing users to readily share and syndicate your media in a way that is familiar to wide audience.

WebHooks

Webhooks are a form of push notifications, triggered by a defined action, and upon triggering will submit a HTTP POST to a provided URL.

Commonly webhooks allow developers to choose the action, URL and fields associated with the webhook push. Once triggered a webhook will pass all associated fields to the developers provided URL, where they can handle the HTTP POST, and process field values.

Webhooks are a great way for you reduce polling on an API, pushing data to API developers only when the action is triggered. Webhooks make the API a two-way street, not just allowing developers to pull data from your platform, but also receive data in real-time as events occur.
Onboarding

Frictionless onboarding of developers with an API is priority number one. API owners will have one chance to onboard a developers successfully, if developers have a bad experience, the chances they will return for another try is very slim. After a well designed API, there are some proven building blocks that assist developers with onboarding.

Self-Service Registration

You’ve launched an API to stimulate innovation around your company’s resources. Don’t stand in the way of developers innovating. Provide self-service registration for developers to gain access.

Even if an API is not meant to be public, there are plenty of ways to provide self-service access while still securing the API—such as using invitation codes or limited entry service tiers, that limit access when developers first register. Innovation around an API will occur around the clock, don’t make developers wait for access.

Another area to consider when trying to reduce friction during registration, is allowing developers to register using their existing social network accounts like Twitter, Facebook and Google. When employing this option, make sure and include Github, which is the most logical of social networks for developers to use when registering for an API—Github allows the API owner to stay in tune with the profile of a developer that is most relevant to APIs and programming, and not just their everyday social life.

Getting Started

The API landing page clearly defines the what of an API, and the getting started should deliver the how of API integration. Developer onboarding is dependent on laying out simple, clear steps for developers on how to register, authenticate, access documentation and code samples, get support and any other details that are essential to API integration.

A quality getting started guide should be easily accessed from the API landing page, and kept simple. Try to only provide what a developer needs to get started with as few possible steps as possible. The getting started will not just make developer onboarding frictionless, it will speed up onboarding, increasing the chance a developer’s successful integration.
Best Practices

An API delivers access to a business's valuable resources, and no doubt there will a level of conduct that will be expected of developers while accessing these resources. While a developer is onboarding, this is the best time to provide a plain English introduction to the best practices around integrating with an API.

Best practices should be an introductory lesson for developers of how to properly integrate with an API, it should not dive into the legalese and granular detail provided by terms of use. Think of your best practices as a more friendlier, and easier to follow version of your terms of use--setting the proper tone of how a developer will use APIs as they are getting started.

Documentation

Documentation is the guide to accessing an API. It provides a description of each of the API endpoints as well their individual methods, parameters and other necessary details. Poor API documentation is the number one pain point developers suffer from when integrating with an API--there are several building blocks that alleviate this pain.

List of Endpoints

Before a developer is thrown into the full detail of API documentation, it helps to introduce them to all available API endpoints, getting them acquainted with the resources available. A simple listing of all endpoints provides a quick introduction, that will prime developers for a deeper dive.

After reviewing all API endpoints a developer can start to imagine how their application will integrate with an API, further understanding the value the API will bring to their application. Sometimes it's hard to see the 100K view of an API from regular documentation, start with just listing the API endpoints.

Documentation

Quality API documentation is the gateway to a successful API. API documentation needs to be complete, yet simple--a very difficult balance to achieve. This balance takes work and will take the work of more than one individual on an API development team to make happen.

API documentation can be written by developers of the API, but additional edits should be made by developers who were not responsible for deploying the API. As a developer, it’s easy to overlook parameters and other details that developers have made assumptions about.

<table>
<thead>
<tr>
<th>API Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>activity</td>
</tr>
<tr>
<td>• flickr.activity.userComments</td>
</tr>
<tr>
<td>• flickr.activity.userPhotos</td>
</tr>
<tr>
<td>auth</td>
</tr>
<tr>
<td>• flickr.auth.checkToken</td>
</tr>
<tr>
<td>• flickr.auth.getFrob</td>
</tr>
<tr>
<td>• flickr.auth.getFullToken</td>
</tr>
<tr>
<td>• flickr.auth getToken</td>
</tr>
<tr>
<td>auth.oauth</td>
</tr>
<tr>
<td>• flickr.auth.oauth.checkToken</td>
</tr>
<tr>
<td>• flickr.auth.oauth.getAccessToken</td>
</tr>
<tr>
<td>blogs</td>
</tr>
<tr>
<td>• flickr.blogs.getList</td>
</tr>
<tr>
<td>• flickr.blogs.getServices</td>
</tr>
<tr>
<td>• flickr.blogs.postPhoto</td>
</tr>
<tr>
<td>collections</td>
</tr>
<tr>
<td>• flickr.collections.getInfo</td>
</tr>
<tr>
<td>• flickr.collections.getTree</td>
</tr>
</tbody>
</table>
Error Response Code

Errors are an inevitable part of API integration, and providing not only a robust set of clear and meaningful API error response codes, but a clear listing of these codes for developers to follow and learn from is essential.

