Developers Hate Marketing

Attract Developers to Your API

apigee

Contents

Introduction

If you are starting a developer program and want to be successful in reaching developers, this ebook is for you.

Ten years ago, only big platform companies like Oracle or Microsoft had developer programs. Five years ago, companies with developer programs were at the leading edge. Today there are hundreds of developer programs and so many apps and APIs that it is much more difficult to stand out.

Building the apps and APIs is no longer the hard part. The hard part now is marketing the apps and the APIs. Attracting your channel partners—the developers who will build apps for customers using your API—is the real challenge.

Add extra spice to this challenge: Developers hate marketing. This is the dilemma you face when architecting your developer program.

We wrote this ebook to help you squarely face the challenge of drawing developers to your API and to avoid the many common mistakes that companies make when designing their developer programs.

The principles in this ebook apply broadly to drawing developers to all types of developer tools, but will focus on API programs, including public APIs where you are trying to attract developers you don't know, partner APIs where you are trying to attract partners to build on your API, or internal APIs used only by the organization's own developers. You might suppose that marketing to internal developers is less work than reaching unknown developers, but the fact is that many of the same principles apply. And if you want a successful developer program, you'll need to learn these principles.

Regardless of the type of API you are pursuing, the economic benefits of attracting developers are compelling. Customers and partners are moving off the browser and onto apps. The critical link in this process is developers. Developers carry your products or services to customers through apps. You may not have thought of them this way, but they are your channel partners, your indirect sales team, your value-added resellers.

If developers have the tools to innovate, a way to get recognition for their work, and a means to get paid, an API can spark amazing innovation. Engaging developers, making them happy, and making them successful is your goal.

To take advantage of this channel, an API must be delivered reliably, be scalable, and offer supporting technology for experimentation, testing, and monetization. Marketing and supporting that API, once it is created, is as much a social marketing issue as a technical one. This ebook focuses on the social aspect of improving your API. Simply put, developers have to know that your API exists, that it has compelling value, that it is designed to support their needs, and that it is ready to go.

Which brings us to the central issue: having designed an API, how do you bring it to market and attract developers who will use it to create apps that reach your customers? Once you have built it, how can you ensure that developers will come? And not only that: how will you ensure their success?

Marketing to Developers: Everything You Know Is Wrong

The first mistake many companies make with developers is to use the word marketing.

Developers live in the world of the tangible. They want to see their apps used by people. They have a keen eye for anything that sounds like vaporware and have an immune response to traditional marketing approaches and offers.

In order to build a developer community, you must speak directly to developers' motivations:

They want to build their skills. A developer's marketability depends on skills in the latest platforms. Developers reject technologies that they see as dead-ends. Many developer products have failed because they pushed a proprietary technology with limited applicability. Developers gravitate to technology and APIs with a big footprint and that support widely applicable standards such as HTML5.

They want tools to solve their problems. Developers are practical. APIs like Twitter and frameworks such as Node.js are successful because they help developers solve a tough problem, improve their individual or team efficiency, or help them get to market (or to their next project) faster.

They want to get paid. Although many developer communities have a zest for pure innovation, most developers are not just fooling around. The success of the iPhone App Store and Google Play are a testament to the fact that, with almost zero overhead, developers are working to drive a business or an income.

They want recognition. Developers want to see the fruits of their labor in use. They strongly prefer to work either on projects that bring in a lot of customers or that provide peer recognition. The developer world is hierarchical, complete with "superstar" developers that others look to and emulate. Their careers are to some degree dependent on their personal visibility and the visibility of the products they create.

The Implications of Developer Needs

Traditional marketing campaigns won't work with developers—they have an immune response to hyperbole and a traditional marketing approach. That said, developers are also highly collaborative and eager to learn. The company that responds with the right content and tools, approaching the right people, and in a way that clearly reflects that you've done your homework, may be rewarded with the attention of developers. Do it really well and you may get to the next level: adoption and advocacy.

Segment Your Approach

Although they share many of the characteristics we've described, developers are not a homogenous crowd. They can be segmented into smaller groups, for example, by interest, type of application, or preferred platform.

An additional dimension cuts across the level of professional achievement of the developer. There are hobbyists, professional, and enterprise developers. Understand the segment you are targeting. The issues, capabilities, freedoms, resources, and constraints for the lead software architect at Goldman Sachs are vastly different from those of a hobbyist working from home, or an independent app development agency owner.

Highly segmented, individual campaigns are much more successful than simultaneously trying to reach every possible user of your API. It is much more viable and usually more effective to begin a campaign by saying, “I’m going to reach Twitter developers building iPhone apps” than it is to say, “I’m going to reach everyone.” It may be somewhat counterintuitive, but often the more focused your target segment, the more effective your outreach, and the better results you can drive.

Use Influence Marketing: Find the Alpha Geeks!

