7 mistakes

you need to avoid
when building long-tail
product integrations

MERGE

Contents

SECTION 1

Introduction 1

Overview on long-tail integrations 1

SECTION 2

Mistakes when building long-tail integrations: 3

Assuming each long-tail integration build is the same 4

Failing to create internal documentation 6

Treating onboarding as an afterthought 8

Ignoring crucial integration tests 10

Operating without monitoring and alerting processes 12

Limiting the number of engineers on an integration project 14

Deciding to only evaluate embedded iPaaS solutions 16

SECTION 3

Introducing Merge 18

Build long-tail integrations without issues through Merge's Unified APIs 19

SECTION 1

Overview on long-tail integrations

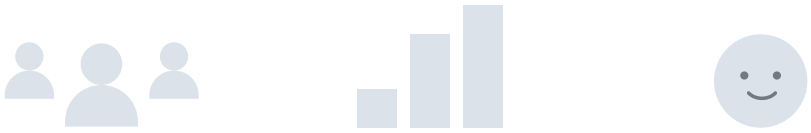
Overview on long-tail integrations

A long-tail integration, or an integration that's used by a minority of your clients and prospects, is often undervalued and, as a result, under-resourced.

Organizations see them as a necessary evil for winning a deal or keeping a few clients happy and don't fully appreciate the potential impact they can have on the business.

This evaluation is often misguided. Long-tail integrations can be used by your biggest clients; they can attract the most lucrative sales opportunities; they can prove crucial in helping you expand to target markets; and **they can eventually become core integrations for your business.**

Core benefits of long-tail integrations



Customer acquisition | **Market expansion** | **Retention**

Long-tail integrations can ultimately help you close more deals, retain more clients, and expand to more markets. But they only offer these benefits when they're built effectively.

Reaping the full benefits of long-tail integrations involves building and maintaining them thoughtfully and strategically. And this involves avoiding several significant mistakes.

This guide covers many of these mistakes in detail so that you can avoid them.

SECTION 2

Mistakes when building long-tail integrations

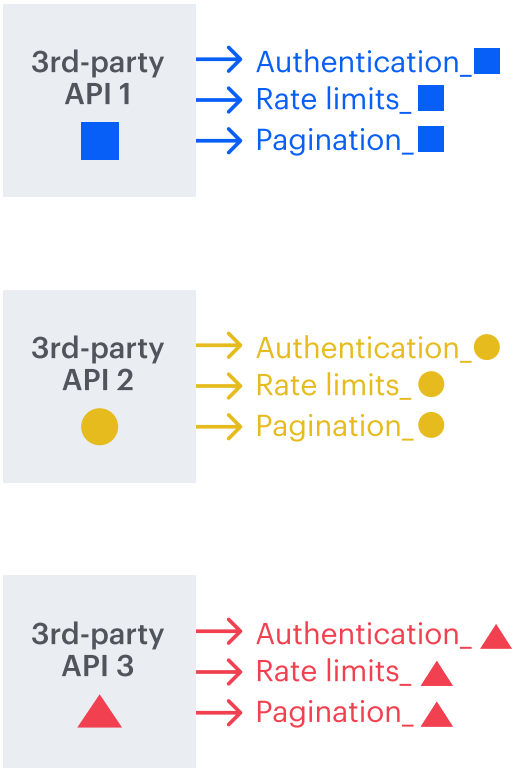
Mistake 1:

Assuming each long-tail integration build is the same

Most integration builders wrongly assume that APIs are similar across vendors.

In reality, every long-tail integration's API can differ in significant ways, whether it's their approach to authentication, pagination, rate limiting, etc.

Failing to consider these differences across long-tail APIs can lead you to underestimate the resources needed to build your next integration. In turn, your timeline for building the integration can extend well beyond your intended target date, causing your clients and go-to-market teams to become frustrated and disappointed.



Why organizations make this incorrect assumption:

The engineers involved have limited experience with building integrations.

The engineers may have quickly reviewed the 3rd-party vendors' API docs and found some overlap—leading them to assume that the builds would be largely similar.

There's little publicity around the differences in APIs across providers and how difficult this makes it for product and engineering to integrate at scale.

