tray.io

THE BEGINNER'S GUIDE TO

Monetizing integrations

VERSION 1.0 | 04.04.21

TABLE OF CONTENTS

Introduction

For software companies, providing integrations for customers and sales prospects is both more critical but also more fundamentally "table stakes" than ever. In this guide, you'll learn how to turn customer integrations - traditionally a cost center - into a revenue driver by strategically building integrations into pricing tiers, marketplaces, or even as part of a freemium conversion model.

Why Integrations?

Every B2B SaaS team knows that their products don't exist in a vacuum. The SaaS explosion has created an endless array of highly usable, purpose-built applications that teams across the organization have readily adopted. Tray.io's recent State of automation report found that 40% of teams currently have more than 40 different applications in their tech stacks. And software acquisition isn't slowing at all. 83% of teams expect the number of applications they use to increase in the next three to five years[1].

So much software means that almost every business process spans multiple applications. Completing tasks requires teams to do work or create data within a specific application, which they must then share with other applications. Teams' growing need to share data across applications translates into integration requests. Buyers of software expect that the applications they buy will "play nice" with the other tools in their stack by providing easy, maintenance-free integration capabilities. Integration requirements show up both pre- and post-sale, derailing opportunities and customer renewal discussions.

40% **of teams currently have more than 40 different applications in their tech stacks.**

[1]State of Automation, Tray.io 2020

The integration backlog

Almost every SaaS product team has a significant backlog of integration features—the customer and prospect requests for integrations between their own software products and specific tools. And most of the integration requests remain in the backlog. There are two factors that push integration requests to the bottom of the queue: the perceived trade-offs associated with developing integrations, and the “cost-center” mindset that prevents teams from realizing the potential value of these features.

Product and engineering teams understand the complexity associated with building integrations. Unlike internal features, integrations are reliant on third-party APIs. In other words, it’s not enough to just learn the API of the tools you need to integrate and build a one-off feature. Instead, building integrations requires constant monitoring and maintenance. APIs aren’t static. As other products change, add, and remove features, their APIs adjust accordingly. An integration that works perfectly one day may fail the next when a specific call is deprecated. For product teams, agreeing to build an integration isn’t a one-time allocation of engineering resources, it’s an ongoing commitment. When evaluated against other product priorities, the disproportionate resource requirement of building integrations in-house often leads teams to deprioritize integration projects across the board.

The ongoing maintenance requirement of building integrations drives the mindset that integration work is primarily a cost driver. And unfortunately, software buyers often consider integrations to largely be “table stakes” features, so that the value they provide may not always outweigh the resources required. As a result, product teams are hesitant to develop integrations internally, generally push back on requests, and only relent when they receive a critical mass of complaints, or it’s a necessary requirement to close or retain a strategic customer.

“...software buyers often consider integrations to largely be ‘table stakes’ features ”

The idea that integrations are primarily a cost center misses two important points. First, there are many ways to deliver integrations, and not all of them require significant engineering resources. Second, while integrations might not appear to be highly differentiating features, there are many successful monetization strategies that turn them into significant revenue drivers. This guide explores a number of different ways that SaaS companies can monetize their integration features, and provides an ROI framework that companies can use to build a business case for them.

Adopting a monetization mindset

Rather than approaching integrations as a sometimes-necessary pain, SaaS companies should focus their approach around monetization and revenue. Yes, every company experiences integration pressure from customers and prospects, but having a strong strategy in place can help companies get ahead of inevitable requests. Effective integration strategy should focus both on the delivery model and the monetization approach.

The default position of many product teams is to think of integrations as product features—something that end users access and configure within the product itself. Considering integrations as features is a common, and highly visible deployment model, but it's not the only way to roll them out.

Another common approach to providing integrations is through a professional services model. Rather than trying to abstract and productize a broad set of customer requirements into a single feature, some firms develop and deliver integrations as bespoke implementations for individual clients. Yet another way in which companies deliver integrations is during the sales cycle. As part of a proof-of-concept (POC), pre-sales teams will develop a lightweight integration prototype to demonstrate a critical capability to a prospective customer. Some SaaS companies look to a partnership or community model that relies on third-party developers to build integrations. The partnership model typically pairs third-party builds with some type of marketplace where developers can list and sell the solutions they build.

Understanding the delivery model that best suits your product and customers can help alleviate some of the integration anxiety that keeps product teams from making investments. In many cases, there may be more-lightweight delivery approaches that are a better fit for customers and also require fewer development resources.

Delivery model is also an important prerequisite for monetization. There are a broad number of monetization strategies covered in this guide, but many of them align better with different delivery models. For example, charging per integration is a no-brainer when each integration is part of a custom services engagement, but it's a harder case to make for a simple three-click feature in the UI.