The developer world has a pecking order. Winning developers means winning the influencers, the “alpha geeks”—top developers who have the respect and admiration of their peers, as well as the capability to reach and to be a trusted advocate for other developers. There is a wide gulf between the quality of work and productivity of the best developers and that of average developers. It is often said that a great developer is as much as ten times as productive and effective as an average developer, so find and work with the truly great developers for the best feedback and results.

Spend some time researching user group leaders, relevant blogs, and community boards in your target segments, and it won’t be long before you figure out who the alpha geeks and community leaders are. They tend to post the most, respond the most to others, and organize activities in real life, which, contrary to popular belief, are essential to developer communities’ continuity. Contact them and reach out to them personally—preferably in-person. Take them for coffee, listen to their feedback, and get to know them. If they are a help to you, give back to their community by supporting or sponsoring the community events that they put together.

How do you win an influencer? It’s not a superficial act. You have to engage them and solicit their feedback. And you have to act on it—usually it’s great advice. At times, alpha geeks can advocate for you by talking about your value with peers and assisting others in building with your product or service. They may evangelize its value online through blogs, tweets, and forum posts. In so doing, you will be rewarded for your attentiveness and receive it in return, from developers who, in many cases, wind up behaving as if they work for you—sometimes with a greater degree of loyalty than some of your employees.

The best strategy *is not* promoting your API to an audience of the willing—talking at them doesn't work. The best strategies hinge upon *listening*, and letting the developer help influence your product and become a part of your extended team. Major software companies have successfully deployed this strategy, hosting events for their top-alpha-geek third-party developers, working collaboratively with them, even issuing them business cards.

The same approach can be used with other influencers. Remember, building an API community is as much about authentic person-to-person contact as it as about online social networking outreach. A great example is Yelp, which built up its core community by hosting live events in the cities where it brought together its Yelp elites to meet each other and build relationships that were reinforced online.

Pay Attention to Complementary Products and Programs

It pays to identify other APIs and developer communities that your target developers frequently use—these might be great partners. If your developers are building apps that use the SoundCloud or Salesforce APIs, you might consider participating in their activities or showing up where they do.

If the developers you are targeting are used to working with code on GitHub, your sample code and SDKs should also be on GitHub. Developers expect that you will answer questions not only on your own developer forum, but also on Stack Overflow. If you think there are synergies with other content providers, reach out to them and team up on events, outreach, content, and code samples.

A Developer Program's Hierarchy of Needs

Based on the discussion so far, it's clear that this is a different type of marketing with a different audience and different techniques. It can be overwhelming if you don't have a way to structure your approach and to determine what the most important parts of the program are.

What about Hackathons?

Hackathons are a currently very popular type of developer event where developers team up to create 'hacks' or prototype apps—usually for prizes and the chance to demonstrate their creation to the event attendees. Their 'hacks' may or may not use your technology, depending on how the event is structured. The awareness of hackathons has increased dramatically in the last couple of years and now there are hackathons in many cities on just about every weekend with just about every type of developer product or content.

Hackathons are a great place to get your feet wet engaging with developers, but you have to know what your end goal is. Hackathons are probably the best place to get on-the-ground feedback from developers on your product and developer experience. You'll see just how easy it is for a developer to register, learn, and use your API under pressure to build something quickly.

That said, hackathons (and events in general) are just one part of a well-balanced developer strategy. You also have to do the other things that we outline in this section. You might not get many registered developers or actual apps built with a hackathon-only strategy. More on hackathons later in this ebook.

Your developer program has a hierarchy of needs (see Figure 1). Think about creating a balanced and prioritized architected program that is built on a strong foundation. Let's look at the most important parts of a developer program, starting at the foundation and moving up.

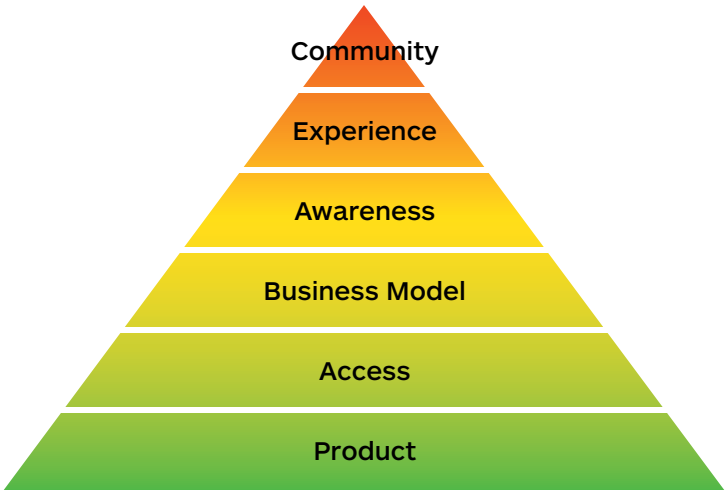

Figure 1. Developer Program Hierarchy of Needs

Fundamentally, your program must provide:

- A great **product**.
- **Access** to that product (and to you).
- A **business model** that incents and rewards developers.
- An **awareness** campaign so that developers know your program exists.
- A great **experience** for developers.
- A **community** where developers can collaborate and help each other.

