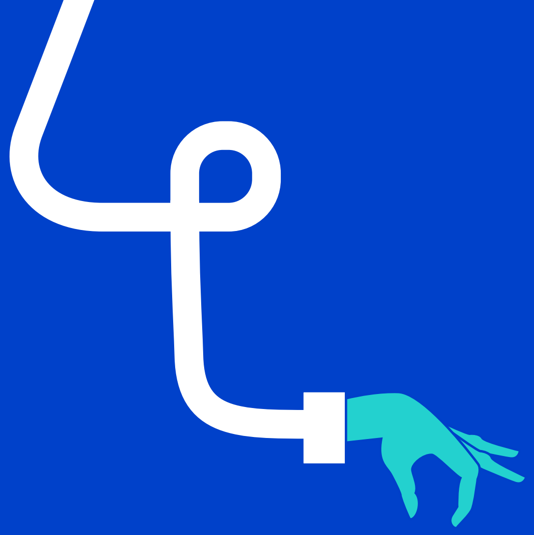
THE API Buyer's Guide

Everything Technical Leaders Need to Know About Purchasing APIs



Nylas

OVERVIEW

It's no secret that businesses need to move fast to stay competitive. With the pace of innovation showing no signs of slowing, technical leaders are tasked with increasing velocity. This imperative has forced a shift not just in how we work — agile methodologies, DevOps cultures, and CI/CD practices — but what technical solutions are used by engineering teams to quickly build and ship new products and features.

To complicate matters, products and services are now hyper-connected. According to Forbes,

> "How a business wins or loses is increasingly dependent on how well they connect to external party apps, devices, and services."

This forces in-house technical teams to get creative and either become experts at technologies that are outside of their core competencies or adopt API solutions to help quickly build reliable and secure integrations.

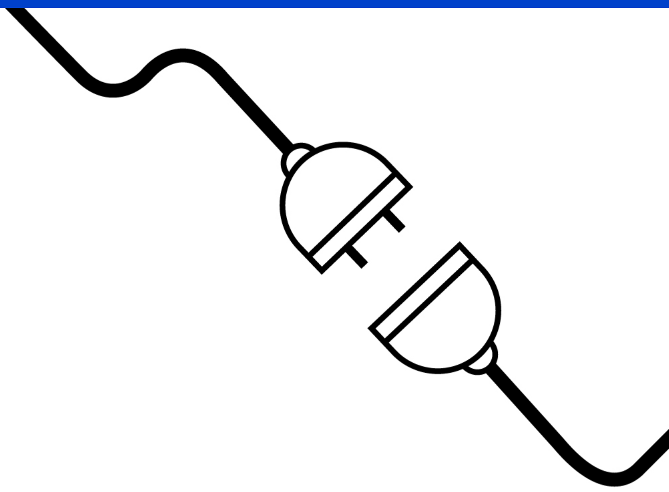
This is the classic "Build vs. Buy" conundrum that technical leaders face while trying to innovate and go to market faster. It's no surprise that adopting APIs is becoming the de facto solution, as it solves a host of these challenges — namely, the ability to build integrations without requiring as much time and resources from engineering teams.

The demand for speed *x* reliability *x* security has created a surge in API solutions. In this guide, we'll outline the key questions technical leaders should ask when considering an API solution and provide best practices for evaluating options, including:

1. *The Benefits of API Solutions*
2. *The Cost of Building an Integration In-House*
3. *Technical Criteria for Evaluating APIs*
4. *How to Structure a POC (Proof of Concept)*
5. *How to Successfully Implement an API*

ABOUT NYLAS

Nylas is a pioneer and leading provider of unified communications APIs that allow developers to connect their applications to any inbox, calendar, or contacts book in the world. Over 22,000 developers around the globe rely on the Nylas communications platform to handle over 100 million API requests per day to Gmail, Microsoft Exchange, Outlook.com, and more. Nylas' flagship email API has synced more than 15 billion emails to date and is used by global enterprises such as Hyundai, Fox News Corp, and Realtor.com.



The Benefits of API Solutions

The #1 reason technical leaders purchase APIs is to help their teams launch new features faster. APIs help engineering teams tackle specific, domain-centric challenges that otherwise require specialized expertise and skill-sets.