“SaaS companies should focus their approach around monetization and revenue.”

With a clear understanding of the optimal way to deliver integrations, and an appropriate monetization strategy, product teams will be much better equipped to address integration requests as they surface. However, being better equipped doesn't mean every company can and should implement every integration request, but rather that teams will have a much better framework to evaluate requests that takes into account both the resources required, and the potential returns.

Monetization approaches

It turns out that there are many ways to approach monetizing integrations. Some are direct, transactional models, some take a bundling or tiering approach, and others rely on integrations as growth drivers. There is no "right" model as fit depends on the type of customer you're selling to, their integration requirements, and your delivery model. However, there is almost always a way to turn integrations into a revenue generator.

While there is no "one-size-fits-all" model, there are some common guidelines that all monetization approaches should follow:

1. Aim for 10-15% of total contract value. Think about integration pricing in the context of your overall price point and the value that customers receive from your product. No customer is going to pay 50% more above the base value of your product for integration features—no matter how important they are.

2. Recurring revenue is key. Integrations aren't ever a "one-and-done" effort. Even with a great development and delivery process, there will be ongoing maintenance. End users understand this, and are generally willing to pay on an ongoing basis to ensure smooth, continuous operations of their workflows. One-time charges are easier to sell with some approaches, but can often result in SaaS companies leaving money on the table.

3. Position integrations as an add-on. It's harder (though not impossible) to monetize integrations if they're included into default packages. Relevant integrations provide tremendous value to customers that need them, but those that don't may resent what they perceive to be an additional charge. To be clear, your company shouldn't necessarily sell integrations one at a time in every case, but you should position them to command a premium over base product offerings.

10 ways to monetize integrations

While not an exhaustive list, the following approaches represent some of the most common ways in which SaaS companies look to monetize their integration offerings. Some are direct transactional models, some are built around packaging, and others are geared toward customer or channel growth.

1. Price per integration.

Pricing for individual integrations is the most straightforward model, and one that works well in a couple of situations. If your company is expected to deliver custom integrations on a per-customer basis—especially through a professional services model, it simply makes sense to charge per integration. Pricing per integration can also work if individual integrations customers perceive integrations to be particularly strategic or complex. It's usually easy to monetize any high-value feature separately, and integrations can often fall into this category.

One important consideration for product teams is whether to price each integration as a one-time fee or a subscription. While it's common to price professional services engagements on an hourly basis, companies should consider the fact that integrations often have a significant amount of post-production maintenance associated with them. Unless the customer is expected to maintain and support the integration themselves, some sort of subscription pricing should be used to cover ongoing support.

2. Create integration bundles.

This second monetization model maintains a transactional approach, but rather than charging individually, it uses one or more packaging bundles to sell relevant integrations to different customer segments. It's possible to align integration bundles to specific customer segments, such as a set of integrations to EMR, HCM, and billing tools for healthcare customers, or a combination of marketing automation and digital advertising tools for marketers.

Another approach is to bundle based on complexity. For instance, simple point-to-point integrations for productivity or project management tools could be in one (lower-priced) package, and a set of integrations with enterprise ERP or CRM tools could be in a higher-level bundle. Just like other software features, the strategy here is to create packages that align with customer values.

3. Build an integration marketplace.

Many large SaaS companies—especially those that sell into consumer or SMB markets—end up creating some sort of solutions or integrations marketplace for integration features. Marketplaces are an easy way for end users to discover integrations, and they provide a vehicle for third-party developers to participate.

Having an active development community and a way for third-party developers to monetize the integration solutions they build can take some of the integration burden off of your own development teams. However, marketplaces come with the burden of supporting third-party integrations. It's common to price marketplaces on a pay-per-use basis or to charge a universal access fee.

4. Add to standard offering.

Yes, including integrations with your base product largely runs contrary to the best practices we outlined earlier, but there are times where it makes sense to include integration features into a standard or base offering of a product, rather than positioning them as an add-on. Packaging integrations with your base product can be useful if an integration provides value for a broad set of users, and can justify an overall price increase. Integrations to very common enterprise tools like major CRM systems can be deployed this way. In this way, integrations can be a useful tool for product teams to quickly grow a product's average sale price (ASP).

5. Create premium tiers.

Premium tiering combines the idea of integration bundles with integration feature inclusion in a core product offering. Rather than adding integration features to a standard offering, companies can use tiers as the basis for additional, higher-priced solution packages. For example, it's possible to offer integrations to major enterprise tools (such as ERP, CRM, and/or HCM) as part of an "enterprise" tier. Effectively, integrations can serve as additional differentiation between tiers when customers don't perceive enough value to upgrade.