The Foundation: Developers Want a Great Product

Think of your API as a product. Fundamentally, your API program is not going to do well if you don't have a great product. Most importantly, you need to offer real value to the developer. There are thousands of APIs—why should they learn about or use your service rather than the alternatives (whether a competitor's product or something they build themselves)? Here are a few core principles to keep in mind:

- **Less is more.** Often the more features you have, the harder it is to communicate the unique and differentiating value of your API, and the harder it is to do a great job functionally meeting the key use cases. Don't throw everything but the kitchen sink into your API.
- **Keep it focused.** Sometimes it can help to hone in on a particular application type and design the API around it. If you say, "We must be everywhere, so we'll target every language, and every kind of app," you're likely to have a hard time appealing to any one particular developer or app segment.
- **Make it cool.** Your product must be cool! It must be something that developers want to build with. Of course, what's cool to some people is not cool to everyone, but your API should be cool to your target developers, whether they are analytics geeks or 3D game developers.
- **Design from the outside in.** The best designs approach the problem from the customer's perspective, not the perspective inside the four walls of your company. Your API should be intuitive. Your API design should read like a book so that developers can grok what it does and how it behaves without cracking open the documentation.

- **Hold their hands.** Support is critical. At some point, every developer will have questions and no product addresses every need perfectly. When that happens, your developers want swift attention and notification of what's happening. Even if you don't know the answer or have a fix, respond immediately to let them know you are on it.

Sure, some products are so great that developers will put up with a bad experience, but they'll still complain. Even though it had one of the largest developer communities and support structures in the world, Facebook was excoriated for "bugs, poor documentation, never-ending API changes, slow response times, and other headaches." Facebook recognized that making their developers more successful would make Facebook more successful, and so they responded by redoubling their Operation Developer Love efforts.[1]

Segment Your Developer Audience

Before you go much farther with your API program, figure out who you want to address. You don't have unlimited resources to pursue the entire universe. The more focused your marketing, the more relevant it will be. Sometimes it's better to create a fantastic experience for 100 people than an average experience for 10,000. Once you have a core group, you can expand, one use case or segment at a time.

Developers are segmented in several ways:

- **By the technology that they use.** Many developers are loyal to their platform. Figure 2 shows the relative size of different developer communities.
- **By the kind of company they work for.** Developers work for different types of companies, such as banks, some of them work for startups like Flipboard, and some work for themselves. That said, many developers have a day job and a project on the side and switch companies regularly, so treat every developer as a potential customer down the road.
- **By the kind of API strategy you have.** Are you creating public apps for third-party developers or partners? Private apps for employees?

1 Rik Myslewski, "Facebook upgrades Operation Developer Love," The Register, July 27, 2011, http://www.theregister.co.uk/2011/07/27/facebook_developer_upgrade/.

How Many Developers Are There?

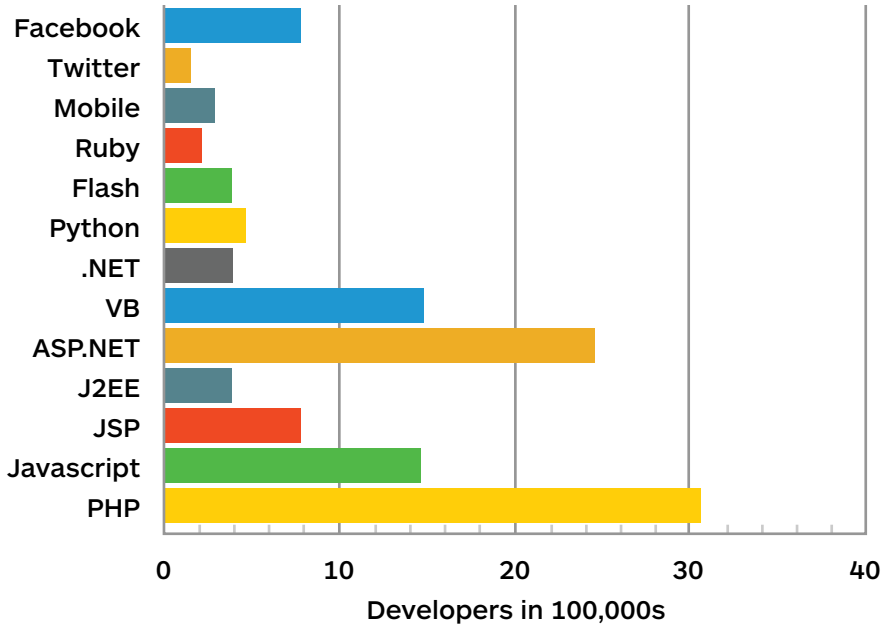

Figure 2. Relative size of developer communities
(Source: Apigee, "How to Segment Developers for Your API Strategy")

How Many Developers Are There?

