Best Practices in Building Internal API Programs

Rapid

Whether their goals are to accelerate innovation, increase revenue, or expand product lifecycles, today's digital transformation initiatives rely on APIs. But for an organization looking to become a leading digital enterprise, simply having APIs is not nearly enough. To fully execute on their digital transformation goals and objectives, leading enterprises need a clear API strategy built on a productized approach to internal APIs.

The following details the key steps and best practices for productizing your APIs. These recommendations provide guidance on defining APIs as products, building the framework to operationalize your API program, and designing a rollout plan.

Objectives for Productizing your APIs

Before diving into how to productize your APIs, it's critical that business and technical leaders establish and agree upon the objectives of their internal API program. This shared understanding and cross-functional agreement will help inform value messaging to internal audiences, the communication plan, and best practices around the operational aspects of the program.

For internal programs the most common objectives include:

- Accelerate innovation
- Speedier time to market (applicable to both internal and external applications)
- Enable collaboration and API discovery across the enterprise

These objectives (among others that your organization may determine) should serve as the core criteria for defining and recognizing successful execution of your API program. With these objectives in mind, project and organizational leaders can begin outlining and planning for how they will contribute to and benefit from this collective approach to an API program which views, operationalizes, and leverages APIs as products.

4 Steps To Take To Define APIs as Products

For many organizations, especially those that have not been software-first companies historically, APIs have been viewed purely as a technical tool. This is particularly prevalent in those organizations which leverage APIs exclusively for internal purposes. Ultimately this is a narrow view of APIs and their utility. To be successful in their transformation into becoming a powerful digital enterprise, organizations need to instill a product mindset when it comes to their APIs - even those that are intended strictly for internal use.

1 Instill a Product Mindset

The first step in building this mindset is to think of developers both as builders (providers in a traditional vendor relationship sense) and consumers (buyers). Of course keeping in mind that an individual developer's role will shift depending on what they are trying to accomplish. The same developer who is adding an API to the organization's catalog one day is very likely also subscribing to APIs created by other developers within the enterprise.

Instilling this mindset of APIs as products forces those APIs builders to clearly define the function and value of their APIs. The clarity derived from these definitions will not only make it easier for developers to understand how they can benefit from using a particular API but will also help inform some of the operational aspects for that APIs search and discovery. A clear articulation of the API's value and function enables that builder to contribute to more informative documentation. It also ensures that that builder can properly describe and tag that API when it is published to an API hub, making it easier for consumers to discover and increasing its chances of that API's adoption.

2 Define Requirements

Next, generate a requirements document. This document should become the source for truth for the objectives of the API program and outline the needs in achieving those outcomes including required tools, budget, and people. This document should seek to outline the function of the program's API and align their value with its top-level objectives. The scope should not be limited to the APIs that are already built, but should also include a view of the net new APIs that will need to be added (along with associated headcount) to achieve the program's goals.

3 Begin in Design

Ensuring API adoption does not start at the categorization and tagging stage, it starts at the earliest stages in the development process with API design. Starting with clear API design principles which are implemented across all of the APIs an organization develops is one of the most critical steps to ensuring an intuitive and consistent experience that helps ensure API adoption. This consistency not only makes it easier (thus incentivizing) individual API consumers to subscribe to and utilize the APIs which are listed in the API hub, but also facilitates more effective collaboration across multiple types of users such as developers and QA testers. This ultimately accelerates build and delivery times and reduces costs attributable to redundant code and processes.

4 Solicit Developer Feedback

Just like any good or service, instilling a product mindset with internal APIs requires a focus on collecting and addressing developer feedback. Prioritizing developer feedback has two key benefits.

- Identify Bugs and Unexpected Behaviors
- Identify and Prioritize New APIs

With a firm grasp on how to think about and build APIs as products, it's time to start looking at how to effectively operationalize their use across the organization with proper resourcing of both personnel and tooling.

Resourcing People and Tools

The responsibility to productize APIs is not solely reliant on the developers who are building them but rather the entire organization to enable the API program with the tools, resources, and structure to realize its potential and meet its objectives. Critically important is the organizational structure that will oversee the API program.