API errors are directly related to frustration during developer integration, the more friendlier and meaningful they are, the greater the chance a developer will move forward after encountering an error. Put a lot of consideration into your error responses and the documentation that educates developers.

API Explorer

REST API explorers allow users to make calls against APIs using a web interface, providing users with the ability to authenticate, navigate endpoints, and make requests—with the ability to see live responses, resulting JSON or XML and HTTP headers.

API Explorers get developers familiar with an API and its endpoints, allowing them to see live data from the API without having to write code. This process allows developers to get familiar with an API, see and experience the value it delivers before investing time and energy into integration.

Interactive Documentation

There is a new movement in API documentation, one that is moving beyond static, often boring documentation and into a new realm where API documentation is live and interactive. Following in the footsteps of API explorers these new interactive documentation formats like Swagger and Mashery I/O Docs, allow developers to authenticate, navigate endpoints and make requests with live responses returned.

In a little over a year, interactive API documentation has gone from a new innovation of a select few APIs, to being a standard offering among many of the leading APIs in the space. There is no better way to get your developers acquainted with an API, than allowing them to interact with your API while reading documentation—turning API documentation into a hands on experience.
Authentication

Authenticating with an API is a challenge every developer faces during integration, whether it’s a simple key or OAuth, there are some common building blocks API owners use to help reduce friction for developers when it comes to authentication.

Overview

Always provide an overview of what type of authentication is provided for an API. Don’t assume developers will know anything about Basic Auth or OAuth. Walk developers through goals behind authentication, with links to tutorials regarding authentication technology.

If an API employs OAuth, make sure and take extra special attention to provide clear instructions on how to use, as well as language specific code as part of your API SDKs. After poor API documentation, OAuth integration is the number one stumbling point for API developers.

Tester

When possible, provide a testing tool for authentication. From key and Basic Auth to OAuth, allow developers to enter their keys or tokens and validate the credentials they are using, to make sure they are using the proper credentials. A simple tester can provide quick validation that they are doing it right or show them where they are making a mistake, eliminating serious frustration while programming.
Code

API integration is all about the code. You can have the best written documentation, complete with interactive documentation and developers will walk away if you don’t provide quality code samples and SDKs that allow developers to go from registration to functional apps in minutes.

There are several code related building blocks that the top API providers are using to make integration as easy as copy and paste.

Code Samples

Working code samples in all the top programming languages are common place in the most successful APIs. Documentation will describe in a general way, how to use an API, but code samples will speak in the specific language of developers.

Make sure and provide a wide variety of code samples, not just what the API development team uses. Consider popular web languages like PHP, Python and Ruby as well as lesser languages like ColdFusion and Perl. Make sure and consider enterprise developers with Java, C# and even VisualBasic. Then consider JavaScript and the popular JS platform NodeJS, which will feed into later building blocks of a healthy embeddable strategy.

Remember your developers may not speak HTTP and REST, but will speak one of these programming languages fluently, so make code samples as inclusive as possible, reaching the widest possible audience.

Software Development Kits

Software Development Kits (SDK) are the next step in providing code for developers, after basic code samples. SDKs are more complete code libraries that usually include authentication and production ready objects, that developers can use after they are more familiar with an API and are ready for integration.

Just like with code samples, SDKs should be provided in as many common programming languages as possible. Code samples will help developers understand an API, while SDKs will actually facilitate their integration of an API into their application. When providing SDKs make sure and consider a software licensing that gives your developers as much flexibility as possible in their commercial products.
Starter Projects

Many API owners are going beyond just code samples and generic SDKs for their API ecosystems and providing open-source, private label applications built on top of an API that developers can download, modify and deploy.

These projects go by many names, but are commonly known as starter kits or projects. Starter kits can act as code samples, and may contain a version of an SDK, but provide a complete application that reflects common integrations with an API. As with samples and SDKs, start kits will speed up integrations, providing developers with the path of least resistance from registration to active API integration.

Platform Development Kits (PDK)

In the era of cloud computing there are numerous Platforms as a Service (PaaS) that allow developers to integrate with existing, thriving ecosystems like Facebook or Salesforce. While planning code related building blocks consider these platforms, and the possibility of providing platform specific development kits.

Every API that deploys will suffer from a lack of developers. Building Platform Development Kits (PDK) can help an API go from zero users to an active community of developers by embracing these existing providers. There are many different incarnations of these platforms, such as Facebook, Wordpress, Drupal, Salesforce and Heroku--spend the time and get to know which platforms may best suit your API.

