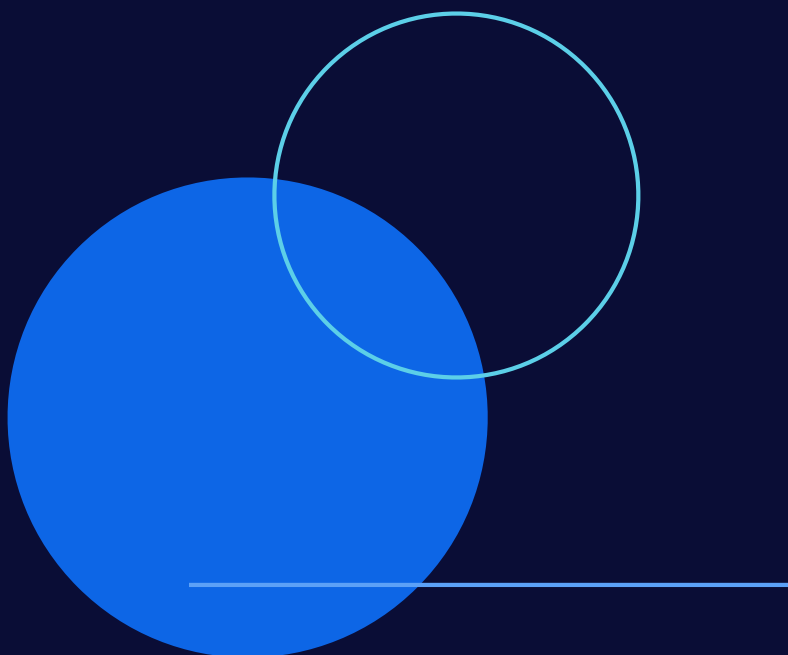




THE DEFINITIVE GUIDE TO

# Embedding Integrations in Your Product



## TABLE OF CONTENTS

<b>Why it's time to add integration to your product</b>	<b>3</b>
<b>10 benefits of using an embedded integration platform</b>	<b>4</b>
<b>10 key criteria for customer integrations</b>	<b>6</b>
1. Assess integration development velocity, efficiency, and flexibility	6
2. Evaluate not only scalability—but lights-out elasticity	9
3. Sweat the details on the end user activation (or face the consequences)	10
4. Ensure your roadmap won't hit brutal customer connectivity roadblocks	12
5. Check if embedded integration is a first-class object for your provider	14
6. Make sure there are modern and complete APIs for your engineers	15
7. Drill into maintenance, monitoring, and management	17
8. Ensure there are options for both productized and services-led integrations	19
9. Get into the weeds on certifications and security	22
10. Avoid contract nightmares. Negotiate predictability and simplicity	25
<b>Your embedded integration checklist</b>	<b>28</b>

## Why it's time to add integration to your product

You've come to the right place. Modern software customers now expect the applications they buy to work together—fast, efficiently, and ideally out of the box. For providers, adding an embedded integration platform is the most practical way for their product and services teams to deliver on growing market demands.

Customers want the software they buy to include packaged integrations that enable them to connect to their stack in minutes, not months, ideally in just a few clicks. And they'll choose vendors that provide speed and ease of use over those that don't—because they can achieve results faster.

**“By 2023, use of packaged integration processes will grow from less than 30% in 2020 to above 65% of new integration projects”**

— Accelerate Your Integration Delivery by Using Packaged Integration Processes, Gartner, 17 November 2020

Application stacks in your customers' marketing, sales, service, finance, or operations departments are more fragmented than ever. The average department runs more than 40 apps—and they're growing in number by more than 10% per year. So software and services providers experience a massive headache with several critical challenges: First, how to meet the growing out-of-the-box packaged integration demands of prospects and customers; second, how to avoid crushing engineering and services with connectivity and endpoint maintenance; and third, how to avoid compromising the user experience.

As a result, software and services organizations are looking to embedded integration platforms to deliver integrations efficiently and fast. But while there is a vast range of integration technologies, some new, others that have been on the market for decades, the majority are designed for IT or end business users, rather than for software companies to embed into their own products. As a result, most solutions don't meet the unique needs of product teams to embed integrations into their commercial products or services teams to speed delivery.

This guide is based on proven implementations by product leaders who have embedded integrations into their products to successfully accelerate their customer integration roadmap, including Eventbrite, Typeform, HackerOne, and hundreds more. It provides a complete drill down into all the criteria so you can deliver the best integration experience for your customers, product, and services team.

## What is an embedded integration platform anyway?

Embedded integration platforms empower software companies or services organizations to deliver integrations to their customers (end customers) quickly, efficiently, and seamlessly. They include low-code tools, connectors, and APIs to enable software company developers to rapidly deliver reusable in-product integrations, and for services/implementation teams to deliver ad hoc, bespoke integrations if needed. They're designed to be rebranded with embeddable components that developers can use to add integration to their products that guide end-users through configuring and self-activating integrations in-product or through a marketplace. Finally, they provide governance, compliance, elasticity, and control so that providers can deliver integrations to their customers securely and at scale.

## 10 benefits of using an embedded integration platform

Since you're reading this guide, you've probably already settled on using an embedded integration platform to deliver your customer integrations. But what you may not realize is using a platform designed to streamline and accelerate customer integration delivery can provide even broader benefits.

- 1. Free up your engineering team from time-consuming integrations.** Building integrations for different apps from the ground up, connecting to ever-changing REST endpoints, and managing authentications can seriously drain engineering resources. Using an embedded integration platform instead can typically boost developer productivity by more than 60%.
- 2. Accelerate your integration velocity.** Building integrations in-house isn't just resource-intensive; it's slow going and code-intensive. Embedded integration platforms, especially those that incorporate low-code, can provide a velocity boost of 4X or more.
- 3. Focus your engineering team on your core product.** Engineering resources are more valuable than ever, which is why you don't want them tied up customizing integrations for specific customers. Even worse, once you commit your engineering team to building an integration internally, you've also committed them to go back and maintain it forever as different apps in the tech stack update their APIs over time. Embedded integration platforms cut out the painstaking integration builds and endless maintenance so the team can focus on core platform features and differentiators.