How can you estimate how many developers are in your target segment? It may be difficult to find a specific market sizing or study, but you can often piece together evidence by looking at a couple of different sources that can act as proxies for a full market sizing.

Start with the web. Many web and enterprise software companies from Microsoft to Oracle commission their own developer studies and surveys and cite their findings occasionally. Search for them.

You can also get creative in using other information as a proxy to understand trends. O'Reilly Media publishes a 'heat map' of annual book sales on different topics that shows the rise and fall of tech trends.

Finally, there are specialty analyst firms that perform developer research, such as Evans Data.

Access to Your API: Is It Fast and Free?

APIs are all about achieving better information exchange between parties. With that in mind, access should be fast and free. Many companies have APIs that are difficult to get to and that you must pay to use. It's really important to provide developers with an easy sign-up process that shows some subset of the API's functionality in action—immediately—so they can see that your product meets their needs in minutes.

Here are some suggestions for streamlining access to your API:

Make sign-up easy. Rule number one: Smooth sign up for immediate access. A good rule of thumb is that a developer should get some instant gratification within minutes as an incentive to keep going.

Make something free. Charging for access to an API or developer program will introduce major friction to adoption. Or, for those programs that are required to charge for access, successful ones make some functionality free to enable developers to understand how the API works and the value of the data or service. Sometimes, creating the perception that you are offering something for free (such as a month's free access) can reduce barriers to adoption.

Don't make authentication a barrier. Authentication is often the hard part of using an API. Consider providing one endpoint that doesn't require credentials so that developers can try before signing up.

Make documentation accessible. Make sure your API documentation is complete and straightforward. Try one of the interactive API documentation consoles, where developers can test API calls and see the results. There are lots of creative solutions that make documentation more tangible and interactive, among them the open source Swagger or the free Apigee API Console that you can embed on your developer portal (apologies for shameless self-promotion).

Make terms of use clear. Make sure your policies complement your accessibility. Be very clear about your policies and careful about changing terms. Developers often shy away from products that are cagey about terms or seem to hint that they may change in the near future, or without warning. Your terms should address issues of data ownership, "what happens if I close my account?" and privacy.

Be transparent. There are two dimensions to transparency. On the technology side, if you have rate-limiting policies for API consumption, display them clearly, like Twitter does. On the communication side, if there's a problem, put a notice front and center and make sure developers are in the loop. Solid examples of this in action are API status pages from Amazon and Salesforce.

Provide access to you. Send a welcome email with a name and phone number. Be sure it lists a person, not a generic email address like feedback@company.com. Use Google alerts to monitor blogs and newsgroups for mentions of your API. Participate on Stack Overflow and respond quickly to tweets. The person who answers developers should be responsive, engaged, and knowledgeable. Again, you don't even have to know the answer right away, but it is critical to respond and acknowledge a developer's problem immediately. It won't hurt if the person is an extrovert, because building communities means meeting up in the real world as well as online.

Developers Want to Get Paid: Have a Business Model

Developers are in essence business or channel partners. They're creating the apps that reach your customers (or users for internal APIs).

Developers need a way to get paid. How does your business model help them get paid? Figure 3 shows API business models circa 2005.

In 2005, most API business models fell into one of four categories. While the free model is still the same, the other three types of business models evolved into many more specialized categories. John Musser of the ProgrammableWeb has an expanded classification of API business models circa 2013 in his Slideshare presentation.

API Business Models, circa 2005

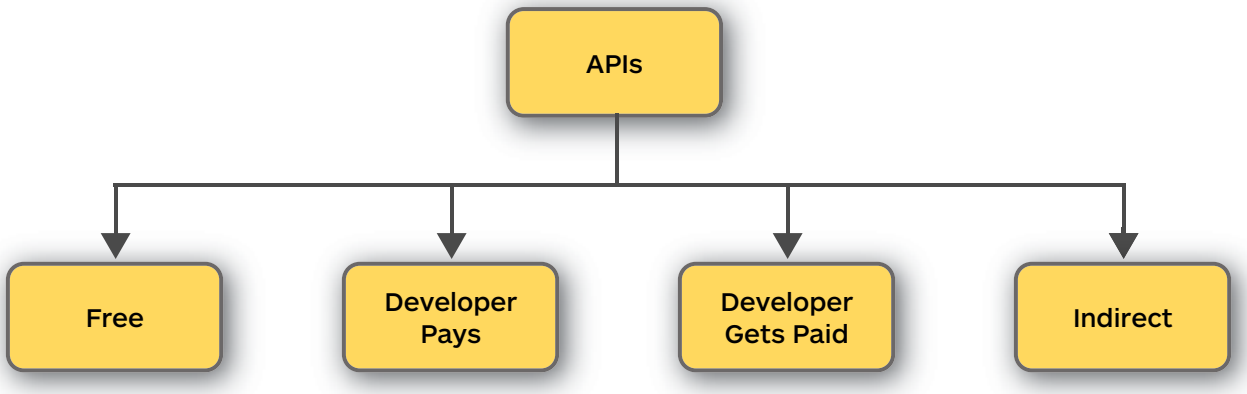

Figure 3. API business models evolved considerably from 2005 to 2012 (Sources: John Musser at ProgrammableWeb, http://www.getelastic.com/20-api-business-models-deconstructed/)