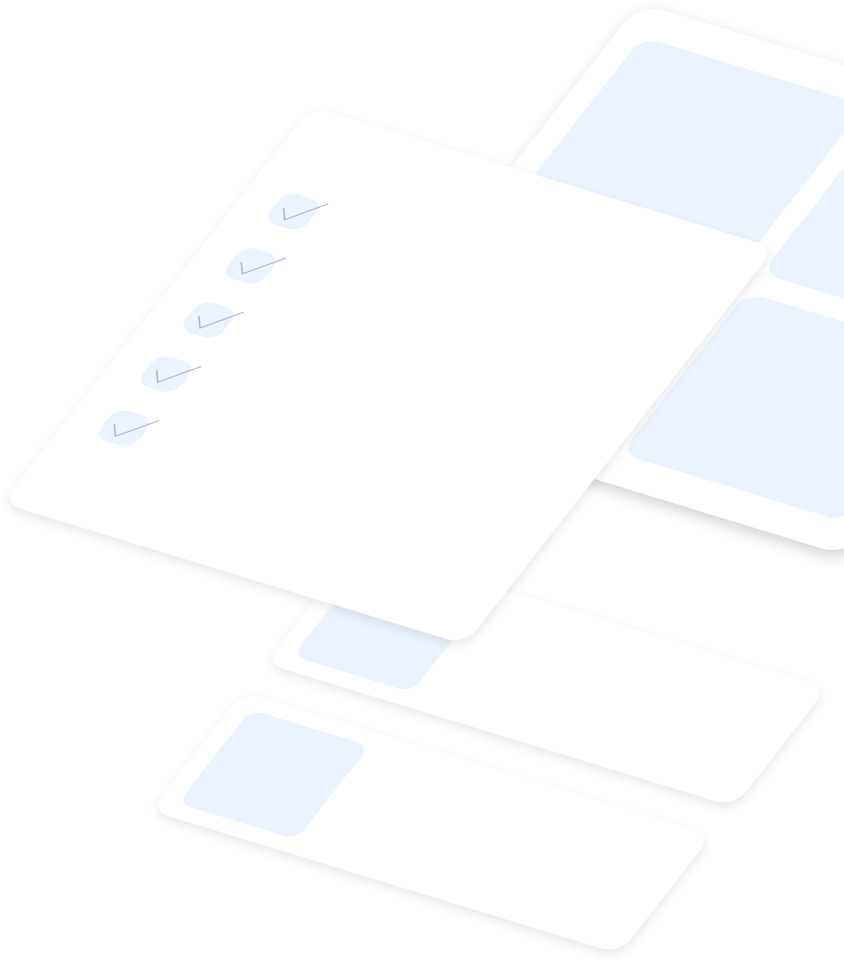
Mistake 2:

Failing to create internal documentation

Internal documentation on long-tail integrations is critical for maintaining and enhancing each integration.

It allows engineers to get high-level information, such as the purpose of the integration. And it can provide more specific insights around how the integration handles errors, manages rate limits, etc.

Failing to create and maintain this internal documentation can leave you vulnerable when the engineer who's assigned to the integration leaves. And even if they stay, they'll likely forget key details over time. Therefore, internal documentation also prevents you from relying on someone's memory.



Why organizations fail to build out and maintain internal documentation on their long-tail integrations:

The engineers assigned to the long-tail integration see little reason to create documentation since they themselves are doing all the work.

The time involved in building out and maintaining the documentation for these integrations is seen as “not worth it.”

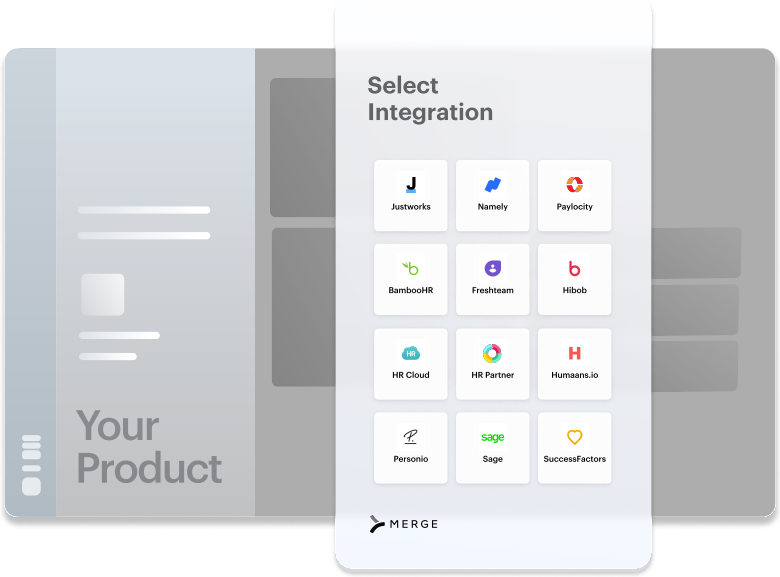
Organizations don’t see the value of integration documentation, as they’re focused on a short time horizon; this mindset trickles down to long-tail integrations.

Mistake 3:

Treating onboarding as an afterthought

For a given long-tail integration, you may be willing to offer high-level authentication information, such as the authentication method it uses. But you're less likely to research and provide the actual steps for completing the authentication process.