4. **Hiring avoidance.** With less human power required to build integrations, whether in the product or services team, you can avoid full-time employees (FTEs)—or simply hire for other activities that drive more value, not to mention redeploying existing teams previously focused on integrations toward more high-value projects. Your choice.
5. **Unlock more monetization opportunities.** Integrations can often be added to the price list and drive recurring revenue. If you can deliver more integrations faster, that's an excellent opportunity to grow customer average sale price (ASP) or upsell existing customers. Also, the more integrations your customers have with your product, the stickier your product becomes (and the less likely they are to churn).
6. **Boost your win rate.** If your services organization is delivering eye-popping implementation quotes due to the cost of integration, or your pre-sales team is getting hamstrung by ad hoc integration needs from prospects, then an embedded integration platform can deliver integration solutions that increase your win rates while ensuring your customers don't get sticker shock.
7. **Increase customer satisfaction (CSAT).** Bad things happen when synchronizations break. Setting up integrations can require endless technical support calls. Worse, your customers' projects may get overrun due to integration effort. As a result, your organization will take a hit to customer satisfaction. Enabling painless integration delivery and setup can boost your NPS scores in a world where customers increasingly expect faster resolution.
8. **Improve your customer retention.** When your application is more integrated into your customers' stacks, that's a recipe for better retention. Boosting stickiness by using integrations can increase retention by 15% or more, and the more integrations you deploy for customers, the more your product becomes a mission-critical part of their infrastructure.
9. **Grow project profitability.** If you're in services providing fixed-price bids for integration delivery, using an integration platform instead of hand-coding integrations can reduce the effort in some cases by more than 90%. You can take the hours of engineering and maintenance you save right to the bottom line.
10. **Easier to hire and train for.** Whether you're in product or services, making integrations easy to create and deliver means you can fast-track training in-house and avoid hiring expensive integration experts for delivery.

## Case Study: Typeform

**In 2019, Typeform had developed just a few integrations and had a backlog of integration requests from customers. Then in 2020, the company adopted an embedded integration solution and developed 20 integrations. By 2021, it had developed 70 more integrations with the help of its partners and, by using its embedded integration solution, had delivered more than 100 integrations in total.**

Learn more: [How Typeform Went From 30 Integrations to 100+ in Just One Year, Crossbeam Blog](#)

# 10 KEY CRITERIA FOR CUSTOMER INTEGRATIONS

## 1. Assess integration development velocity, efficiency, and flexibility

You're in the market for an embedded integration platform to clear your backlog and accelerate your roadmap. And that means one thing: The platform you embed must enable substantially faster integration velocity than you have today. So you've got to get this one right—because if the platform you choose doesn't deliver velocity and developer efficiency advantages, it'll kill your ROI and risk a future tear out.



### Business case tip:

The less engineering effort required to build and maintain integrations than building in-house—the more significant the ROI on your embedded integration. It's that simple.

The solution you choose must speed connectivity to your end customers' stacks with built-in connectors (we'll cover what to look for later). But it also must enable the rapid development of reusable business logic, such as branches, lookups, data transformations, and error handling—all while ensuring as little engineering effort as possible with maximum reusability.

Modern low-code embedded integration platforms enable your development teams to visually build the whole integration workflow, from your customers' stack to your product, and all the logic in-between. You'll also want to look for solutions that give you the option to turn your integrations into reusable templates for any of your customers with similar needs to activate them. Of course, low-code isn't a panacea; it's essential to have the ability to add in code to enable more customization if needed.

Here's where things get tricky: Some integration platforms are code-first. Such platforms will provide some connector APIs, but your developers will have to write and maintain all the business logic in code, which is an effort that quickly adds up. Others are “low-code light”—they'll handle some rudimentary business logic visually, but developers will have to heavily augment with script or workaround the limitations entirely with code. Either way is less than optimal for speed, efficiency, and maintainability.

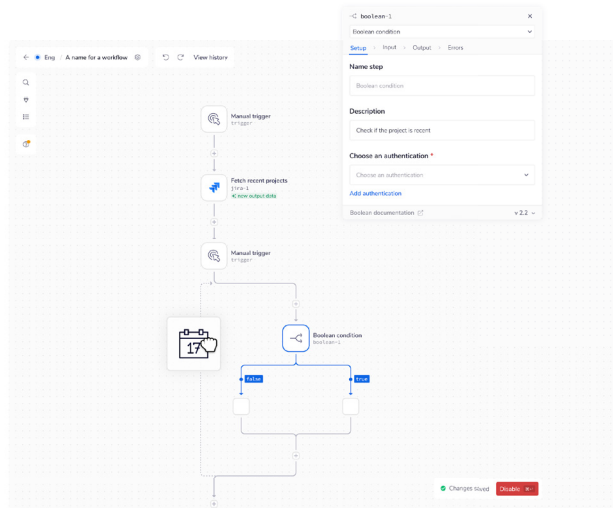
**“With our low-code embedded platform, we’re able to build integrations without the need to compete for engineering roadmap time. We’ve been able to deploy four times as fast”**



— Will Clifford, Director of Product, Integrations, LeafLink

[Learn more](#)

Finally, there's developer-grade low-code, which gives your team the power to build a substantial amount of integration logic visually, while requiring minimal code augmentation. It typically provides the best integration velocity with the least effort. And it's often also easier to maintain (because integration workflows aren't buried in code).



*Developer-grade low-code integration speeds delivery*


## What to look for in developer-grade low-code integration development


The best way to assess a solution is to get hands-on, and compare tools based on developer velocity. Look for free trials to see whether development or product management can build

and publish integrations fast—and whether updating them is easy

But here’s a low-code business logic checklist you won’t want to compromise on to minimize code and speed delivery:

- |  |   |  |
|--|---|--|
| <input type="checkbox"/> Loops, nesting, and delays          | <input type="checkbox"/> CSV and flat-file processing       | <input type="checkbox"/> Error handling                  |
| <input type="checkbox"/> Multiway branches                   | <input type="checkbox"/> Data storage                       | <input type="checkbox"/> JSON reader                     |
| <input type="checkbox"/> Call sub-workflows                  | <input type="checkbox"/> Lookups                            | <input type="checkbox"/> Crypto functions                |
| <input type="checkbox"/> Text transformations and processing | <input type="checkbox"/> Visual field selection and mapping | <input type="checkbox"/> Phone number helpers            |
| <input type="checkbox"/> Date and time transformations       | <input type="checkbox"/> Math functions                     | <input type="checkbox"/> Add Javascript/Python if needed |
| <input type="checkbox"/> List manipulations                  | <input type="checkbox"/> Parameterized workflows            | <input type="checkbox"/> XML and XLSX                    |
| <input type="checkbox"/> HTML DOM parser                     | <input type="checkbox"/> Zip and compression helpers        | <input type="checkbox"/> JDBC client                     |

  
**Evaluation tip:**  
It seems like everyone says they’re low-code these days. Get hands-on, try to build your most pressing integrations—and see if they’re up to the task in terms of speed and functionality

  
**Integration tip:**  
Ensure the integrations your team develops can easily be turned into parameterized templates—to maximize reusability and minimize one-offs

**“With our low-code embedded integration platform, we can re-task roughly 60% of our integration squad to work on other projects and build even more integrations.”**



— Kabir Mathur, Director of Product Partnerships, [Typeform](#)



## 2. Evaluate not only scalability—but lights-out elasticity

The good news is that most integration platforms will scale for embedded integration needs. But whether your measurement is the number of API calls per second, transactional volume, inserts, upserts, or requests, of course, you'll want to validate it and put the vendor through its paces on your specific metrics.