What kind of business model should you use? Well, it's your business and you know it better than any vendor—ultimately you must answer that question based on your objectives. APIs: A Strategy Guide (O'Reilly 2011) presents more detailed frameworks for this analysis.

Developers Need to Know Your API Exists: Awareness

If an API is built and no one knows about it, does it exist? Don't start an API program unless you have a plan for raising awareness (notice we didn't say "marketing"). And make one of your first, critical hires your developer evangelist. Here are a few channels for getting the word out:

- **Public relations firm.** You have to get the word out! Your PR firm can help. Point them toward the numerous tech blogs and pubs that developers read (ReadWrite, ProgrammableWeb, and GigaOm: the list goes on). Corporate PR departments sometimes err on the side of caution, protecting their brand. But for APIs, this is not always good thing because the API blogs and pundits will write about your API anyway; you want to make sure that you communicate all the great things about your product and why it's different. Building awareness yourself helps ensure that the information being disseminated is correct.

- **Content distribution.** Most developers will discover you on the web, so make sure you can be found through Google easily. Also, think beyond the walls of your organization and reach out to other development communities—content distribution is not all about your developer portal (but we will talk about your portal in the next section).
- **Social media and forums.** Besides GitHub and Stack Overflow, social media such as Facebook, Twitter, Google+, Slideshare, YouTube, GetSatisfaction, Quora, and LinkedIn are important channels where developers expect to be able to get content and answers as well as a good place to find kindred spirits and likely partners. In addition to your own social channels, get involved in forum debates, and when you do, be sure to deliver a quick response if someone addresses you or asks a question. It goes without saying that a respectful, authentic tone is a best practice—and this goes triple with developers—as is admitting mistakes and recognizing product deficiencies when pointed out.
- **In-person, at events.** Great evangelists get out of their offices and solicit feedback at developer events, including meetups, hackathons, seminars, conferences, and workshops. It is at these "real" events where you build key relationships.
- **Through other developer communities.** Building developer community doesn't always mean building your own community, all by yourself. Instead, plug into other communities where developers already participate. All of these programs thrive on content and contributions from across the industry. Great communities to know about include:
  - Developer water coolers: GitHub, Stack Overflow, Hacker News, Reddit
  - Prominent vendors and platform communities: Microsoft, Facebook, IBM, Oracle, Twitter…
  - Programming language communities: Node.js, Jquery, Ruby…

Make sure you participate in GitHub and Stack Overflow. These sites have huge developer audiences and developers expect your content and participation there. You want to be where your target developers already hang out.

Add relevant meetups and other events. It's important to be on the ground at a variety of events, reinforcing the relationships you're developing online.

Developers Need a Great Experience

It may seem obvious, but it bears repeating and a little explanation: your API portal needs to deliver a great experience from beginning to end.

Do developers have all the things they need to be successful? Developer resources and tools? Do you offer great SDKs for the common platforms? Do developers have access to other developers so that they can get answers to questions? Is there a place to search FAQs? Is it easy to find terms of service?

There's an intangible side to experience as well. When you ask developers to talk about what makes their experience with a particular API great, the answer is often not what but who. The feeling that people are behind their success, reaching out to them, and pulling for them is an important element of the developer experience.

Onboarding is a key element of the developer experience as well. The first step shouldn't be the hardest. A great onboarding experience can be a differentiator. The sign-up process should be super smooth.

Developer Portal and Experience

Developer portals are part of the experience. They encompass a variety of elements, including:

Information about the offering. What is the API designed to do? What are the terms for using it? What business assets does it expose? What is the process for getting started? What are the benefits?

Documentation. The Twitter API portal leads you straight to documentation where you can learn more, clearly laid out for exploration and headlined to attract developers.

SDKs and code samples. Year after year, in developer surveys, developers consistently cite SDKs and code samples in the top three resources that make them successful. Invest in providing great, well documented and well commented code samples that showcase how to use your API, and make sure they appear in your SDK and documentation.

An easy way to try the API. Public APIs make registration fast so that you can try out API calls using your browser. Instagram's developer portal has an Apigee API Console that enables you to try out commands.