This puts the burden squarely on clients. They'll have to rely on the long-tail integration's API documentation for guidance, which introduces friction that can ultimately dissuade them from integrating with your product.



Merge Link—a UI component that our clients can embed directly into their applications—provides our clients' customers (or end-users) with clear, step-by-step instructions to authenticate their applications.

Why organizations don’t provide onboarding support for long-tail integrations:

Researching the authentication process for each long-tail integration and explaining how clients can go through the process in a way that’s concise and clear is both time and resource-intensive.

Since only a few clients may be using a long-tail integration, organizations may see little upside to providing it.

The first few clients may have gone through the process without any complaints, giving the organization a false sense of confidence that they don’t need it.

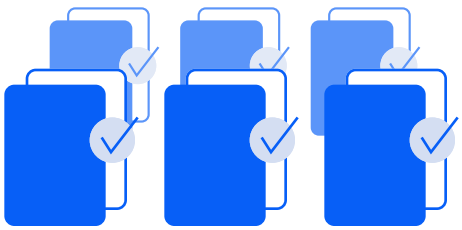
Mistake 4:

Ignoring crucial integration tests

Scale and load testing are fundamental components of integration testing.

Performing BOTH of these tests is crucial in validating that an integration can, actually, work at scale without any degradation in performance.

Scale testing
allows you to see how the integration performs when dealing with significant volumes of data.



Load testing
can help you determine how effective the integration is in processing a high number of simultaneous requests.

Why organizations often avoid scale and load testing long-tail integrations:

Since the long-tail integration may only be used by just a few small clients today or in the short-term, the organization assumes that it'll never deal with large data sets.

Running these tests require specific technical expertise (e.g. being proficient in tools like LoadRunner). This likely leaves just a few engineers who can perform these tests, and organizations are often unwilling to invest their limited resources on this.

These tests can be resource intensive compared to others, so organizations often skip over them with the thought that performing less demanding tests is "good enough".

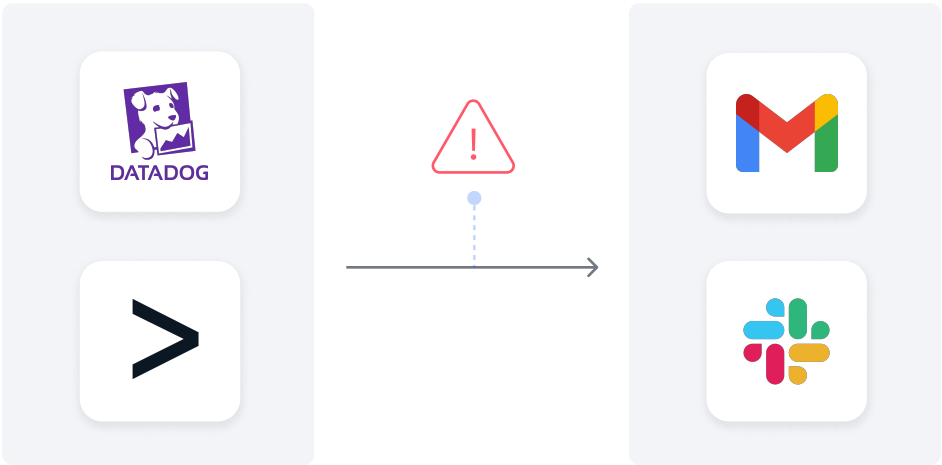
The accounts associated with performing scale tests have to have high volumes of data, making them expensive to maintain over time.

Mistake 5:

Operating without monitoring and alerting processes

Every integration will inevitably break. API response bodies are impossible to predict, and even minor discrepancies between what you expected and how an API response comes back can be enough to break the integration (e.g. you expected the response to be an integer but it returns as a string).

To minimize the impact an integration issue has on a client(s), you can adopt tools that can monitor your integrations, like Datadog or Splunk, and build alerts that notify your team in specific channels (e.g. Slack) when predefined issues occur.



Unfortunately, many organizations fail to implement even basic monitoring and alerting processes for long-tail integrations. As a result, integration failures will go unaddressed or persist for lengthy periods of time—leading clients to get upset and eventually leave.

Why organizations don’t establish monitoring and alerting processes for long-tail integrations:

Organizations wrongly assume that since the integration is used by just a few clients, there will be few issues, if any, to deal with.

Adding a long-tail integration to your existing monitoring and alerting systems can take time and may not be so straightforward. As a result, adding it to these systems can get deprioritized.