But if you're looking at embedding at scale—rolling out integrations to hundreds, thousands, or even tens of thousands of end customers, pure scalability isn't enough. You need to also understand the operational cost to achieve scalability, that is, how much operational overhead you'll need to meet end customer demands including monitoring, provisioning, and ongoing sizing to deliver with your embedded integration platform.

Here's where different compute models come in. Older-generation integration platforms are server-centric, which works up until a point. Depending on the integration vendor, you'll be looking at sizing and adding workers, vCores, or perhaps Atoms and Molecules to support end customer growth. Typically, you'll pay a fee to the vendor for each worker—and may end up paying more for anticipated or peak load.

The more significant issue is that monitoring server-based “workers” on older platforms can be painful. In many cases, the only indication you get that you're running out of resources is when end customer performance issues start to arise. Other issues with older technology include slow response times and frequent timeouts, application restarts, or errors in the logs that relate to performance issues.

### **Modern serverless computing enables embedded integration at scale**

You're probably familiar with serverless computing. If not, here's a quick primer: It's a cloud computing model where the provider dynamically allocates the exact resources needed, on-demand. In a true serverless architecture for an embedded integration platform, each step or task in an integration flow (such as a trigger, transformation, or insert) is a serverless function. Serverless architecture elastically allocates resources on demand.

Because there are no persistent workers, there's no need to size or pay for anticipated demand. And there's no need for your ops team to continually monitor dashboards and logs for customer issues related to under-sizing. As a result, serverless is a recipe for elasticity and end customer integrations at scale.

One final consideration on scalability: While an embedded integration provider must be able to scale elastically, it shouldn't risk uptime and availability. Ensure your provider includes a financially-backed SLA with 99.5% uptime and delivers transparent, updated status and a strong track record of availability for all its embedded and platform services.

## Case Study: **eventbrite**

### Why Eventbrite chose a serverless embedded integration platform

**Eventbrite (NYSE: EB) is the global self-service ticketing platform for live experiences. The company needed to deliver self-service integrations to its customers at a massive scale. It has more than 650,000 creators on its platform, managing more than 4.6M events in nearly 180 countries. As a result, any self-service integration Eventbrite rolls out can easily create a surge in demand on its embedded integration platform**

**By choosing an entirely serverless embedded integration platform, Eventbrite enabled more than 59,000 active customer integrations in less than 12 months without adding any staff to its operations team. So, while it was able to reduce the number of engineers for each integration from six to one—more importantly, Eventbrite avoided the massive operational overhead from integrations at scale that comes with older, non-serverless integration platforms. [Learn more](#)**

### **3. Sweat the details on end user activation (or face the consequences)**

One of the biggest reasons embedded integrations fail is that the end user marketplace activation experience is painful. So let's take a moment to name and shame some of the recipes for failure:

**“The Homegrown Horror.”** Code-first embedded integration platforms' inflexibility can lead to significant customer pain. When using such tools, your development team may have embedded some connectors, but the platform doesn't provide an end user activation experience. So, your development team must build out an activation experience in-house, which requires enabling end users to easily authenticate with their apps, allowing them to pick parameters and map fields step-by-step—in a way that's flexible for different integrations. What often happens instead is the activation experience ends up being limited, it breaks, or it's inflexible, and it ultimately drags down CSAT.

**“The Wait-What? UX.”** You can't just IFrame an entire, unguided workflow builder and brute-force it in your application—it'll clash awkwardly with your product's native UX and be painful for users to learn. And you don't want an embedded integration activation that doesn't feel

like your app because it looks like a whole new page or renders in a different style. When your users perceive they are no longer in your app, it naturally creates a bad user experience. Not to mention, your competitors will happily use your product's poor UX as "fear, uncertainty, and doubt" (FUD) in competitive evaluations.

**"The Tech Support Takedown."** Another painful avenue for activation is a high-touch experience that requires too much hand-holding to get up and running. Think detailed configuration, tracing error codes in logs, and client services getting involved. An activation experience that's too complicated requires calls to tech support to get working. As a result, your resources, margins, and customer satisfaction all take a hit.

For many integration vendors, providing an embedded, customizable end user activation experience is an afterthought. Instead, ensure your vendor has invested in an embedded integration experience that your team can customize, embed, and is simple for your end user to engage with.

**"We needed a solution with a pre-built interface that could seamlessly embed in our existing integration marketplace. The solution we chose offered a unique approach to help us deliver high-quality customer integrations at scale, eliminate complexity in the backend, and present a clean interface to our customers."**



— Martijn Russchen, Senior Product Manager, HackerOne

[Learn more](#)

## What to look for to deliver a robust and scalable end user activation experience

**Complete set of APIs to list all available integrations within your app**

When a user heads to your integration marketplace, or the integration page in your app, you'll need the vendor's APIs to query the integrations that are available and present in your UX

**Easy for developers to publish new integrations to your app or marketplace**

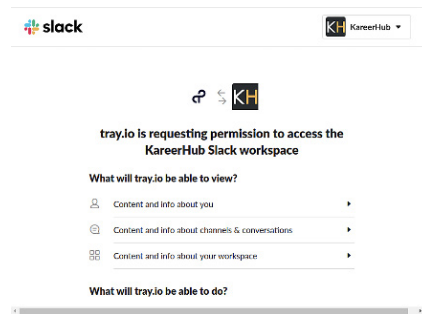
Ensure development can build integrations as templates. Your embedded integration solution should also provide a simple way to expose configuration parameters required for end user activation

**Provides a dedicated configuration wizard component**

Don't settle for "code-it-yourself" or simply an IFrame embed of a solution's standard builder—look for an integration component designed for end users to activate parameterized integration templates, in-product.

Enables end users to authenticate with their apps easily

Ensure the end user activation component supports common authentications such as OAuth 2.0, to self-activate.