App gallery. Developers looking for ideas can look at the app gallery on most developer portals. Foursquare's (https://foursquare.com/apps/) gallery has a wide range of sample applications. Galleries help developers (and business owners) get ideas for using the API in interesting ways.

Means of contact. There should be an easy way for the developer to get in contact with the API team. Make sure that a responsive person is on the other end so that developers immediately feel that personal touch reaching back to them.

Ways to meet other developers in person. All this online and virtual stuff is supported by face-to-face meetings. In addition to conferences, there are less formal meetings where developers can meet and get ideas and brainstorm with the API team.

Information about the business model. Developers want to know the details of the business arrangement and the rules for using branding. With private APIs, it could be a formal partner agreement with a series of steps and legal documents. With public APIs, it can be as simple as agreeing to terms of use.

Developers Need a Community

Ultimately, you want to establish a community dynamic to create leverage for your community manager or evangelist to work with hundreds or thousands of developers. Your first evidence of that community dynamic will be when someone who doesn't work for you helps someone else who doesn't work for you. At this point, nurture that mutual support, but don't expect that you can lean back and bask in the helpful community you've established. Growing a vibrant community requires spade work and a nurturing approach. Build real relationships with real people, online and in person. Showcase and appreciate what developers are doing, and ask them for feedback about your product. Then act on that feedback and show that you've acted on it.

Events and Hackathons

Part of having a strong community is achieved by having a ground game or participating in events. Go to meetups, to workshops, and to boutique, small, and regional events.

Hackathons are one type of event that has become very popular. Hackathons are coding competitions where teams of developers build a prototype app or hack under a time constraint (usually a day or two), and usually for moderate to substantial cash prizes. Here are some tips for holding a successful hackathon.

- Know your objective. Hackathons can be great for getting deep feedback on your API. They are usually less effective for producing large numbers of developer registrations or sample apps.
- Start internally. Internal hackathons (just employees) are a great way to start a hackathon program before going external. There are lots of benefits from getting internal energy, ideas, and feedback flowing, and an internal hackathon provides a good opportunity to get some practice running these kinds of events.
- Get your feet wet further by sponsoring an existing hackathon before organizing your own. You can usually offer your own prize and set up a table to take feedback from attendees on your API.
- Recruit developers at the event to incorporate your API or product into their hack. Offer them technical support if they do so to help them win. Tie prizes to the outcome you want. Make sure that the rules at your hackathon state that they must use your API in order to qualify to receive the prize.
- Don't forget that recognition is an important prize, especially for internal hackathons. Often, just having the ability to demonstrate their skills in front of peers or executives is enough motivation (but monetary prizes are also common and, of course, appreciated).

Other factors for hosting a great hackathon include:

- Finding great partners to give the event a stronger appeal and offer more products and services for developers to learn about.

- Locating a great venue. Ensure that solid high-speed Internet is in place (have a backup!) and make sure the venue has good audio-visual equipment.
- Ensuring you can arrange workspaces into teams; have conference rooms available if possible.
- Plenty of good food and drink (especially drink)!
- Having help on standby. Make sure that there are lots of technical people available to help developers. If you're looking for feedback, it's especially important to have technical people who can help stalled teams get back up and running.
- Keeping sponsor presentations short, slide-free, demo-focused and aimed at the developer audience (not a marketing pitch).
- Offering short mini-tech sessions during the day.
- Promoting the event through local meetups and interest groups.

Is it Working? Defining Metrics

You can't improve what you don't measure, so we suggest creating a model of the metrics the API can be expected to deliver and how it impacts your business. For example, if your company sells ads, you need to know how the API affects your ad-supported business. If your company sells cars, you need to know the API supports your company's ability to sell cars effectively.

Analytics provide you with information to make future investments and decisions. Bear in mind that information can mean a variety of different things. A surge in traffic might indicate that the API is succeeding, but it could also mean that the API is being used inefficiently, resulting in inflated traffic.

Metric Reality Check: Signups Don't Equal Conversions

It's exciting to have developers sign up to use your API, but if they try it for 10 minutes and never use it again, that's not really active use or a reflection of adoption. You want to make sure that developers can build apps using your API, not just sign up to use your API.

Develop a business metrics framework. At minimum, you'll want to look at page views, traffic sources, the number of conversions on content or other measures of loyalty, the number of developers registered on your site, numbers of apps built and in production, and the number of requests made to the API.

Here are some things that we have found helpful when setting up metrics on API programs:

- **Get early buy-in on the top 3.** Focus on one to three top-level strategic metrics and get early, wide agreement from all parts of the team: the sponsoring executive, project manager, engineering head, business development, and operations manager. If different stakeholders measure success using different metrics (say, number of developer sign-ups vs. API traffic vs. revenue), this can pull resources in different directions.
- **Track against realistic projections.** Set expectations early by modeling anticipated results, and then track actuals against this estimate. For a private API, you might project that mobile product teams can innovate on their products 50% faster than prior to the API. For a public API, you might make projections for community engagement, such as 20% of registered developers might build an app, 10% of those apps might drive ongoing traffic, each of those apps might drive a certain volume of traffic, and so on.
- **Publish a weekly dashboard.** Proactively highlight product updates and community activities that do or don't change your key metrics so that you can quickly adjust tactics and think of new ideas to improve the metrics.