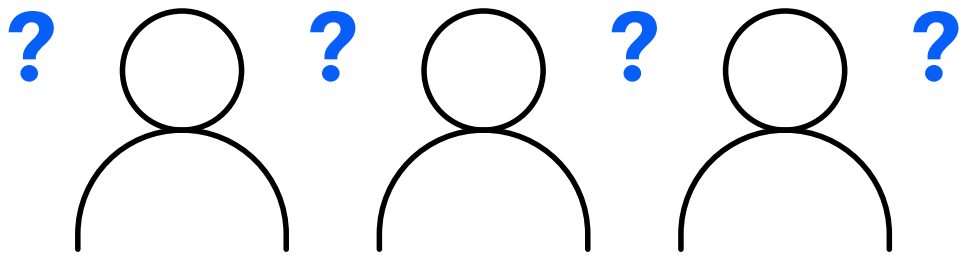
Core integration issues are prioritized over long-tail integrations due to the difference in the number of clients affected.

Mistake 6:

Limiting the number of engineers on an integration project

You might just have 1 or 2 engineers dedicated to building and maintaining a long-tail integration (all the while tasking an entire team of engineers with supporting a core integration).

While this situation might make sense, the reality is that these 1-2 engineers can leave your company at any point. Once they do, they'll take precious knowledge of the long-tail integration they built and maintained with them—leaving your remaining engineers in a bad position.



Why organizations assign few engineering resources toward building and maintaining long-tail integrations:

There's a failure to recognize that long-tail integrations are just as time consuming to build and maintain as core integrations.

Organizations are reluctant to move engineers away from core integrations and/or other critical projects to integrations that provide relatively less value to the business today.

Organizations don't believe that the potential returns of building and maintaining a long-tail integration effectively offsets the costs of allocating more engineering resources toward it.

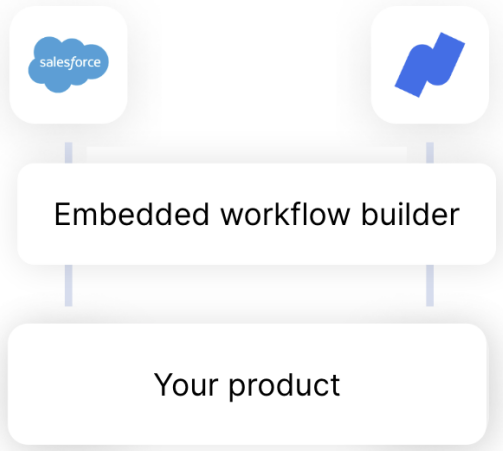
Mistake 7:

Deciding to only evaluate embedded iPaaS solutions

Embedded iPaaS tools, like Workato and Tray.io, let you build one integration at a time with your product.

While they may seem like a promising approach to building integrations cost-effectively, they're anything but.

Your engineers are forced to learn and become comfortable with a new UI, which can take them weeks. They're also forced to build integrations incrementally, which prevents you from building tens, and eventually hundreds, of integrations quickly and easily. Finally, the platform lacks the maintenance support and monitoring tooling you need to provide clients with reliable and high-performing integrations.



Why organizations only evaluate embedded iPaaS tools:

They may have liked the vendor's iPaaS tool and assumed the embedded version would work just as well for their customer-facing integrations.

Embedded iPaaS tools falsely advertise offering a low-code/no-code UX, which appeals to organizations that want to move their engineers away from integration projects.

Embedded iPaaS vendors often have extensive go-to-market operations, so they may be the first and only 3rd-party product integration solution an organization comes across.

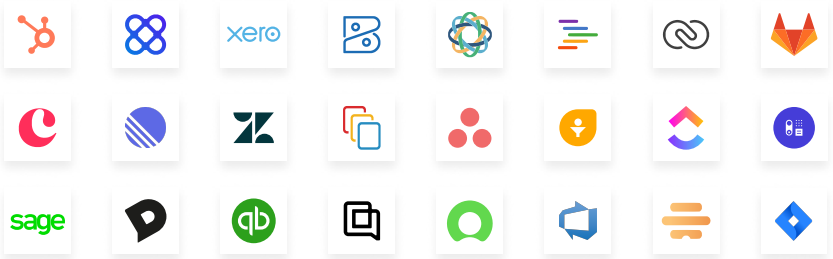
SECTION 3

Introducing Merge

Build long-tail integrations without issues through Merge's Unified API

Merge, which is a single API to add hundreds of integrations to your product, lets you avoid making the mistakes outlined above—among countless others.

Simply build to Merge's Unified API to offer all the long-tail integrations your clients and prospects want in a given software category.



A snapshot of just some of the CRM, accounting, and ticketing integrations you'll be able to access by building to Merge's Unified API.

Merge also offers maintenance support and management tooling to help keep your integrations healthy and performing at a high level; comprehensive Common Models to ensure your clients can access and sync the data they care about; and broad category coverage, which includes HRIS, CRM, file storage, ticketing, ATS, accounting, and marketing automation, to account for all the integrations your product needs.

You can learn more about Merge by scheduling a demo with one of our integration experts.