Guided drop-down menus and parameters

Ideally, a configuration wizard should walk the user through the process, with the correct parameters and pick lists based on the integration at hand

Rebrandable and CSS

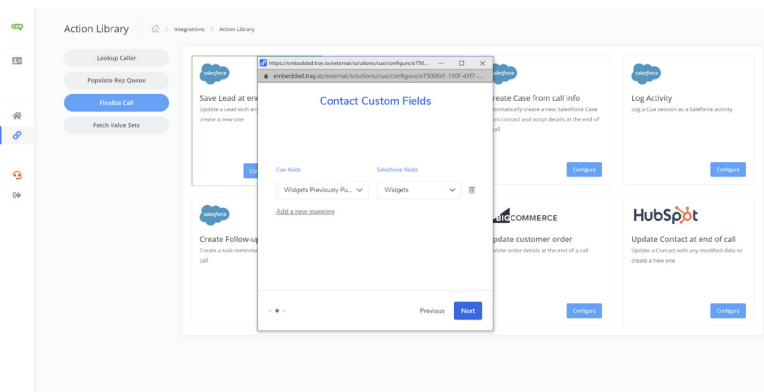
Supports adding your CSS to match your branding/look and feel

JavaScript customization

Look for solutions that give you control over customizing each activation step, such as granular control over prompts and logic

APIs to manage the entire activation experience, if needed

If your team plans to control the experience even more tightly, then ensure the vendor provides APIs to authenticate, query parameters, and activate the integration



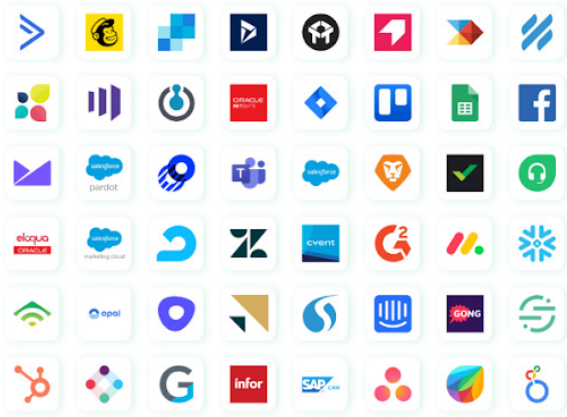
*Cue chose an integration activation experience that seamlessly embedded into its app*

#### 4. Ensure your roadmap won't hit brutal customer connectivity roadblocks

If there's one truth these days, it's that you'll need a lot more connectors for your roadmap than you think. It turns out that the average enterprise now has—wait for it—1,295 cloud services, and it keeps rising every year. So the days of an integration platform only delivering a few pre-built connectors are gone—you'll need closer to five hundred to have a comfort zone for current and future customer requirements.

But there are other considerations too. For example, your customers will often heavily customize apps such as Salesforce, Marketo, and NetSuite, so you'll need to deliver integrations that handle non-standard custom data fields (or standard fields that they might, dare we say, "misuse"). So, if your team develops a productized integration, it'll have to easily connect to each customer's unique customizations during activation.

Not only do your integrations have to be flexible enough to handle customization, they must be user-friendly enough for your end customers to do some self-service mapping themselves to their specific customizations at setup—ideally using a wizard. That way, you (and your services and support teams) won't be on the hook for every customer activation and mapping exercise.



And sometimes, even the most extensive connector libraries are missing connectors, or connectors are missing endpoints—and that can't be a delivery deal-breaker. So it's important for an embedded integration solution to provide you with other options. For example, you'll want to have access to a connector SDK. Alternatively, you'll want to ensure your vendor can connect to any REST, SOAP, or GraphQL endpoint outside of standard connectors. And ideally, your integration platform provider should be able knock out a new connector in a few weeks (not months), on-demand.

**“Our embedded integration platform has allowed us to quickly deliver high-quality, self-maintained integrations for our customers.”**



— Dan Peterson, Sr. Director of Product, Eventbrite

[Learn more](#)

## The critical connector questions you've got to ask:

How many connectors are included as standard?

Is there enough connectivity for future roadmap plans?

What if the provider doesn't have a connector?

Is there an SDK to build a connector?

Can we connect to any REST, SOAP, or GraphQL API?

What if my end customers have on-premises apps?

How fast can the vendor build a connector for me?

What if my end customers have customizations?

Does additional connectivity cost extra?

Does the provider develop all the connectors?

How many connectors did you deliver last year?

How often do you refresh connector endpoints?

Are features such as API retry logic and exponential back-off standard?

Is it simple to trace connectivity errors?

## 5. Check if embedded integration is a first-class object for your provider

So, here's a dirty little secret about many integration platform providers: Providing embedded integration isn't their first order of business. For some of them, it may be far down their list of priorities. Instead, they're focused on selling to IT or to your end users. Those vendors don't care about things like rebranding or providing self-service activation wizards, APIs for embedding, or connector SDKs—but you do.

So it's crucial to assess how committed the vendor is to providing a top-quality embedded solution—and supporting you every step of the way. Without it, you run the risk of subpar developer support, leading to a roadmap that heads in a different direction from where your company needs it. In short, going with a vendor that doesn't prioritize embedded integrations is a recipe for failure. Here are some questions to start the vendor conversation:

- Do you have a dedicated product manager/team focused on embedded integrations?
- Is there a specific roadmap for embedded integration?
- What embedded integration enhancements have you delivered over the past year?
- Do you have developers solely focused on your embedded product?
- Is there a specific enhancement request queue devoted to your embedded product line?
- Is there a developer support team to handle ongoing questions?
- Show me customers like me that are successful embedding your solution?

**“Our embedded integration provider was with us every step of the way. The team’s implementation specialists went above and beyond to make sure we were supported and enabled us to move forward quickly. It didn’t take long for us to get to a place where we could start building at a pace that met our customers’ need for integrations.”**



— Martijn Russchen, Senior Product Manager, HackerOne

[Learn more](#)

## 6. Make sure there are modern and complete APIs for your engineers

If you are planning on productizing your integrations, that is, providing them “in-product,” then the platform must offer a robust set of APIs for complete control. Earlier, we covered the importance of developer-grade, low-code development in quickly building integration logic and workflows, but the product will still need APIs to manage and embed.

Many integration vendors provide REST APIs. However, it’s essential to ensure the vendor actually designed the APIs to cover the specific needs of embedding and solution management. Also, more-modern integration vendors provide GraphQL APIs, which can accelerate the process of embedding the platform.



