IT IS TIME TO GET SERIOUS ABOUT APIS

Application programming interfaces (APIs) are no longer new to the telecommunications industry, but their true potential as a game-changer remains untapped. When thinking about APIs, the most important thing to remember is that they are not a technology for squeezing revenue out of existing business models. They engender new ones. No company gets it right all the time, but the most successful of them are continuously improving their API ecosystems.

Creating a continuously evolving API program is not easy. It requires a willingness to commit to a strategy and monitor its progress closely. It requires continued analysis and modification, because even the best programs do not start out that way. Perhaps most importantly, it requires embracing APIs as an ongoing part of a larger business strategy, one that eschews simple and familiar monetization models for more complex, but ultimately more rewarding outcomes.

A successful API program serves as the foundation of a platform strategy. Creating a powerful platform in turn depends on tying all of these functions into a repeatable framework that makes platform adoption faster and more efficient.

HOW APIS BUILD DOMINANT PLATFORMS

You do not have to do much digging to find examples from outside the telecoms industry that illustrate the importance of continuously refining an ecosystem. In particular, the following two juggernauts are now squaring off across several markets, and theirs is a battle of platform versus platform.

Amazon

Information technology initiatives, even inside technology-driven companies, are known for going awry. They start and stop. They are altered midstream. Executives buy into them then lose interest. It can be tempting to think of API-driven strategies as something to try out. Maybe it will work, maybe not.

Jeff Bezos, Amazon's CEO, did not take that approach with APIs. A decade ago he issued an edict that said from that point forward, all data and functionality in Amazon projects would be exposed through services. Furthermore, every service had to be built in a way that would allow it to be exposed outside Amazon. It took years of effort, but Amazon learned a tremendous amount along the way, and now the company is a platform. Everything Amazon does, from managing cloud storage to delivering customer recommendations, can be exposed to other internal applications, and where needed, to third-party developers.

The absolute value of Amazon's platform approach is difficult to calculate, but a sample of the long list of seemingly disparate revenue streams beyond retail illuminate the benefits of a service-driven architecture: App Store, Associates, Cloud Drive, Cloud Reader, Payments, and Web Services. As opportunities arise, Amazon can capitalize on them and build out new applications, each with tailored business models, as if it were all planned in advance. This flexibility gives Amazon a tremendous strategic advantage as it moves into new markets.

Apple

From the earliest days of the Macintosh computer, Apple had to expose certain APIs to entice programmers to create software for its operating system. The company was famous for enforcing The Macintosh Way, and equally famous for making changes to developer APIs that appeared arbitrary to third-party developers. Still, Apple needed to accommodate them to keep the operating system alive, and in turn to sell computers.

When the iPhone arrived, Steve Jobs was adamantly against allowing third-party development for the device. Apple's initial pitch to developers was this: Build web widgets for the iPhone. They reacted with a resounding call for deeper access. They wanted to build full applications that could take full advantage of everything the iPhone offered.

Jobs resisted, but the demand could not be ignored. Caring for external developers takes a lot of work, from prepping APIs for release to the outside world, to sophisticated documentation and tutorials, to managing expectations.

The last point is particularly salient with a product that is early in its life cycle. Later versions of the iPhone's iOS provided more functionality by way of more efficient and polished core code. This in turn required changes to the APIs, and brought with it the risk of breaking every application on the device, the content which drove its adoption.

Apple has managed to walk the line between continuous innovation and shepherding third-party developers, primarily by telegraphing as far in advance as possible when APIs are to be deprecated. This gives developers time to make changes to their applications and submit them to the App Store. It also gives them time to share techniques for avoiding pitfalls in migration from the old to the new. The time, money, and effort Apple has put into its ecosystem have given rise to a staggeringly successful profit machine.

TAKING A LIFE CYCLE APPROACH TO PLATFORM DEVELOPMENT

The trailblazing in other industries has already been done, and it has become clear that winning with APIs is about building a repeatable process and structure just as you would for any other part of your business. There are a vast number of processes to implement and best practices to put in place. However, at the simplest level there are three key phases to consider when building out an API program.

Definition is the first phase. This is about goal setting for the business (yes, this is a business) and based on that goal, it is about identifying the value in your business that is critical to achieve your goal. Design is vitally important. How the developers inside or outside your company access your business will play a critical role in the program's success. Simplicity of use, regardless of the complexity of the information being accessed, is a must. Lastly, the deployment model is what makes it a business. Think about security. Who has access and how much access? Think about efficiency. How do you optimize traffic and performance? Think about operations. What do you do when you need to update a service or stop offering the service? The methodology takes on the shape of the business you want.

Definition

You cannot reach your goal if you do not have one. What is your goal? Do you want to extend engagement, broaden your addressable market, increase revenue, decrease product cycle time, create barriers for entry to competitors, or is it something else? APIs can support a wide variety of implementations, from tiered service level agreements (SLAs) and quality of service (QoS) guarantee deals with content providers, to integrated billing arrangements with app developers, to traffic-smoothing incentives for consumers, and beyond. However, you will not find the correct implementation without a clearly articulated business goal.