Application Showcase

Complete, functioning applications built on an API is the end goal of any API owner. Make sure and showcase all applications that are built on an API using an application showcase or directory. App showcases are a great way to showcase not just applications built by the API owner, but also showcase the successful integrations of ecosystem partners and individual developers.

Do not hesitate populating an application showcase with your own active or starter kit applications. As with all of the API code, make sure and provide as liberal licensing as possible to ensure developers can be successful with use.
Github / Gists

Github is a social coding platform allowing developers to publicly or privately build code repositories and interact with other developers around these repositories—providing the ability to download or fork a repository, as well as contribute back, resulting in a collaborative environment for software development.

Github provides an excellent platform for API owners to engage with developers outside of a local API ecosystem. Developers are already actively pushing code and interacting on Github, making it a perfect opportunity for API owners. It is standard practice for API owners to establish Github repositories for all SDKs, starter projects and Platform Development Kits (PDK), as well as deploy Github Gists for actively displaying all code samples.

Beyond using Github for pushing code, as stated above in the self-service API registration building block, Github also provides a preferred account integration for API developers to use that provides opportunities for healthy social interactions amongst API ecosystem developers.

Open Source

An API is inherently an external part of a company. The documentation, code samples, SDKs, starter kits, platform development kits and any code related to an API, should be considered external intellectual property and licensed accordingly. Consider open sourcing all of the code associated with an API.

Open source will fuel the innovation that is already present in API ecosystems, further reducing the friction experienced by developers in successfully integrating their applications and businesses with an API.
Mobile

Mobile is currently the largest driving force behind API growth. The distributed nature of APIs are perfect for delivering bite size chunks of resources that mobile applications can consume--fueling a massive demand for web APIs by mobile developers.

While mobile historically was lumped in with all other code and documentation for an API, because of the demand, it is common to provide dedicated resources for mobile developers. Make sure and consider a variety of popular trends in API deployment specifically tailored for mobile development.

iOS

The trends is clear. Apple is the dominant platform for mobile application development. API owners need to have a clear understanding of what iOS developers are needing for both iPhone and iPad application development. When possible, provide iOS specific code samples, SDKs and other resources iOS developers can employ to make their API integrations successful.

Android

When it comes to mobile development, Google’s Android platform is definitely the number two player in the space, and warrants similar attention as the iOS building block. Consider providing Java code samples and SDKs specifically for the Android mobile platform. Android is picking up momentum in the space and with new devices being released all the time, API owners can’t ignore the platform as a serious contender.

HTML5

While the native app vs. HTML5 app development battle rages on, API owners have to closely pay attention to HTML5 as a viable alternative offering for their API developers, right alongside iOS and Android. For many web developers, HTML5 is a natural transition to mobile development--a factor that may tip the future toward more HTML5 mobile implementations. Big players like Apple, Facebook and Google are investing heavily in the future being HTML5, which sends the signal to API owners, that they should do the same.

PhoneGap and Trigger.io

One way to address the iOS, Android and HTML5 building blocks all at once is using cross-platform mobile development platforms like PhoneGap and Trigger.io. These platforms provide a suite of mobile development tools that allow developers to build mobile applications that can be deployed as native apps for iOS and Android while also providing mobile web versions. Providing API developers with PhoneGap and/or Trigger.io resources, API owners can rapidly grow the number of mobile apps developed on top of an API, with fewer resources.
Tiggzi

A new breed of mobile development platforms are emerging, and the leader of these is a product called Tiggzi, from established technology company Exadel. Using Tiggzi developers and even non-developers can build cross-platform mobile applications using a GUI building environment. Tiggzi provides a suite of API connectors allowing for rapid mobile application development using APIs, with the ability to then deploy as native iOS and Android as well as web mobile applications. API owners should consider working with Exadel to deploy an API connector for their companies API.

Self-Service Support

Development on an API will occur 24 hours a day, 7 days a week. With this type of schedule, developers will need support around the clock. Providing direct support around the clock will be expensive, this is why API ecosystems have adopted self-service support building blocks to meet the demands of developers.

Forum

Forums have become an essential building block in API communities for self-service support. A well moderated, active forum can evolve an APIs development area into an actual community. All forum communities will require the API owner to engage developers, keeping conversations active and questions answered, but with the right developers your forum can become self regulating—with opportunities for more senior developers to answer the questions of newer, more junior users, providing potentially free resources for an API owner.

Frequently Asked Questions (FAQ)

FAQs are a standard in many online platforms, not just APIs. API owners will know what the common questions developers ask when onboarding with an API, or learning advanced concepts. These questions, with relevant answers need to be published where developers can quickly find them. An FAQ building block is a great way to address these questions in a self-service way--make sure and keep the page a living document, adding new questions received through other support channels, as well as keeping older questions fresh.