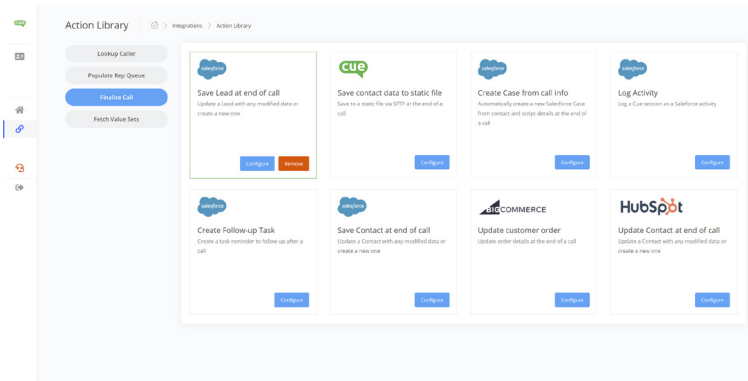
**Tip:**

### **GraphQL APIs can speed up embedded development versus REST**

GraphQL, originally an open-source project by Facebook, is increasingly becoming a more-efficient, flexible, and powerful way of working with APIs than REST. In GraphQL, developers can pick the fields they want, and as a result, GraphQL requests are always smaller and more efficient. In addition, developers can spend less time going through excess data and more time on things that move the needle by only working with the specified data set.

## What to look for in Embedded APIs

- **Solutions management.** When your product team creates integration templates that users can activate in-product, you'll need a programmatic way of querying available integrations for display.



*Cue uses its embedded platform APIs to list out integrations in-product*

- **User management.** Look for APIs that return your end customer user IDs, and can enable creation/delete/update of users (and other areas), so you have complete control of your end customers' access, in-product
- **Authentication management.** It must be easy for developers to create a new user authentication for a service (for example, to Dropbox or Airtable) within your app and delete it when needed.
- **Updates and version management.** The integrations your team builds will change over time. As such, you must ensure you have access to APIs that flag changes to integrations, such as when end users require new information and parameters. You'll also want APIs to push integration upgrades with new config data, if needed, automatically.
- **Integration management.** This one goes without saying, but be sure your vendor's platform makes APIs available to activate integrations based on your end user's preferred configuration and parameters (with the ability to delete when no longer needed).
- **Data query.** Another valuable feature to look for: Robust APIs to pull data from a particular service connector operation and display it within your app. As an example, a specific end user of your embedded application might need to display all the 'warm' Salesforce leads assigned to them for a sales use case, or display all customers with overdue payment balances for a services/finance use case.

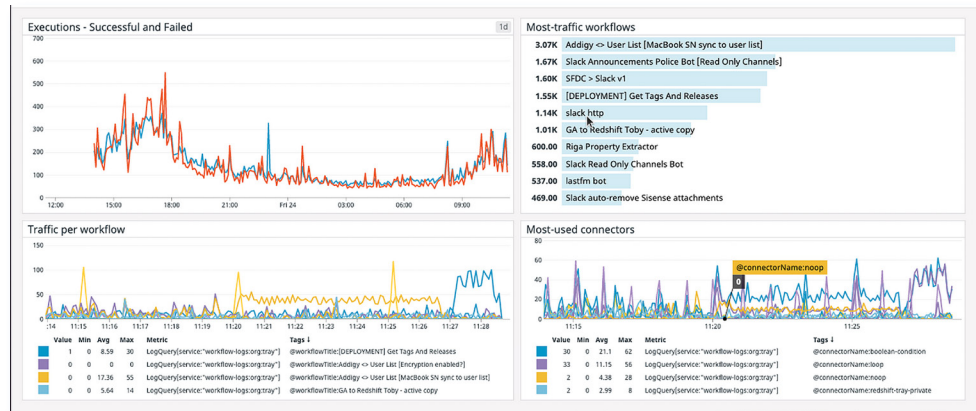


## 7. Drill into maintenance, monitoring, and management

Whether you plan on deploying integrations to tens, hundreds, or thousands of customers, ensuring you can effectively manage, maintain, and monitor your embedded integration platform is crucial to avoiding operational costs and CSAT issues.

### Look for integration to application performance monitoring (APM) tools

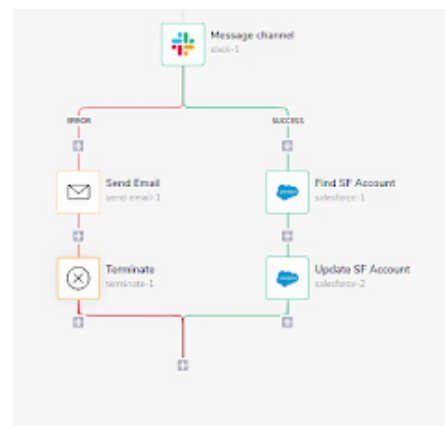
Built-in analytics to monitor performance can be helpful, but it's typically better to stream logs to dedicated third-party services for maximum operational visibility at scale. Look for integration platforms that provide built-in log streaming to simplify output workflow performance, executions, and trends to services such as Datadog, Sentry, New Relic, or Kibana for further analysis, and so that you can display statistics in a meaningful dashboard.



Monitoring an embedded integration using log streaming in Datadog

### Consider the importance of visual error handling

Every integration you deploy should have robust error handling built-in to exit gracefully, continue, or alert the team on a failure. Unfortunately, if your error handling is buried in code, your technical support teams will often find it challenging to provide support. Your engineering team will also find it hard to continuously improve your integrations in order to consistently deliver for your customers, especially if there are potential error conditions hidden on different steps and branches of the flow.



What makes error handling easier to manage and maintain? Low-code, visual environments simplify your job considerably, since they make it easier to design robust integrations and continuously improve them. In addition, when paired with log management, visual interfaces enable technical support teams to diagnose customer issues quickly.



#### Evaluation tip:

If your engineering teams use visual, low-code platforms to build out integrations and error handling, your technical support team will thank you. Having visual workflows makes it much easier for support teams to trace and diagnose customer support issues.

### Don't compromise on version control and update management

Here's where embedded integration platforms can differ substantially from their standard integration counterparts. Embedded solutions must have a way for your product team to push updates to end customers' workflows. Without an efficient way to send updates, you'll find that maintenance issues will pile up quickly. Having to handle updates across your customer base individually will add up to costly, painful maintenance work that never ends. Look for multiple options to deploy updates:

- **Lights-out automatic updates when possible.** When you have a new version of an integration built, your team should be able to roll it out automatically to your end users if they require no extra configuration.
- **Easily manage updates when new user information is required.** For example, suppose your team updates an integration to require new information, such as a new parameter or an updated authentication. Your embedded integration platform should make it simple to prompt the user to supply the required information to complete the upgrade.

### Ensure complete lifecycle management, testing, and sandbox

Whether you're working on a product-led integration or services-led integration project, solution management is critical. After all, you need to fully build, iterate, and test your solutions in a development environment (such as using test/sandbox accounts and test data) before pushing them to a production environment.

Modern embedded integration platforms make it simple to export your workflows from your development environment in formats such as JSON documents and import them into your production environment, either manually or by using GraphQL APIs to automate the process.