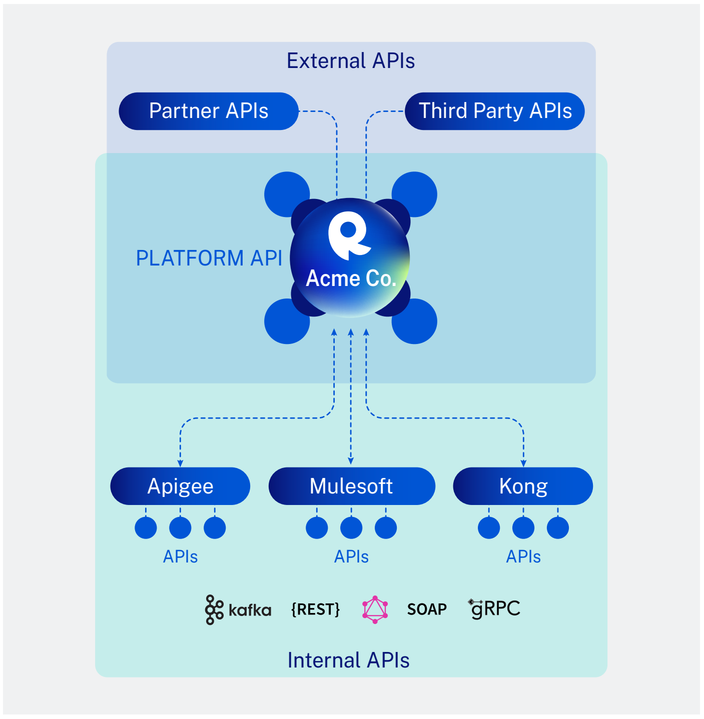
Organizational Structure

There are three common structures which an organization can elect to deploy: single team ownership, horizontal cross-organizational, or a hybrid structure which combines some aspects of the two.

	Pros	Cons
Single Team Ownership Accountability and Autonomy	• Greatest level of autonomy • Creates a sense of ownership • Allows for a greater degree of experimentation	• Can hinder participation from the broader organization
Cross-Operational Investment and Support	• Streamlines interoperability • Leverages specialized expertise • Fosters investment in program success	• Reduces overall agility • Can obscure individual roles and responsibilities
Single Team Ownership Accountability and Autonomy	• Most flexible overall structure • Can take advantage of both efficiencies in decision making and buying power • Allows for both autonomy and collaboration	• Requires clearest boundaries in decision making and responsibility • Can create bottlenecks in decision making

Selecting An API Hub

Having an effective API hub, that enables all of its users, is absolutely essential to ensuring that the goals for the API program are met. This hub is the place where all of your developers and API users should be able to come and access all of the APIs and tools they need to discover and collaborate on APIs to build integrations and applications that will transform the business.



Requirements for an API Hub:

Centralizes All APIs –Creating a centralized place where developers can discover, collaborate on, and reuse all of the organization's available APIs is perhaps the most critical element of what an API hub has to deliver. Without this capability developers are forced to jump from place to place to find all of the APIs available to them, oftentimes not even knowing where all those places might be. This slows the development process, creates redundancy in code, and can ultimately lead to the failure of API programs.

Most enterprises have a number of API gateways comprising their API ecosystem. To centralize all of an enterprise's APIs an API hub must support multi-gateway API ecosystems. To be effective, an API hub must be able to work with this multi-gateway architecture to centralize all of the APIs into a single place. Having this centralized location with all of an enterprise's API regardless of the gateway they run through simplifies the discovery process for developers and helps ensure API adoption.

Optimizes Developer Experience – But that API discovery is only part of the developer experience. A good API hub needs to support the entirety of the developer experience through integrated developer tools for API design and development, to testing and publishing. Having this breadth of capabilities integrated into the API hub allows for the developer to stay within the same context. This delivers a better overall developer experience, and expedites the build and delivery of their APIs. Additionally, this intuitive experience should be designed to facilitate collaboration amongst developers across the entirety of the development lifecycle and support their use of multiple API types including REST, SOAP, GraphQL and Kafka.

Ensures Governance and Access Controls – There are two primary things to look for when assessing an API hub's governance and access controls requirements. First, is its ability to provide a simple and intuitive onboarding experience for those developers subscribing to APIs, one of the most critical elements in ensuring API adoption. Second (but equally important) is its ability to ensure governance
and access controls across the entirety of the API ecosystem. To be effective, an API hub must be able to establish visibility and role-based access controls across all APIs, regardless
of the gateways which they might be running through, and provide clear insights into who is using them and at what level.