Stack Exchange

Developers do not always rely on the forum or support of a single API. They have long established their own communities for support across many public APIs, programming languages and platforms. The leading open forum for this developer activity is Stack Exchange.
Stack Exchange provides a community for developers of all types to share knowledge and answer questions encountered during development of any type. API owners need to actively monitor and participate in conversations on Stack Exchange and not expect developers to always engage on their own local API forum.

Some API ecosystems like Foursquare and Facebook have even migrated their entire forum to a dedicated Stack Exchange forum, in recognition of the value developers put into the Stack Exchange.

**Direct Support**

Self-service support for an API is an essential area, but doesn’t meet the needs of all developers. Alongside self-service support, API owners will also need to provide a suite of support building blocks where developers can directly engage with API owners.

**Office Hours**

Open office hours is a great way to provide direct support for developers, in a controlled and sustainable format for API owners. Many popular APIs post office hours each week, giving developers an open time they can engage with support representatives via Skype, Google Hangouts or sometimes even in person. Consider the possibility of using API open office hours to support an API community.

**Email**

Email can be overwhelming in some support situations, but for many developers, email support is still an important way to communicate with support representatives and solve their problems. Don’t dismiss providing an email address that developers can use to directly engage with API support representatives, it may be the critical support link to help developers be successful during their integration.

**Phone**

Much like email, a phone number can be a solid choice for API developer support. Especially if you have a dedicated, partner target audience. Phone can be the instant gratification that developers need when they hit problems, get their questions answered and move forward with their API integration. Not all API providers will have the resources for phone support, but in some circumstances it will do the trick.

**Tickets**

Providing developers with a support ticketing tool, using a custom system, or via a popular platform like Zendesk, can be the healthy way to support the needs of an API ecosystem. Not all developers will want to publicly post their problems, and support tickets can be a very organized way to handle the direct support needs of developers, allowing API owners to respond quickly to easy questions, but also allowing them to organize larger scope items into lists that can be used in API product development, providing a direct link between developer support and the API roadmap.
Communications

Communication from an API provider can be the one area that will make or break even the most successful API. Consider the Twitter API ecosystem when you think about API communications. While Twitter has executed in amazing ways on many of its API building blocks, because of severe communication missteps, its developer community has be perpetually up in arms since March of 2010. There are several proven API building blocks when it comes to communication with developers.

Blog

An active blog can provide a quality SEO presence for an API, attracting developers and businesses to the API. Secondarily a blog can provide essentials communications for the developer community. While researching this white paper and reviewing 6000+ APIs, a blog is the number one way I could tell when an API is dead and nobody is supporting the community. A blog can easily provide the communication to keep an API active and growing, while also be the barometer of whether developers should steer clear of an API.

Twitter

Twitter as a communication platform, much like a blog is a great way to establish an active presence for an API, providing updates about API endpoints, build relationships with developers and establish partnerships with other API providers. Also in line with a blog, it can be the communication tool that demonstrates your supporting your API, while an out of date Twitter stream can show that nobody is home to support an API--sending the signal developers should steer clear of the platform.

LinkedIn

LinkedIn is a powerful business communications platform. While LinkedIn is not the preferred platform of many open API developers, it is the preferred platform of enterprise developers. As an API communications building block, an active presence on LinkedIn is recommended for API owners, it can add a healthy dimension to your communication strategy and reach older, more established developers that may not always be considered when deploying public APIs.

Facebook

Like LinkedIn, Facebook carries a great deal of social weight when it comes to working with developers. Depending on your target developer audience, Facebook may or may not make sense as part of your communication strategy. Facebook is larger than just the individual social network accounts, and a Facebook Page can be a great way for API owners to attract and engage with developers who are building applications. Consider the Facebook effect when assembling API communication building blocks.
Google+

While Google+ is not as popular as Twitter or Facebook, it does have as many active users as LinkedIn these days, and with considerable SEO benefits, it is recommended that you consider Google’s social network as one of the API communication building blocks. Google+ has a tremendous amount of network effects beyond just the social network and Google SEO. Tools like Google Hangouts can be used as part of API open office hours, and events can be used to coordinate API focused gatherings.

Email Newsletter

An email newsletter is a proven communication tool beyond APIs, and while many developers will not be open to receiving regular emails about an API, there are some developers who are still receptive to this format. As an API owner, there is also a positive effect from having to gather thoughts each week for an email newsletter that goes beyond just communicating with the developer community.

Calendar

An active API will have many events that can be shared with its community, ranging from conferences, hackathons and meetups the API provider will be attending, to industry related events that developers can benefit from. A published calendar is a great way to publicize these events, while also showing that the API is actively engaged within the API community and beyond.