## Check for complete log management and diagnostics

Robust logging and clear error codes and diagnostics are crucial for quickly resolving customer and integration issues. When debugging more-complex workflows, logging and error codes are also vital so developers can see what each workflow step receives (such as the input) and compare it to what they expect to happen. Look for embedded integration platforms that:

- Indicate which runs are successful, processing, failed, or stopped
- Provide an easy way to drill down into any workflow to view how many steps your workflow has completed and their status
- Has robust search built-in, with the ability to filter for factors such as all executions before or after a specific date, or based on successful, failed, running, or specific workflow steps.
- Ensure transparency into data and execution behind each step in a workflow, showing inputs, outputs, and data, ideally in standard formats such as JSON.

## 8. Ensure options for both productized and services-led integrations

Take a close look at your company's current integration trajectory, and you'll see two distinctly different delivery models. You may not even need both now, but you may end up growing from one into the other in the future.

### Productized integrations

One delivery model is productized integrations. We're talking about relatively cookie-cutter integrations that you can provide in-product or via an integration marketplace. Such integrations will vary little from customer to customer. You can handle any differences across customers by parameters that customers can fill out when they self-activate the integration within your product.

Typically, your development team will build and maintain productized integrations (such as syncing customer data between a particular cloud app and your product), since such integrations are generally for self-activation at scale across your customer base.

For productized integrations, the platform you embed should provide:

1. Custom white-labeling
2. An integration activation component for end customers
3. Development of reusable, parameterized integration templates
4. End customer authentication management
5. Robust APIs to enable a seamless embedded experience

### Services-led integrations

The other delivery model is services-led integrations. As the name suggests, your services organization (which may include integration specialists) implements such integrations, often as part of a statement of work (SOW).

**“For our services team, it was critical to have a platform that non-developers could build customer integrations for workflows on, and transition to an embedded, self-service model, to fully offload integrations over time.”**



— Clark Hager, Client Solutions Director, Bizzabo

Often, services-led integrations are different because they may have higher variation from customer to customer, which makes it impractical to productize them. For example, Customer A might need a homegrown system integrated with your application, while Customer B might have an ERP it has heavily customized, and so on. So while you can templatize certain aspects of your integrations, services will ultimately be scoping, quoting, and building each for custom delivery. Or perhaps your services team is building out a specific integration for the first time for a customer, which means that services teams are looking to take the lead—and that it’s unclear whether the integration is worth productizing.

For services-led integrations, your embedded integration solution should provide:

1. A highly flexible, low-code integration builder
2. Collaboration and secure workspaces for project development
3. Single sign-on (SSO) for integration builders
4. On-premises integration to connect complex stacks
5. Easy-to-consume learning materials to enable services teams



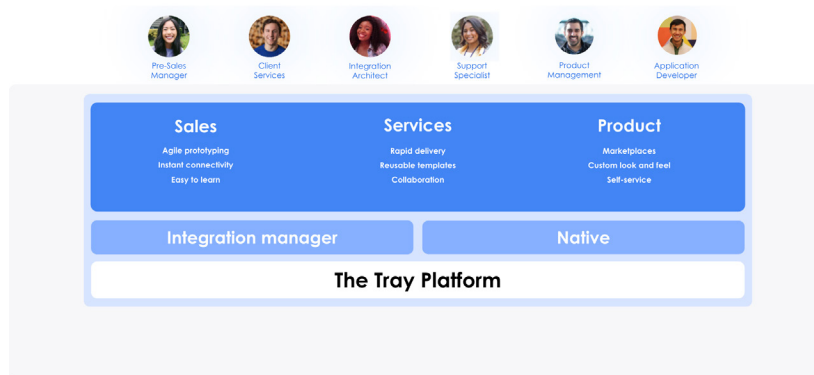
### Evaluation tip:

Services-led integrations may have a broader array of integration pattern needs depending on customer SOW. The scope for such projects may range from bulk-data integration for analytics, to real-time data flows, or triggering integrations from webhooks or application triggers—so it’s a good idea to ensure your embedded integration platform is flexible enough to handle all such use cases.

## Ensure your platform can handle both delivery models

Some embedded integration platforms are code-centric. They’ll provide a whole set of REST APIs designed to ease connectivity. But because code-centric platforms are primarily for developers, such platforms are typically light on providing flexible tooling for low-code integration development. So, while code-centric platforms may work for productized integrations, such platforms are ultimately impractical for services-led integrations—because coding and maintaining each specific customer integration is too technical, time-consuming, and maintenance intensive.

Other embedded integration platforms are specifically for services-led integrations (often with roots in IT-led delivery). Services-first platforms may provide tooling for visual integration development, but are a poor fit for productized integrations. For example, such platforms may have legacy APIs that make embedding a poor fit with your more-modern codebase, or they may be hard to customize for look and feel, or they may lack a self-service component for in-product activation.



*Ensure your integration platform can handle multiple delivery models*

If you choose an integration platform that's only good for one use case, chances are you'll end up having to add another platform in the future for the other. Worse still, you may have to migrate to a totally new platform that can handle both scenarios later on. As such, choosing an embedded integration platform that can handle both productized and services-led integrations from the start tends to be the smarter play.

### A single platform for services and developers enables reuse

**“We iterate an integration across multiple customer requests until we end up with a really solid integration template that suits all of their needs. We develop our template through various configuration factors in our processes that gather across different implementations on various customer accounts.”**



— Will Clifford, Director of Product, Integrations, LeafLink

[Learn more](#)

There's one other significant benefit of everyone being on the same platform across your services and product organization. Services teams can start by building integrations based on specific customer requirements. They can templatize integrations that have repeatability to improve their margins and accelerate delivery. Even better, they can hand off highly repeatable integrations they have built over to the development team, who can then turn them into productized integrations. As services and product teams create and templatize integrations, smart companies use the process to create a full-on integration pipeline.

## 9. Get into the weeds on certifications and security

Ensuring your provider meets the most stringent security, compliance, and privacy requirements is critical to your company's reputation and financial integrity, as well as your end customers' trust. Among other things, it's vital to confirm that your embedded integration vendor has information security (infosec) measures, procedures, and policies, backed by certifications and annual audits for SOC 2 Type 2, along with independent penetration tests.

You'll want to ensure your provider has fully documented both its technical and organizational security measures (see the complete list below). Also, embedded integration vendors must

provide onboarding and annual data protection and information security training for their staff. Your vendor should also have a nominated data protection lead (such as a security and compliance officer). And your embedded integration vendor should also provide complete transparency over the use of data through an up-to-date privacy policy.