However, beyond the ability to build simply integrations quickly, APIs also help with:

SECURITY

APIs can offer specialized, enhanced security. Reliable providers can tout security in the form of SOC 2 certification, GDPR compliance, HIPAA and FINRA readiness, and Privacy Shield Certification. Adopting these solutions has the carry-over effect of helping you maintain compliance as security requirements continue to evolve and change, which should help your engineering teams sleep easier at night.

RELIABILITY

Accelerating growth and expanding to meet the needs of a booming base of customers is a great challenge to face, but as products scale, so do bugs and customer support issues. This is a reason technical leaders tend to adopt API solutions as they can reduce (or eliminate) the need for ongoing maintenance, bugs, and support.

> We already live in an API economy where CIOs must look beyond APIs as technology and instead build their company’s business models, digital strategies, and ecosystems on them.

INNOVATION

Adopting an API solution gives teams valuable time back to solve other challenges and focus on building robust, differentiating features. The core need to increase the speed of innovation is why Gartner VP of Research Paolo Malinverno (10 Steps to the API Economy) stated, “We already live in an API economy where CIOs must look beyond APIs as technology and instead build their company’s business models, digital strategies, and ecosystems on them.”

USER ENGAGEMENT

Integrated products tend to be “sticky” products, as users favor applications that integrate with the services they rely on, such as SMS, email, or GPS. Delivering tightly integrated features removes the need for your users to abandon your “home base” and switch between applications.

REVENUE

Using in-house developers to build direct integrations can be a costly drain on resources. However, the real problem is when feature and product launches are delayed due to unforeseen complexities inherent in building direct integrations. This is where revenue can take a hit; the maxim of “features = customer acquisition = revenue” almost always holds true. This is why technical leaders also tend to look to APIs as an investment in both customer engagement and retention.

The Cost of Building an Integration In-House

Building an integration in-house requires technical teams to become experts in specialized areas — such as processing payments or sending and receiving SMS communications. Depending on the type of integration that is being built and the data that is processed, engineering teams may also need to become experts in regulatory practices for processing data in the banking, healthcare, or legal industries.

This means the cost of ownership of building and maintaining an integration in-house can be quite steep. Here’s an estimated breakdown of a few key areas:

HEADCOUNT

According to Glassdoor, the average level 1 software engineer’s salary in the US is approximately $103,000 per year. In certain areas (such as the San Francisco Bay Area) salaries can quickly rise by 50% or greater vs. the national average.

Ultimately, headcount costs vary depending on the complexity of the integration. When assessing the integration, take into account everything from scoping, building, testing, and any ongoing maintenance and support that is required. In a conservative scenario where an integration takes just two engineers six weeks to build start-to-finish and another 4 weeks to maintain in the first 12 months, the headcount costs eclipse $35,000.

SERVERS

Server costs can vary wildly depending on the number of users you have as well as the type of data you're processing. Larger data files that update frequently (like email data, photos, and documents) are the most costly.

Other factors to consider are the service level you request out of your cloud storage system, as well as the amount of data you're storing, and the amount of time you retain that data. AWS charges around $.10 per gigabyte per month, so if you need to retain and store 50 terabytes of data, that would cost you $60,000 per year, not inclusive of data processing costs (i.e., the cost of servers) or additional account services.

SECURITY

If you are processing any user data, you must consider security. Calculate the cost of the staff you'll need to oversee the security audits and implementation of security processes and features. Auditing firms range wildly depending on the scope of the audit and the prestige of the firm — this can run anywhere from a few thousand dollars to tens of thousands per month for the duration of the engagement.

Attaining industry-standard security certifications like SOC 2 runs an additional $20,000-$30,000 for the most basic certification. This is another reason why technical leaders look to API vendors as they can help decrease security costs and drastically reduce the amount of implementation or oversight needed for the integration.

**Note that these costs are estimates based on industry averages.*

SUPPORT

The work isn't done once you've built the integration — there are additional costs required to support the integration over time. This includes debugging customer issues, managing version updates, and diving into edge case issues that arise.