It can be tempting to think that if you build it they will come. The Long Tail has value, but it will support, rather than drive your strategy. You want to forge relationships with players who are big enough to matter and agile enough to move as fast as your ecosystem will allow. Larger, slower companies may look like good matches for partnering, but unless they are API-savvy, they will burn up too much time trying to get their own infrastructure ready to work with yours. Seek out API-driven businesses.

With a clear long-term goal and intermediate targets in mind, you will have to determine which pieces of your existing functionality, services, and data can be tapped with APIs. You may have more assets here, including private APIs, than you might think. The trick is discerning which components can be modified for use in your API initiative, and which must be rewritten entirely.

As you build your strategy, it is important to remember that with APIs flexibility is the byword. Experiment with pricing levels. Communicate early and often with partners to understand what will help them succeed. You must always be watching, adjusting, and communicating. In all cases, a relentless emphasis on key performance indicators (KPIs) should drive every aspect of your API strategy, and the metrics you gather should be paired with your business goals so that you can make adjustments with confidence.

Design

As you build from existing capabilities and add new ones, be sure your technology choices match your business goals. Your back-end capabilities need to be capable of supporting the type and volume of calls being made by your APIs. Security and compliance measures need to be incorporated into every point on the API round-trip journey.

The protocols you use, the structure and complexity of the APIs and their inputs and outputs will have tremendous bearing on whether and how third-party developers use them. It is important to remember that a technically superior protocol may not always win out. The stamp of approval from a standards body does not guarantee credibility either. Sometimes *de facto* standards win, and which ones dominate depends on the industry. Pay close attention to the standards your most important partners are using. In the API world, developers gravitate toward technologies that other developers are using.

With APIs, the details matter. It is not enough, for example, to support OAuth authentication. Early implementers will likely have embraced version 1.0, while newcomers may have opted for 2.0 or decided to sit on the fence as security concerns are hashed out. You need to know the advantages and disadvantages of each protocol you support, and stay abreast of changes; mindshare can change rapidly.

Supporting developers is critical to the success of your API strategy. If by using your APIs they can achieve their own goals of revenue and visibility, you are halfway there. The first step is to make your APIs work for developers. That means naming them in a fashion that minimizes confusion, designing them in a way that streamlines rather than introduces complexity, maintaining concise scope for each API, and adhering to established API coding best practices. It also means using the API before exposing it to external developers, preferably by bringing in someone who was not part of the API team to use it on a real project.

Lay the foundation then nurture a vibrant ecosystem with documentation, sample code, forums, and other tools that make it easy for them to incorporate your APIs into powerful, useful apps, and iterate your APIs based on what developers are doing with them. Do these things and ecosystem success will propel platform success.

As with your business strategy, in developing a healthy ecosystem around your API it is critical to think of it as tending to a garden, rather than building a house. Sometimes you have to remove and replant. Simple Object Access Protocol (SOAP) is an instructive example. Microsoft developed the SOAP protocol and pushed it heavily for awhile, but over time it became clear that many developers

preferred Representational State Transfer (REST)-ful services. The company listened, and has built a wide variety of tools for RESTful development and translation with SOAP. To Microsoft it was more important to stay relevant to developers than to advance a specific web services technology.

Deployment

The rigor you apply to the business planning and design of your API program applies in the deployment phase as well. Technical performance must be tested against target metrics. Are response times hitting the mark? In simulated peak load situations, is the system scaling up appropriately? Does black hat testing reveal security holes? Here the engineering-driven attention to detail of service provider culture is a competitive advantage. Leverage it to ensure that services on the back end and the APIs in front of them work exactly as planned.

An API platform does not get built once; it is continuously monitored and improved based on developer response, application usage, and evolving business strategy. Access rights, rate limits, security, performance management, and scaling all must be managed diligently.

It is also imperative that as you build a third-party developer community, you communicate clearly about shared goals, so developers can understand and work within your framework. The more they know of both the how and the why of your policies, the more empowered they will be to create apps that showcase your APIs.

The right analytics tools can help you maintain control of API use, and they can help you understand how you are meeting your business objectives. As you gain knowledge of how your API program is progressing against technical requirements and business goals, you can rapidly iterate through the definition, design, and deployment cycle.

THE ALCATEL-LUCENT API LIFECYCLE METHODOLOGY

The Design, Define, Deploy approach is at the heart of the Alcatel-Lucent API Lifecycle Methodology. To find out more about how we can help your organization build a powerful platform business, please contact AE@alcatel-lucent.com.

www.alcatel-lucent.com Alcatel, Lucent, Alcatel-Lucent and the Alcatel-Lucent logo are trademarks of Alcatel-Lucent. All other trademarks are the property of their respective owners. The information presented is subject to change without notice. Alcatel-Lucent assumes no responsibility for inaccuracies contained herein. Copyright © 2012 Alcatel-Lucent. All rights reserved.
M2012053293 (June)

Alcatel·Lucent