Updates

Past, present and future updates about an API are a critical piece of API operations. Developers are relying on knowing where an API owner is headed with the API, and knowing the current status of the API is critical to overall ecosystem health. There are three building blocks that are used for managing updates in an API ecosystem.

Roadmap

API owners are asking developers to invest in building applications on their platform. This is asking for a lot of trust, and the best way an API owner can build this trust with its developers is with a transparent roadmap. API roadmaps are usually a simple, bulleted list, derived from the APIs own internal roadmap, showing what the future holds for the platform. Transparency around an APIs roadmap is a tough balance, since you don’t want to give away too much, alerting your competitors, but your developer ecosystem needs to know what’s next. API owners need to find a balance that works for their company, and maintain an active roadmap outlining where the platform is headed.

API Status

Uptime is critical for any website or application and the same is true for an API. API status dashboards are becoming a common way for API owners to keep developers informed of the availability of API endpoints. An API status dashboard can help alleviate support calls during outages, and provide an overall
availability rating for an API that developers can use when deciding whether or not they should use an API. Don’t be afraid to show your status, if you have a quality product and team, it won’t be a problem.

Change Log
Knowing the past is a big part of understanding where things are in store for the future. A change log should work in sync with the API roadmap building block, but provide much more detailed information about changes that have occurred with an API. Developers will not always pay attention to announced updates, but can use a change log as a guide for deciding what changes they need to make in their applications and how they will use an API. The change log will be another building block to keep developers updated, and reduce overall support resources needed.

Service Levels
Offering a variety of service levels for an API is common practice. Self-service registration for an API commonly represents the entry service level of an API. Developer access can be regulated through a well thought out framework of service levels, pricing and rate limits. There are three common building blocks to properly implement and manage API services.

Service Tiers
A well planned API will have multiple service tiers for developers to take advantage of. Before developers begin integrating their applications with an API, they need to have a clear understanding of what services are available to them. Successful API owners need to openly communicate all service tiers available, and provide simple and comprehensive descriptions of each. With no surprises on services available to them, developers can confidently build their applications on top of an API, understanding at which levels they will need to adjust their integration to take advantage of new levels of a platform.

Pricing
Pricing doesn’t always apply to APIs. It’s very common to provide API service for free. However, whether or not you charge for an API, you should clarify this for developers. Provide a pricing page, outlining what a developer gets for free and provide clear pricing for any other service levels, so developers will know what to expect as their usage grows. Even if the API is free, API owners should put thought into the future of the platform, set realistic expectations of how the platform will generate revenue to say in operation.

Rate Limits
Rate limiting is the industry standard for controlling what a developer can do with an API and reducing abuse on the platform. As with service tiers and pricing, put a considerable amount of thought into API rate limits. Make sure rate limits are in sync with service tiers, with pricing clearly posted, as well as proper relief valves so developers do not end up frustrated. Rate limits are meant to prevent abuse, not stifle innovation--put a lot of thought into the rate limits, and actively evaluate them to make sure they are achieving the objectives of the API platform.
Monetization

It costs money to build applications and maintain operations. If an API owner expects developers to be successful and build viable businesses, there needs to be monetization opportunities. At the very least the API terms of use need to allow for external advertising and commercial opportunities, but there are a couple of common monetization building blocks used by popular API providers.

Affiliate Program

Affiliate programs are a proven way to share revenue with users of a platform, getting adoption on popular e-commerce platforms like Amazon E-Commerce, and popularized by aggregate platforms like Commission Junction. These same affiliate models can be applied within an API platform to incentivize and share revenue with developers who drive traffic via API endpoints. API owners need to consider what their core value is of their API endpoints, define conversion events around this value, then educate and attach revenue via an affiliate program for developers to build into their own business models.

Advertising

Advertising is a proven monetization tool when building web or mobile advertising. Many popular APIs are deploying their own advertising platforms or integrating with existing advertising platforms for web or mobile devices. Advertising is not relevant for all APIs, but API owners should consider the opportunities advertising could offer for their API developers.

There are many other monetization models being developed by API owners, and goes beyond the scope of this paper. Don’t stop with just affiliate programs and advertising, make sure and look a little deeper.

Resources

API owners need to heavily invest in developers success by providing as many resources as possible to help showcase the value an API offers, the opportunities available via an API platform, and how to properly integrate in any language or platform. There are four common resource building blocks to support API developers.

How-To Guides

Many developers can get up and running without any help at all. Other developers need a helping hand, showing them how to use the API and put code samples and SDKs to use. How-to guides can provide the essential resources for developers to get up and going with an API. Start with common integration scenarios and build how-to guides around them, then as new ways of integrations emerge create fresh how-to guides using these new ways of taking advantage of the API.