In some cases, a team of technical support engineers may be required to debug your customer's issues. To put this in context, Glassdoor reports that a single Technical Support Engineer's salary runs $63,000 on average.

THE BENEFITS AND DRAWBACKS OF "BYO"

The "BYO" approach (Building Your Own) can have its benefits, but the path is peppered with landmines. Here's a quick side-by-side comparison of taking the BYO route.

BENEFITS OF BYO

- ✓ Full control over your infrastructure
- ✓ Decreased vendor reliance
- ✓ Build expertise and experience in a niche field/topic
- ✓ No contracts or subscription fees

DRAWBACKS OF BYO

- ✗ Difficult and costly to maintain
- ✗ Increase in customer support overhead
- ✗ Separate security infrastructure needed for integration
- ✗ Slower time to market for integrations

Purchasing an API has many benefits including lowering workload, improved security, and a faster go to market time, but it also comes with some drawbacks.

BENEFITS OF AN API

- ✓ Focus on top business priorities: innovation, revenue, launching products faster, reaching new audiences
- ✓ Free up time for developers and engineers to build new features
- ✓ Low maintenance
- ✓ Security is handled for you
- ✓ Go to market faster
- ✓ No breaking changes

DRAWBACKS OF AN API

- ✗ Loss of complete control over end-to-end integration
- ✗ Reliance on third-party vendor
- ✗ Contracts and subscription fees

COMPLEXITY: THE CASE AGAINST BUILDING DIRECT INTEGRATIONS

Complexity is one of the biggest reasons direct integrations fail. For example, if your product is a CRM and you want to integrate your user's email accounts so they can send and receive email from within your interface, you'll need to build individual, direct integrations for each email service provider (Gmail, Exchange, Outlook).

Lexicata, a large legal software company based in Los Angeles, reduced costs by adopting the Nylas email API. "It would have cost us at least a year to build out just the Exchange and Google integration," says Aaron George, Director of Product Management. By building on top of an API, the Lexicata team was able to focus on other core components of their CRM.

> "It would have cost us at least a year to build out just the Exchange and Google integration."

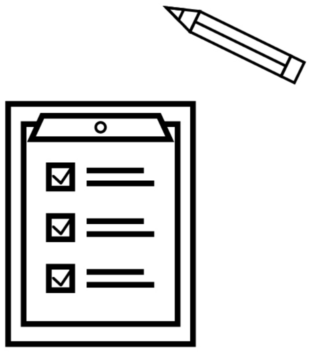
Lexicata

The same can be said for financial transactions —building infrastructure in your app to support connectivity to one bank takes long enough, let alone adding new banks or different payment options (like PayPal). Stripe, a payments API founded in 2010, provides a fully integrated payments platform that helps solve this challenge. Stripe's customers embed the API into their platform, allowing them to accept payments from various payment providers, set up recurring monthly bills, and increase revenue through a streamlined checkout process, all without dealing with the complexity of building and managing direct integrations.

PersistIQ, a provider of sales engagement tools, used the Nylas APIs to remove the complexity of managing direct integrations. "At the time, the decision was 'We can figure this out, or you guys can.' It just seemed cleaner and easier to integrate with one provider," said Cyrus Karbassiyoon, co-founder and CTO at PersistIQ. "It was too hard to figure out how to build the integration ourselves while also figuring out what users want from the rest of our product."

The more complex the integration is, and the more products you need to integrate with, the more daunting the prospect of building direct integrations becomes. This is where APIs can be a big boon for go-to-market speed and team efficiency.

Technical Criteria for Evaluating APIs



Since APIs power critical features within your application, you'll want to ensure they measure up in key areas. Here are the criteria you should consider when assessing API solutions:

PERFORMANCE

A good API should perform well, whether it is your first request or the millionth request. You should first measure an API's performance based on response time — i.e., how quickly does the API return data after your platform requests it? This is critical as users are unforgiving when it comes to poor performance — in the world of e-commerce, for example, shoppers usually abandon shopping carts when they have to wait longer than 3 seconds for web pages to load.

A good API should perform well, whether it is your first request or the millionth request.