Set Realistic Expectations for Your Developer Program

Metrics are very important, as we've stated. You need metrics to show you where you are and where you're going, and how well your program is tracking.

At the outset of your developer program, make sure you set expectations correctly. Model your API program and business like any other channel or sales pipeline.

Consider the experience of one corporate CEO. When he was in college, he was a door to door salesman during the summers. Although his product was good, 19 out of 20 times, the door was slammed in his face. Even the best products might have a sales conversion rate of less than 10%.

Similarly, it's unrealistic to think that every developer who visits your site will sign up for your API program.

Model a realistic conversion rate for how many signups it will take to bring about 10 trial apps, and then completed apps. And then how many of those apps will be good, compelling, successful, or well received. If you do the math down a pipeline you can start to see how much traffic you need to drive and how many apps you might expect over a certain time period. Making your developer program successful will take time and effort—make sure you set expectations accordingly so you can invest wisely and communicate progress effectively to your stakeholders.

The Inverted Pyramid: Unclogging Your Funnel

To attract developers and get successful apps written, you have to actively and aggressively run your developer program like a sales-pipeline driven business. Many businesses picture sales like a funnel of prospects who they seek to convert into active customers.

The point of a funnel is that business start out with a large number of prospects and a few of those ultimately turn into successful sales. A developer program has a similar funnel, but it has different stages:

- Visits to your developer portal. Developers find your developer portal, whether from a site online, by searching, or by direct navigation.
- Signups. After getting to your portal, developers sign up for an account.
- Trials. Developers download your API.
- Completed apps. Of those developers who download your API, some complete an app.
- Production apps. Of those completed apps, a few will be apps that you can point to as examples.

Developer Reach

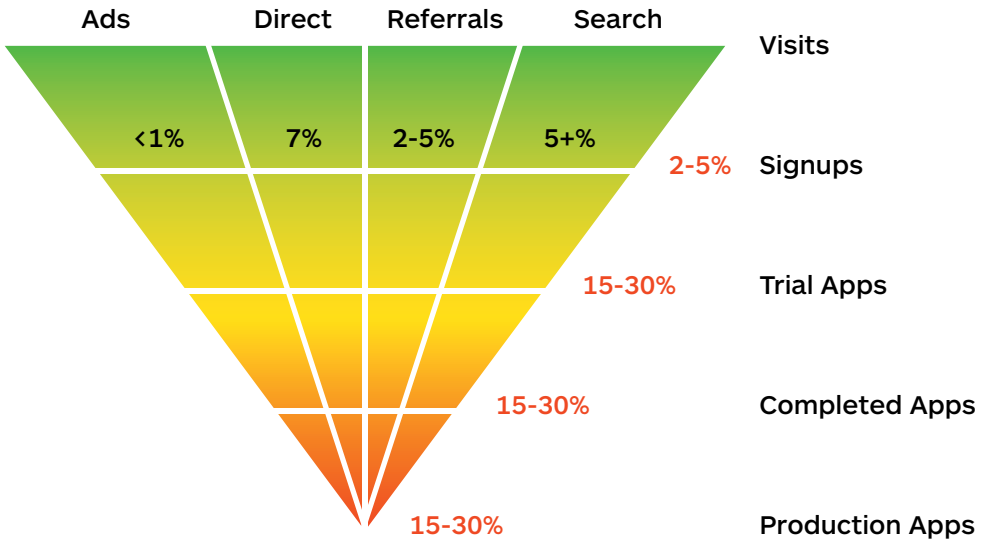

Figure 4. Developer adoption funnel

Funnel analysis has a number of dimensions and conversion rates between stages.

Of all those who visit your developer portal, how many sign up? A typical conversion rate between visits and signups might fall between 2 and 5 percent. Of those who sign up, how many will actually use your API? In other words, how many sign up and then don't use their account? A typical conversion rate for trials is perhaps between 20 and 40 percent.

Next, of those who download your API, how many complete an app? Typical conversion rates are again between 20 and 40 percent, but many of those, bear in mind, are test apps. Lots of developers create apps and name them something like "test me" or "experiment." You know those apps aren't going to be the next Angry Birds.

Finally, of those completed apps, how many wind up in the app store or are something that you can point to as an example of how your API could be used? Typical conversion rates here are between 15 to 30 percent.

As always, your mileage may vary, but the important thing is to think about this model as a way to predict expected results and understand the key levers.

Getting Apps Is Going to Take Too Long!