Offers Seamless Integrations – When looking at integration capabilities, the most important thing to evaluate is an API hub's ability to provide developers and architecture teams with the flexibility to choose the best technology for their need or use case. Delivering this requires an open-platform built on the principles of enabling developer experience. It should therefore offer extensibility through a platform API, support for webhooks as well as CI/CD integrations, and again be constructed to support multi-gateway API ecosystems.

Building a Rollout Plan

Simply making APIs available does not ensure their adoption. A productized approach to APIs requires a well-designed rollout plan. This plan should combine tooling for everything developers need with a deliberate communication plan which articulates the utility and value of all the APIs. Importantly this plan should also provide guidance and instruction on how to most effectively engage with those APIs. There are a couple of key steps in designing and delivering this.

1 Align on an API Hub

Until the organization has a single source of truth for all of its APIs, the developer experience will continue to be fragmented. For guidance on evaluating an API hub's ability to deliver this see the above Choosing An API Hub section.

2 Select an Initial API Set

The next step is to decide on which APIs will be included in the initial offering, those which will be added in subsequent releases and prospective timelines of their delivery. Starting with a subset of the APIs that limits the number of variables in assessing the program's performance. This easier to see trends in levels of adoption and usage, and isolate reasons for that disparity.

3 Outline User Journeys

As with any product, it is important to outline the complete user journey. This applies to both API builders and API consumers. again Having a clear view of both of these audiences will enable a more effective communication plan that speaks to what is important to them and offer clear guidance on how they can interact with APIs in the API hub.

Establishing A Communication Plan

A well-constructed communication plan should mirror the steps and tenants outlined in the rollout plan. This communication plan should clearly outline which APIs are available (both those initially offered as well as providing clear guidance on roadmapped APIs), what their value is to the user, and how to interact with them.

Just like the rollout plan, it's essential that this communication plan takes into account both the API consumers and API builders. To enable API builders and ensure that the APIs they develop have the best chance of adoption this plan should outline best practices and methodology for how to communicate their functionality and benefits. Creating this structure ensures that all participants, API builders and consumers operate with a shared nomenclature that is clear and informative.

The communication plan should:

- Provide guidance on how to communicate an API's value.
- Offer instructions on how users can navigate and interact with the API hub.
- Outline processes for API discovery, requesting access, and contacting support.
- Specify versioning policies.

Evaluating The API Program With Metrics

To understand how the API program is performing it is essential to set quantifiable metrics as part of its initial design and launch. With those measurements the next step is to set goals identifying what success looks like across them at specific milestones and/or standard intervals such as six months, a year, two years, and beyond. There is a wide variety of metrics used to assess internal API program's performance - and not every program needs to leverage every single one of them. Choose metrics which align with the organization's specific outcomes or goals.

Some of the more commonly used metrics include:

Number of APIs – Growth in this figure is indicative of adoption and usage, particularly amongst API builders.

Avg. Daily User Count – Tracking the daily number of users (on avg.) is the clearest indicator of overall engagement.

Issue Resolution Times – Assessing support performance - this figure should be measured against the windows outlined in the communication plan.

Total Number of APIs Consumed – Offers insight into overall program adoption.

API Call Volumes – Indicating the level of adoption and usage, of the APIs themselves or the applications which they comprise.

API Volume Distribution – Analyzing the extent of a specific API's adoption.

Time to 200 – This is the amount of time it takes a developer on average to find, request access to, and make their first API call.

Average time to market for new digital services – Reduction in this figure represents increased API reuse and speedier development times.

Conclusion

Ensuring successful digital transformation relies on organizations productizing the APIs contained in their internal program. Adopting this productized approach helps ensure developer adoption, reuse and collaboration, and drives objectives such as accelerated innovation, faster time to market, and reduced development costs. Leveraging the best practices covered in this guide will help ensure effective API productization and successful digital transformation.

About RapidAPI Enterprise Hub

RapidAPI Enterprise Hub is the next-generation API hub that enables leading enterprises to accelerate innovation and bring software to market faster with one place to discover, govern, and collaborate on all APIs. RapidAPI offers a single platform for all APIs – every API protocol, every API type, and across every API gateway. RapidAPI can programmatically communicate with all your API gateways, enabling you to centralize all of your organization's APIs.

Rapid

Global HQ
85 2nd Street, Fourth Floor-
San Francisco, CA 94105

Contact
info@rapidapi.com
www.rapidapi.com

10/22-Updated 02/23