Note that there are factors that could add latency include the server, ISP, or operating system. On mobile devices, response times are impacted by the carrier's network or WiFi speeds. For web apps, the #1 cause of latency is complexity, or the hodgepodge of third-party services, cloud-based computing, and self-hosted infrastructure. This is why it is important to have a robust testing plan in place for any API to ensure it delivers the performance your users need.

RELIABILITY

APIs should have a guaranteed 99.9% uptime or greater. When APIs experience downtime, the parts of your service that the API supports may be temporarily unavailable, which is why availability metrics are so important. Even one hour of downtime can cost tens or hundreds of thousands of dollars of lost revenue. Demand that any API vendor have an SLA (Service Level Agreement) in place to guarantee reliability.

Demand that any API vendor have an SLA (Service Level Agreement) in place to guarantee reliability.

SECURITY

Since APIs transfer a lot of rich data, they're often secured based upon the highest standards. Enterprise-grade compliance certifications to look for include SOC 2 Certification, GDPR Compliance, EU Privacy Shield Certification, and HIPAA and FINRA readiness. As an extra layer of security, APIs should undergo third-party audits and rigorous penetration tests. Ask for proof from a vendor that third parties regularly test their technology for security.

COST

APIs are priced in different ways — typically either the number of API calls, size of data transferred, the number of packages sent or received, or the number of synced users. Some even use a combination of these variables when calculating your monthly cost. Regardless of the model, billing and pricing should be transparent and predictable. Ask for packages where there is a monthly or annual commitment that locks in rates, and push for a "flat rate" option if you have a high number of active users that are using this feature.

EASE OF INTEGRATION

An API should be easy to integrate with. Look for user-friendly, well-organized documentation, onboarding guides and support, and SDK (software development kit) availability. Also, ask the vendor for customer references that attest to the impact of the integration in terms of time savings and security enhancements.

10 POPULAR BUSINESS APIS

Here are 10 of the top rated APIs for specific business functions:

twilio	stripe	PLAID	shippo
Send voice, SMS, and videos from your platform.	Process and operate online payments - technical, fraud prevention, and banking infrastructure.	Connect consumer's bank accounts to your application for personal finance software.	Embed any shipping provider's data into your platform so users can receive, print, and track shipments all from your application.

Braintree

In-store and mobile payment processing platform, accepting more than 130 currencies.

yelp fusion

Embed reviews and ratings for more than 50 million businesses based on geographic regions or specific locations.

xero

An online accounting system that gives users access to bank transactions, invoices, and reports.

onelogin

Single-click access to applications by eliminating the need to remember strong passwords.

okta

Deploy user authentication to your application in a matter of minutes.

Nylas

Connect your application to any inbox, calendar, or contacts book in the world.

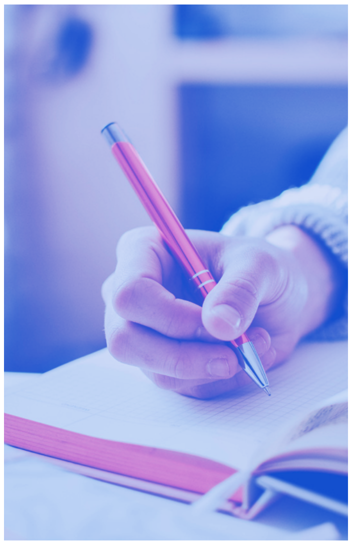
How to Structure a POC (Proof of Concept)

Once you've assessed whether an API solution is fit-for-purpose, you need to go through a test phase to validate the vendor's claims. Here are the most important steps for building a robust POC (Proof of Concept) where you can thoroughly test an API before moving onto a full implementation:

IDENTIFY A USE CASE

The first step in testing an API is to develop a customer use case — for instance, a feature your customers have been asking for; a disruptive feature your competitors have; or a promising new technology that helps protect your product's unique value.

Likewise, you can also consider internal use cases. Is there a significant tool employees use that needs a specific feature to help improve productivity? For example, finding an API that could collect and unify all invoice attachments across every department into one unified source could save your accounting team hundreds of hours of work per year.



DEFINE METRICS FOR SUCCESS

Allocate the timeframe in which you'd like to test the API. Be sure to give the engineering team ample notice, so they can complete their current projects before starting on the POC. Clearly define how many resources you'll need to allocate to test the product, and set a hard deadline for the test to be complete.