## Key certifications

Certification	What to expect
<p><b>Must be SOC 2 Type 2/3 compliant</b></p>	<ul style="list-style-type: none"> <li>- SOC 2 Type 2/3 compliance should be a minimum requirement—not a topic for compromise. Ensure your embedded integration provider is SOC 2-compliant, which means it has passed an audit checking infrastructure, tools, and processes to protect its information from unauthorized access both from within and outside the firm. In addition, if your provider is SOC 2-compliant, it will have met the AICPA's trust principles around security, availability, processing integrity, confidentiality, and privacy</li> <li>- In addition to your SecOps teams demanding it, SOC 2 compliance ensures your provider meets your end customers' need for assurances too</li> </ul>
<p><b>Must be GDPR compliant</b></p>	<ul style="list-style-type: none"> <li>- Ensure that the vendor can be listed as a GDPR sub-processor so you can meet your GDPR compliance obligations. For example, your company probably lists companies such as Amazon, Microsoft, or Google as sub-processors, and you'll need to list your embedded integration provider too</li> <li>- Your provider must be ready to sign a data processing agreement (DPA) with your company that includes breach management and notification, data retention, a data transfer risk assessment, impact assessments, assistance to controllers, and records of processing</li> <li>- A DPA, part of GDPR rules, is a legally binding document to be entered into between the data controller and the data/sub-processor. A DPA regulates the particularities of data processing, such as its scope and purpose, as well as the relationship between the controller and the processor.</li> <li>- The DPA should cover applicability and scope, roles and responsibilities, security, sub-processing, cooperation, security reports and audits, deletion or return of customer data, transfer mechanisms, and other items</li> </ul>
<p><b>Must be CCPA compliant</b></p>	<ul style="list-style-type: none"> <li>- The California Consumer Privacy Act (CCPA) regulations went into effect on 1 January 2020. CCPA applies to California residents and is enforceable for any company with revenues larger than \$25 million and has more than 50,000 people or devices in its database.</li> </ul>
<p><b>HIPAA compliant, if you're managing healthcare data</b></p>	<ul style="list-style-type: none"> <li>- If your company manages PHI data (protected healthcare information), you'll want to ensure your vendor is HIPAA compliant.</li> <li>- Confirm your vendor is compliant with the HIPAA Security Rule and the HITECH Breach Notification Requirement and that it has independent audits to verify.</li> <li>- Ensure your vendor is willing to sign a BAA (Business Associate Agreement)</li> </ul>

**Next up, drill into the technical aspects of your provider's security, disaster recovery, and operations, such as:**

- **Encryption.** Ensure your integration provider encrypts all data, authentication, and tokens in transit and at rest
- **Network security.** Your provider should confirm that it can encrypt all communications between its embedded integration platform, your company, and end customers via HTTPS  
Customizable log data retention. The vendor should offer flexible log data retention policies designed to fit your customer data processing requirements
- **Session management.** Your provider should monitor sessions by IP address, location, time, browser, and operating system and should be prepared to revoke access to prevent unauthorized access to your account
- **Two-factor authentication.** For your developers, services, or administration team building and managing integrations, the provider should offer two-factor authentication for a second layer of protection
- **Behavior modeling.** Your provider should have the capability to auto-detect unusual or suspicious activity on a user's account
- **Penetration testing and vulnerability scans.** Your provider should undergo regular penetration testing by independent third parties to ensure that its platform is secure. The vendor should also perform vulnerability scans on internal and external cloud-hosted systems. In addition, your provider should perform dynamic app security testing to ensure your data is safe from outside threats.
- **Daily offsite backups.** Your provider should perform backups at least daily, and store them in a separate geographic location. Backups should have at least the same level of security and encryption as above.
- **Strong RPO and RTO for disaster recovery.** Look for vendors whose disaster recovery (DR) procedures are designed for a Recovery Time Objective (RTO) of 14 hours and a Recovery Point Objective (RPO) of 24 hours.



## 10. Avoid contract nightmares. Negotiate predictability and simplicity

Let's cover some examples of potential contract challenges. First example: You sign an API volume usage-based OEM contract for an embedded integration platform. After a year, your actual usage is much higher than the contracted estimates. Millions of dollars over the limit. So, it's time to call the vendor's account manager for an emergency renegotiation. It's not a great position to be in—but more common than you think.

Second example: You sign the OEM deal, but that same year into it, your customers are experiencing slow performance and API timeouts. It turns out you've got to buy more worker cores from the vendors for the load—and it adds up to \$100Ks' more in license costs. You didn't want to buy too many cores upfront because you were never sure you'd need the capacity—and now it looks like you'll have to.

The fact is, with integration, it's easy to find yourself in unexpected situations based on how many embedded integration platforms approach commercial terms. For example, sizing API volumes is complex, as is estimating throughput when you haven't built or deployed your first integration yet. The problem with models that use such estimates is that they are notoriously hard to predict and often misaligned with the value being generated for the business.

**“Application leaders often focus on short-term needs, which results in nasty ‘price shocks’ when platform use scales beyond the terms of the initial contract.”**

— How to Compare the Disparate Pricing Metrics of Integration Products, Gartner,  
26 October 2020

### 1. Pick an integration usage metric that won't cause future headaches

It is crucial to ensure you directly align your contract terms to the value that your integrations deliver for your customers and company—using metrics that are predictable and scale with growth. Usage metrics like API usage or transaction volumes are hard to predict, hard to assess in terms of how much value they deliver to the customer—and even harder to ensure it's profitable to sell integrations around, especially if you're planning to charge for integrations through fixed-price subscriptions.

Instead, it's better to rely on usage metrics for which it's easy to estimate both costs and

margins. For example, a vendor’s contract might focus primarily on when your customers’ users authenticate, access, and ultimately activate an integration. As a result, it may cost you a fixed amount per end user authentication to an integration. You’ll know your margin, you’ll know you’ve delivered value since your customer is happy, and end user authentications are inherently predictable, without the pain of “renewal shock.”