Case Studies

APIs are all about partners and developers building new applications and finding innovative ways to integrate. When anyone builds some notable application on top of an API, develop a case study. Case
studies don’t need to be novels, make them short, concise and showcase what a partner or individual developers has done. Case studies will stimulate other developers imaginations, while also showing the API is a viable platform that others are building on top of.

Webinar / Videos
Not everyone likes to read how-to guides or case studies. Many developers prefer to have a visual walk through of how to integrate with an API or the case studies of how other developers have built on top of an API. When appropriate, make webinars and videos around your how-to guides and case studies. If video productions of case studies and how-to guides are standard operating procedure, the work can occur while you produce the core paper. Youtube and Slideshow are great platforms for distributing webinars and videos of API resources.

White Papers
White papers demonstrate domain and industry expertise. APIs are about exposing valuable business resources and assets of a company. Producing white papers can actively demonstrate the expertise a company possess and how the API resources the company offers can solve problems and provide sounds solutions for an industry and business sector. Make white papers a regular part of the API content creation, and when ready, publish to the API area as well as syndicate across the web.

Research & Development
API are amazing research and development environments where ideas are born, experimented with, and maybe go nowhere or maybe they get traction and become full blown apps or features that get integrated into the core roadmap. There are various building blocks success APIs are using to foster research and development.

Labs
Innovative API owners are establishing dedicated lab environments within their API areas, showcasing projects that are being incubated either internally or with partners. APIs are about transparency, and showcasing projects in their early stages is a good way set a tone of innovation within an API community. When possible, find ways to include developers in the process and let them contribute code and feedback to labs projects.

Ideas
As an API owner you will have ideas flowing from all directions--internally, from partners and submissions from the API community. Establishing a revolving door for ideas is important, if they don’t take hold internally and immediately get used, put them out to the community and showcase them in an idea forum. Encourage developers to submit their own, vote on, and comment or take ownership of ideas. Idea showcases stimulate developers, planting the seeds of innovation every API owner wants to see thrive in their ecosystems.
Opportunities

After you’ve managed developer communities for a while you will find developers are very open to suggestion and pointing them in the right direction. Some API owners are creating sections of their API communities that showcase opportunities for developers to start projects and build applications. A dedicated opportunities page, bundled with an idea showcase and a transparent roadmap will help keep developer activity in line with company goals.

Legal

APIs are exposing a business’s valuable resources and assets to partners and sometimes to the open public. There has to be legally binding terms of use, setting the stage for how developers can use a an API, privacy policies to protect users, and branding to help make sure the API owners brand is protected.

Terms of Use

Terms of Use provide a legal framework for developers to operate within. They set the stage for the business development relationships that will occur within an API ecosystem. TOS should protect the API owners company, assets and brand, but should also provide assurances for developers who are building businesses on top of an API. Make sure an APIs TOS pass inspection with the lawyers, but also strike a healthy balance within the ecosystem and foster innovation.

Privacy

Privacy policies protect the rights of partners, developers and platform users while also protecting the API owner from damaging activity via the API platform. Like API terms of use, privacy policies need to strike a balance and protect everyone involved, but also allow for innovation and commercial activity.

Branding Guidelines

Along with the other business assets made available via an API ecosystem, the API owners brand is also being put on the on the line. Branding Guidelines set the tone for how partners and developers can use the resources and assets made available via an API. The branding guidelines will provide a framework for attributing the API owner, how resources can be displayed and provide visual assets to support the company brand. As with terms of use and privacy policies, branding guidelines need protect the API owner but also provide developers with enough freedom to innovate.
Embeddable

Embeddables relate to snippets of code that can be embedded on any site or application. They are usually JavaScript, but also can just be HTML. A good API embeddable strategy will not only extend the reach of an API across the Internet it will open up an API beyond just the developer community. There are three common embeddable API building blocks used by top API providers.

Buttons
Buttons are shareable snippets of code that often share, syndicate or trigger a variety of events that benefit an API platform. Buttons play an important role in social media and social networks. Consider how Twitter’s share button has made Twitter a global communication platform or how the Digg button transformed social news. Embeddable buttons built on top of an API can significantly extend the reach of API resources.

Badges
Badges are common for displaying content and resources delivered via an API and allow these assets to be embedded on any website or application. API platforms like LinkedIn and Google have successfully employed API driven profile badges allowing any user to take advantage of the power of an API, and grow a healthy API embeddable strategy.

Widgets
Widgets are highly functional, API driven JavaScript tools that provide portable applications that can be embedded on any website or application. API widgets provide tools any API user can deploy, and developers can reverse engineer, modify and extend to meet their needs. Widgets really establish an advanced API embeddable strategy and can deliver the value of an API across the Internet.
Environment

APIs are all about development and with valuable resources and assets on the line, API owners should consider the need for separate environments for partners and developers to build on.