INTERVIEW INTERNAL STAKEHOLDERS

A great API should have excellent, clear documentation that's easy to understand. However, this alone is not enough to ensure a successful POC. As your engineering team starts building out the implementation, ask them about their expectations. Document and share their feedback with the API vendor, so they understand your team's needs. Check in frequently to ensure the POC is progressing according to their expectations.

> A great API should have excellent, clear documentation that's easy to understand.

SECURE A SUPPORT CONTACT

As you start your trial of the product, you may have questions for the support team. How responsive is the API provider's customer support team? Do they offer support SLAs with response times? Establish regular check-ins to discuss any difficulties the team has with the vendor so they can quickly provide support.

GET ARMED WITH SDKS (SOFTWARE DEVELOPMENT KITS)

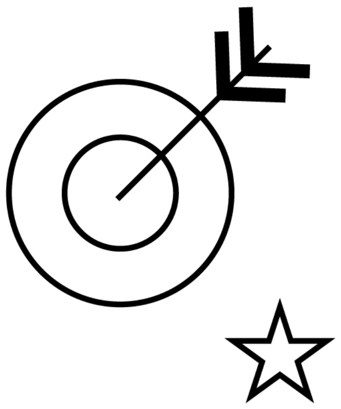
Every credible API provider should offer SDKs and polished documentation that accelerates the integration. SDKs can make integrating as quick as a week or two (depending on the complexity of your integration) — saving a few months over building your integration without an API or SDKs. Be sure that the vendor provides you with their SDKs before officially launching the POC.

LAUNCH THE TEST

You'll always want to test drive an API before buying it. Most APIs offer full-featured trials that allow your developers to test the API before signing a contract. Test the API for the primary use case you have in mind, as well as for any adjacent use cases.

For example, if most of your customers pay via ACH, you want to ensure the ACH integration works smoothly across providers. If only a small portion of users pay via credit card, you'll still want to stress test at least 1-2 different credit card payments via the API.

How to Successfully Implement an API



Once you've tested the API and you're happy with the results, follow these steps to fully implement the API:

DEFINE THE PROTOTYPE

With the POC completed, you're now ready to create a proper prototype. While the POC shows us whether or not a product or feature can be developed, a prototype shows how it will be developed.

Gather the key team leads involved in the project and decide what the POC should look like. Bring in technical team members who are building the integration to understand how long the POC takes to build. Depending on the complexity of the project, this could be anywhere from a few days to a few weeks.

ASSIGN ROLES

Larger, more complex integrations require more team members than smaller ones. At a minimum, you'll want one engineer and an Engineering or Product Manager to manage timelines and to select the significant milestones. You'll also want to bring in a UX/design resource to design the interface for the features that the API powers (like your billing/invoicing portal). If the project specs are documented, the UX design can take place in parallel with the implementation of the API.

If the project specs are documented, the UX design can take place in parallel with the implementation of the API.

BUILD & TEST

An API integration can take anywhere from a few days to a few weeks, depending on the scope and complexity of the project. Once the prototype is ready, share it amongst a select pool of customers with the expectation that this is a beta test. Collect qualitative and quantitative data on how the features powered by the API are used. Is the interface intuitive? Are there any bugs or snags? Relay this feedback to the engineering team. Once the prototype is smoothed out, you're ready to complete the MVP (minimum viable product).

RELEASE THE MVP

An MVP is a functional solution that's ready for your full set of customers to use. The MVP gives you the ability to collect even more data from your broader customer pool, and — perhaps most importantly —to start earning revenue from these new features.

ANALYZE

Track the adoption of the new integration against the success metrics you set out at the onset of the project. Has the API increased user engagement? Has it increased revenue/sales? How can you continue to add value to your users around this feature? This feedback should work as a constant loop that helps inform iterations including additional features that can be built on top of the integration.

CITATIONS

1. Gartner
2. Gartner 2019 SaaS spend
3. Blissfully: "2019 Annual SaaS Trends Report"

Nylas

Nylas.com

Github.com/Nylas

@Nylas