## **2. Pay only for the minimum quantity to get started and “true-up” periodically**

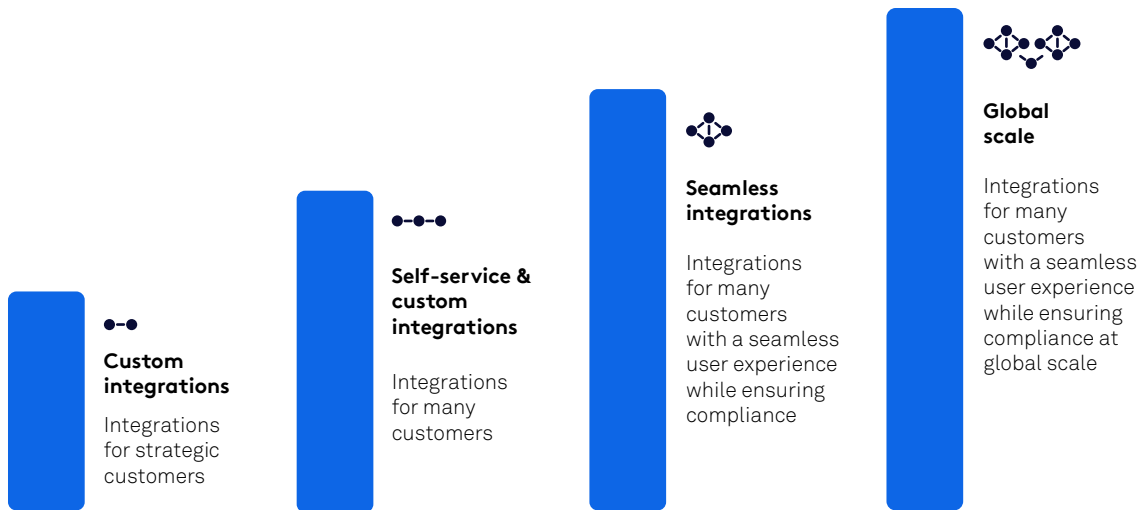
Once you’ve moved away from unpredictable metrics or worries about opaque capacity issues by settling on the right metric, you’re in a stronger position. There’s less need to over-purchase to overcompensate for estimation errors because you can now accurately size usage.

So, while you may want to make your purchases a little ahead of usage and typically be able to lock in some discounts for pre-purchasing—ultimately, you want to closely align the usage you are paying for with what your customers are using. Then, you can set up quarterly “true-ups” with your vendor as usage grows.

## **3. Finally, look for pricing that is designed around your integration journey**

There are a few items you should consider to be non-negotiable. The basics: Expect your vendor to include all connectors with your initial contract. Even at the start of your integration journey, unexpected customer integrations will come up, and you shouldn’t pay extra for them.

It’s also important to ensure your integration vendor provides purpose-designed editions that suit your specific needs. For instance, your organization may just be getting started, and may need to handle ad hoc integrations for strategic customers—which means you don’t need white-labeling or self-service marketplace capabilities. Or, you might be looking to deploy productized and services-led integrations at scale globally with more-robust governance and compliance. There are some key benefits to working with a vendor that offers custom packages—namely, you won’t end up over-buying for functionality depending on where you are in your integration journey.



*Look for your embedded vendor to provide editions mapped to your journey*

# Your embedded integration checklist

## Integration development

- Developer-grade low-code
- Enable development of integrations as reusable templates
- Loops, nesting, and delays
- Multiway branches
- Call sub-workflows
- Text transformations and processing
- Date and time transformations
- List manipulations
- HTML DOM parser
- CSV and flat file processing
- Data storage
- Lookups
- Visual field selection and mapping
- Math functions
- Parameterized workflows
- Zip and compression helpers
- Error handling
- JSON reader
- Crypto functions
- Phone number helpers
- Add JavaScript/Python if needed
- XML and XLSX
- JDBC client
- PDF processing

---

## Scalability, elasticity, and availability

- Modern serverless architecture
- Handles elastic workloads without provisioning
- Published transparent uptime
- Financially backed SLA, 99.5% availability

---

## End user activation capacity

- Publish new integrations to your app or marketplace
- Provides a dedicated self-service integration activation component
- Enables end users to authenticate with their apps easily
- Guided drop-down menus and parameters for end users
- Fully re-brandable with CSS support
- JavaScript customization
- APIs to manage the entire activation experience, if needed

---

## Breadth and depth of connectivity

- Extensive library of connectors, all included as standard
- Provides a connector SDK
- Can deliver a new connector in weeks, if needed
- The vendor has a track record of rapidly delivering connectors
- Has a track record of updating/refreshing connectors
- Connects to any SOAP, REST, GraphQL API
- Provides options to connect to on-premises apps
- Supports features such as API retry logic and exponential back-off
- Transparent error codes

**Vendor embedded strategy and roadmap**

- The vendor has a specific roadmap for embedded integration
  - Product management/engineering team for embedded
  - Dedicated enhancement requests for embedded/OEMs
  - Track record of embedded integration enhancements
  - Documentation and training for embedded use cases
  - Staffed developer support team
- 

**Embedded integration APIs**

- Provides modern REST or GraphQL APIs
  - APIs for solutions management
  - APIs for authentication management
  - APIs for update and version management
  - APIs for integration management
  - APIs for data query
- 

**Maintenance, monitoring, and management**

- Instrumentation and analytics
  - Log streaming
  - Strong integration to Application Performance Monitoring tools
  - Visual error handling
  - Version control and update management
  - Lifecycle management, testing, and sandbox
  - Log management and diagnostics
- 

**Enable product-led integrations**

- Custom white-labeling
  - An integration activation component for end customers
  - Supports creation of reusable, parameterized integration templates
  - End customer authentication management
  - Robust APIs to enable seamless embedded experience
- 

**Enable services-led integrations**

- A highly flexible, low-code integration builder
  - Collaboration and secure workspaces for project development
  - Single sign-on for integration builders
  - On-premises integration to connect complex stacks
  - Easy-to-consume learning materials to enable services team
- 

**Certification and security**

- SOC 2 Type 2/3 compliant
  - GDPR compliant
  - CCPA compliant
  - HIPAA compliant, if you're managing healthcare data
  - Encryption
  - Network security
  - Customizable log data retention
  - Two-factor authentication
  - Behavior modeling
  - Penetration testing and vulnerability scans
  - Daily offsite backups.
  - Strong RPO and RTO for disaster recovery
- 

**Pricing predictability and simplicity**

- Easy to estimate
- Predictable usage metrics
- Editions designed for software and services companies

# The Definitive Guide to Embedding Integrations in Your Product

## About Tray.io and Tray Embedded

Tray.io is the world's leading provider of embedded integration, with software and services organizations everywhere delivering integrations powered by Tray Embedded to more than 50,000 of its customers. The world's most successful solutions providers run Tray Embedded, including Eventbrite, HackerOne, Typeform, and more. Companies choose Tray Embedded to power their integrations for our unique combination of broad connectivity to integrate any stack. In addition, Tray Embedded includes powerful, low-code, reusable integration development which speeds developer and services productivity. Finally, our fully re-brandable and embeddable platform provides a seamless experience to our customers, with full elasticity, governance, and control.

+1.415.418.3570 | [EMAIL](#) | [WEBSITE](#) | [BLOG](#)

## About this guide

Written by

Paul Turner, Tray.io

Copyright © Tray.io Inc. All rights reserved.

THE DEFINITIVE GUIDE TO EMBEDDING INTEGRATIONS IN YOUR PRODUCT  
VERSION 1.0 | 11.22.21