Sandbox

With the sensitive information available via many APIs, providing developers a sandbox environment to develop and test their code might be wise idea. Sandboxes environments will increase the overall cost of an API deployment, but can reduce headaches for developers and can significantly reduce support overhead. Consider the value of a sandbox when planning an API.

Production

When planning an API, consider if all deployments need to have access to live data in real-time or there is the need to require developers to request for separate production access for their API applications. In line with the sandbox building block, a separate API production environment can make for a much healthier API ecosystem.

Developer Account

Developers will need tools to manage their API usage and environment. Right after a developer registration, they should get access to a series of building blocks that will assist them in managing their platform usage.

Reset / Forgot Password

The need to reset an account password access is pretty standard operations for any online platform. Provide the necessary tools for developers to gain access to their account if they lose their password.

Dashboard

Much like the landing page for the entire API platform, developers should have a single dashboard for getting at all their tools, metrics and information they need to successfully manage their usage.

Account Settings

Along with password reset, access to their basic account detail and settings is standard operating procedure for any platform. Don’t make developers look for their settings, give quick access to settings and allow for easy updates.
Usage Log & Analytics

Rate limiting will be part of any API platform, without some sort of usage log and analytics showing developers where they stand, the rate limits will cause nothing but frustration. Clearly show developers where they are at with daily, weekly or monthly API usage and provide proper relief valves allowing them to scale their usage properly.

Billing History

Obviously if an API is entirely free, billing history is not necessary, but if any tier of API requires paid access, provide clear and easy access to what a developer has been billed, allowing them to access and download their billing history.

Multiple Application or Key Manager

Many popular APIs are becoming application centric and provide developers with tools for managing multiple applications or development projects. API owners should consider how developers will be building applications on top of an API and consider that many will need multiple access keys for their separate applications or user groups.
Conclusion
That concludes a walk through of 73 of the most common building blocks used by 1500 of the top public API providers, who are transforming almost every top business sector. Not all these building blocks are applicable to every API, but each one should be considered when planning a new API or deploying the next version of an existing API.

The goal of this whitepaper is to not to provide the details of implementing each of these building blocks, but to highlight the full spectrum of building blocks used by the successful APIs on the Internet today. These building blocks go by different names in various API ecosystem, but common names are applied in this paper.

Developers have grown familiar with consistency across many of these public APIs. Prior to this paper, there is no consistent framework for the business building blocks of an API. The patterns found in many of these API areas is an organic process that has happened from waves of API owners emulating the successful APIs prior to them. This cycle reflects the organic nature of the web API movement which started in the early 2000’s and has picked up significant momentum by 2010.

The Building Blocks of a Successful API is not meant to define a standard for API areas, but to shed light on what approaches are working in the space. There are several appendixes provided with this paper, to assist with planning, designing and deploying successful APIs.

The underlying message of this paper is that the technical building blocks are only the beginning of a successful API, after that proper business and legal building blocks, as well as an energetic evangelism strategy are critical to a successful API.
Appendix 1 - Building Block Checklist

- Landing Page

Technology

- REST or Pragmatic REST
- JSON
- XML
- Basic Auth
- OAuth
- oEmbed
- WebHooks

Onboarding

- Self-Service Registration
  - Twitter
  - Google
  - Facebook
  - Github
- Getting Started
- Best Practices

Docs

- List of Endpoints
- Documentation
- Language Specific Documentation
- Error Response Code
- Explorer
- Interactive Documentation

Authentication

- Overview
- Tester
Code

- Code Samples
  - PHP
  - Python
  - Ruby
  - JavaScript / NodeJS
  - Java
  - C#
  - VB
  - ColdFusion
  - Perl
- SDK
  - PHP
  - Python
  - Ruby
  - JavaScript / NodeJS
  - Java
  - C#
  - VB
  - ColdFusion
  - Perl
- Starter Projects
- Platforms
  - Drupal
  - WordPress
  - SalesForce
- Application Showcase
- Github / Gists
- Open Source

Mobile

- iOS
- Android
- HTML5
  - PhoneGap / Trigger.io
  - Tiggzi

Self-Service Support

- FAQ
- Forum
- Stack Exchange
Direct Support

- Office Hours
- Email
- Phone
- Tickets

Communications

- Blog
- Twitter
- LinkedIn
- Facebook
- Google+
- Email Newsletter
- Calendar

Updates

- Roadmap
- API Status
- Change Log

Service Levels

- Service Tiers
- Pricing
- Rate Limits

Monetization

- Affiliate Program
- Advertising

Resources

- How-To Guides
- Case Studies
- Webinar / Videos
- White Papers
Research & Development