If your developer program is in its early stages, you might well think it's going to take too long to get some finished apps.

In that case, you might want to jumpstart some apps, hiring developers to create apps using your API to get completed apps into the app store that can act as showcase apps and examples for other developers.

Having given you some ballpark figures, go and analyze your own funnel and see what your actual conversion rates are between stages. While those percentages may not be what you are hoping for, it's important to take the time to find out what they are. You can then start applying some pressure to your funnel.

You may think that the only way to get more successful apps built is to drive an outrageous amount of additional traffic to your portal. You might think, how many hackathons, how many TechCrunch articles, how many meetups will it take to get enough developers to the top of the funnel?

Since that would obviously take a lot of work, you might want to start thinking about how to get creative to improve your outcomes. How can you apply pressure to one of the levels of the funnel directly?

As you examine the flow through your funnel, where is there friction that is hindering progress with adoption of your API?

Driving Traffic to Your Portal

Of course you want more visits to your developer portal. How are those visits coming? It's time to do some analytics.

Are visits coming through PR, like TechCrunch or ProgrammableWeb mentions?

Are they coming via Google? If you have a new API, you should make sure that your portal is SEO-friendly. You can do this in part by putting information on your site that is relevant to your topic area, to increase your relevance in the search results. If you have a photo sharing API, you want to rank on the first page of search results for those keywords.

Are developers coming through direct navigation? Do you ask them, during the signup process, how they heard about your API? (Make that question optional to speed up your signup process, but ask it nonetheless. Letting them check off a source is also more likely to get an answer to the question.)

Are they coming through ads you've placed across the web? Advertising can be another source of traffic to your portal, though, again, since developers hate marketing, ads have to be targeted and executed in just the right way.

If you really want to geek out, you can figure out which of these sources is driving the most high value traffic you get, in other words, leading to developers who create active apps. This analysis can show you where you should invest more effort, whether in PR to get articles written, in SEO to improve your rankings, in word of mouth through events and the like, or in advertising. You may find that advertising converts poorly (people clicking on ads may just have a lot of time on their hands) while word-of-mouth converts very well (the people at developer meetups are more likely to be active developers working on projects).

Driving Signups

Do you have a large number of visits, but no signups? This could mean that you need to work on your portal and your forums, particularly on the personal aspect (is there a person they can contact?). Take a hard look at your signup process; is it fast and easy, or slow and hard? Is your messaging clear or clear as mud? You might need to get an outside perspective if you are too close to be able to evaluate your portal objectively.

Driving Trials

Do you have a large number of signups, but developers don't follow through and use the API? In that case, check the obvious, then consider adding interactive doc that lets developers test API calls right from your portal without downloading anything. This will give them a feel for how easy your API is to use.

Increasing Completed Apps

Do you have a large number of trials, but not many completed apps? Do you have the analytics that show repeated app traffic (telling you which apps are production or successful)? A low conversion rate here could indicate that you need to add more sample code on your portal and to showcase the work of other developers to offer inspiration for cool things developers can do with your API.

Increasing Showcase Apps

Do you have as many apps in the app stores as you would like? Get to know the developers who are building to your API and understand what it would take to be able to get them over the finish line. Once they do, ask if you can blog about or otherwise showcase their app as an example and inspiration to other developers.

Give Away Some Ideas

When new products arrive in the marketplace (and remember: your API is a product), people often don't understand them right away. Although good ideas may come from the outside, the people in your organization may each have 10 great ideas about how to use your API. Start getting these ideas out into the marketplace to drum up interest in your API. Ask internal folks to consider building an app or handing their ideas to an interested developer and coaching that person through building the app.

It's Not a Science Experiment

If you're trying to free up your funnel and get more apps built, don't use the scientific method of changing one thing at a time. Although you want to figure out what works, it's more important to get it working. Look at particular areas where your funnel may be stuck and apply pressure there.

Conclusion

The advice in this eBook is based on our experience with hundreds of API programs, from designing APIs, to launching developer programs, to bringing new APIs to market. What has worked for others may not work for you—every successful API program must find its own path to success. However, hopefully some of the lessons learned in this book will make your journey smoother. Good luck!

About Apigee

Apigee is the leading provider of API products and technology for enterprises and developers. Hundreds of companies including AT&T, eBay, Pearson, Gilt Groupe, and Walgreens use Apigee to reach new customers and drive innovation through APIs, apps and data. Apigee's API Platform is built on enterprise bedrock and designed to meet the challenges of the new mobile, social, cloud marketplace head-on. The API Platform enables businesses and developers to create and deliver well designed, scalable APIs and apps, drive developer adoption, and extract business value from their API ecosystem. Learn more at apigee.com.

Find Best Practices to Accelerate your API Strategy

Scale, Control and Secure your Enterprise

Build Cutting-Edge Apps and Intuitive APIs

Get End-to-End Visibility into your Business with Apigee Insights