- Labs
- Ideas
- Opportunities

Legal

- Terms of Use
- Privacy
- Branding

Embeddable

- Buttons
- Badges
- Widgets

Environment

- Sandbox
- Production

Developer Account

- Reset / Forgot Password
- Dashboard
- Account Details
- Usage Log & Analytics
- Billing History
- App Manager
- Key Manager
- Delete Account
## Appendix 2 - List of 30 Top APIs for Review

<table>
<thead>
<tr>
<th>Logo</th>
<th>Service Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Amazon Web Services" /></td>
<td>Amazon Web Services</td>
<td>Cloud Computing</td>
</tr>
<tr>
<td><img src="image" alt="Bitly" /></td>
<td>Bitly</td>
<td>URL Shortening</td>
</tr>
<tr>
<td><img src="image" alt="Box.net" /></td>
<td>Box.net</td>
<td>Cloud Storage</td>
</tr>
<tr>
<td><img src="image" alt="Compete" /></td>
<td>Compete</td>
<td>Web Analytics</td>
</tr>
<tr>
<td><img src="image" alt="Constant Contact" /></td>
<td>Constant Contact</td>
<td>Small Business Marketing</td>
</tr>
<tr>
<td><img src="image" alt="Datasift" /></td>
<td>Datasift</td>
<td>Data Marketplace</td>
</tr>
<tr>
<td><img src="image" alt="Disqus" /></td>
<td>Disqus</td>
<td>Comments and Discussions</td>
</tr>
<tr>
<td><img src="image" alt="Dwolla" /></td>
<td>Dwolla</td>
<td>Payments</td>
</tr>
<tr>
<td><img src="image" alt="Ebay" /></td>
<td>Ebay</td>
<td>Products and Online Auctions</td>
</tr>
<tr>
<td><img src="image" alt="Etsy" /></td>
<td>Etsy</td>
<td>Handmade Marketplace</td>
</tr>
<tr>
<td><img src="image" alt="EventBrite" /></td>
<td>EventBrite</td>
<td>Event Management</td>
</tr>
<tr>
<td>App</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------------------</td>
<td></td>
</tr>
<tr>
<td>EverNote</td>
<td>Memory and Notetaking</td>
<td></td>
</tr>
<tr>
<td>Factual</td>
<td>Data Marketplace</td>
<td></td>
</tr>
<tr>
<td>Flickr</td>
<td>Images</td>
<td></td>
</tr>
<tr>
<td>Foursquare</td>
<td>Locations and Check-Ins</td>
<td></td>
</tr>
<tr>
<td>Full Contact</td>
<td>Contact Intelligence</td>
<td></td>
</tr>
<tr>
<td>Google Maps</td>
<td>Mapping</td>
<td></td>
</tr>
<tr>
<td>Google+</td>
<td>Social Network</td>
<td></td>
</tr>
<tr>
<td>Instagram</td>
<td>Photos</td>
<td></td>
</tr>
<tr>
<td>LinkedIn</td>
<td>Business Social Network</td>
<td></td>
</tr>
<tr>
<td>RapLeaf</td>
<td>Email Contact Intelligence</td>
<td></td>
</tr>
<tr>
<td>Salesforce</td>
<td>CRM and Sales</td>
<td></td>
</tr>
<tr>
<td>Logo</td>
<td>Service</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td><img src="sendgrid-logo.png" alt="SendGrid" /></td>
<td><strong>SendGrid</strong> - Email</td>
<td></td>
</tr>
<tr>
<td><img src="stripe-logo.png" alt="Stripe" /></td>
<td><strong>Stripe</strong> - Payments</td>
<td></td>
</tr>
<tr>
<td><img src="echonest-logo.png" alt="The EchoNest" /></td>
<td><strong>The EchoNest</strong> - Music</td>
<td></td>
</tr>
<tr>
<td><img src="twilio-logo.png" alt="Twilio" /></td>
<td><strong>Twilio</strong> - VoIP and SMS</td>
<td></td>
</tr>
<tr>
<td><img src="twitter-logo.png" alt="Twitter" /></td>
<td><strong>Twitter</strong> - Micro Blogging</td>
<td></td>
</tr>
<tr>
<td><img src="wolframalpha-logo.png" alt="Wolfram Alpha" /></td>
<td><strong>Wolfram Alpha</strong> - Computational Knowledge Engine</td>
<td></td>
</tr>
<tr>
<td><img src="xero-logo.png" alt="Xero" /></td>
<td><strong>Xero</strong> - Accounting</td>
<td></td>
</tr>
<tr>
<td><img src="zappos-logo.png" alt="Zappos" /></td>
<td><strong>Zappos</strong> - Shoes and Clothing</td>
<td></td>
</tr>
</tbody>
</table>