6. Freemium conversion.

For companies that rely on a freemium model for growth, integrations can be a powerful lever to increase conversion rates. Freemium versions of a product can offer no, or limited integration capabilities, and place more-advanced capabilities in a paid tier. One strategy to drive conversions is to initially offer an "integration trial period" for new sign-ups during which all features are available. Once customers have live data from your product flowing across their stack, they will be more likely to pay to keep the integrations running.

7. Customer support add-on.

It can make sense for companies that primarily deliver integrations to include them as part of a service offering. If customer requirements are too specific or varied for a true integration product feature, another way that teams can productize is through support offerings. It's possible to more-specifically define support options than with open-ended services engagements, and more importantly, it's possible to structure support options as a subscription. A subscription model can capture revenue to cover the ongoing maintenance of any customer integration—even when the integration is part of a custom solution.

8. Customer support tier.

Similar to a support add-on, but in this instance, it's possible to add support for one or more custom integrations to a specific support tier or package. Support tiering makes sense if integrations are a regular, consistent customer requirement. Adding integrations to a support tier alongside faster response times or extended implementation support can make premium support tiers more attractive to prospective customers.

9. Expand partner channels.

Leveraging third-party providers like system integrators can be a great approach to manage custom or professional services integration delivery. Third-party providers can take the integration development and maintenance burden from your team, and can become a growth driver for the business as they recommend or even resell your offerings.

Companies can use integrations as a way to cultivate the lucrative partnerships between themselves, partners, and customers. Providing a useful integration toolset that makes it easier for partners to build integrations can encourage companies to build a practice around your offerings—providing integration, implementation, and ultimately driving growth for you.

10. Increase retention rates.

Not technically a monetization technique, but it's important to note the role that integrations can play in making your product "sticky." Customers that deploy integrations are less likely to churn, and integration features themselves can drive a 30%+ increase in retention rates. Factoring in the increased revenue from a higher customer lifetime value can sometimes completely justify the cost of building and maintaining integrations by itself. In situations like this, especially where lack of integrations is a known customer pain point, not placing a premium or separate price on the integrations you deliver can make sense.

It's important to note that an integration strategy doesn't need to rely on a single monetization approach. Companies can use multiple delivery and monetization models for different integration offerings. Even SaaS companies in similar spaces can have widely different approaches. For example, Eventbrite, an event management platform primarily for individuals and smaller organizations, uses a marketplace approach to deliver integrations to its users. Bizzabo, which is also an event management platform, uses a combination of approaches—providing some integrations as paid features, and others as custom services for its customers.

"Companies can use multiple delivery and monetization models for different integration offerings."

Regardless of approach, understanding how to monetize integrations helps any product team better evaluate and prioritize integration requests. Understanding monetization also serves as a key driver in forecasting the return on investment associated with integrations. Next, this guide will walk through an ROI framework to build a business case for integration initiatives to better understand their potentially positive impact on your business.

An integration ROI framework

While this guide is all about turning integrations into revenue drivers, integrations aren't without costs. There are, of course, development, maintenance, and sometimes software costs associated with any integration feature. However, the cost-center mindset can be paralyzing when product teams don't think through all of the potential returns that integrations can provide. The following section outlines a simple framework you can use to evaluate integration projects, and clearly understand whether or not there's a case to be made for specific integration opportunities.

“ the cost-center mindset can be paralyzing ”

Integration costs

This model looks at a five-year cost/payback period, but you can easily adjust the timeframe as needed for your own calculations. First up, we'll look at the costs. There are three basic cost categories to consider: the initial development costs, the ongoing maintenance costs, and costs of software. Costs are dependent on your approach to building integrations. Some companies choose to develop all of their integration capabilities in-house, in which case, required engineering hours determine development costs. Other teams outsource integrations to third-party consultants or developers, in which case the primary development cost is the consulting engagement fees.

A third path is to use embedded integration software to simplify and accelerate solution development. Embedded solutions abstract away much of the API endpoint complexity, and often provide low-code authoring tools that make it easier to develop integrations. Using an embedded solution, there is still some (much lower) engineering implementation cost, but there's an added software subscription license.

When using an embedded solution, integration maintenance usually falls to a combination of customer support and engineering teams. Generally, maintenance costs will be higher with third-party integrations as internal teams won't have as clear an understanding of the solution, and lower for teams that use embedded solutions for integration development. For this example, let's assume that we're leveraging an embedded integration platform to implement two integrations. Our cost structure would look like this:

Costs	Initial	Year 1	Year 2	Year 3	Year 4	Year 5
Software license subscription		$80,000	$80,000	$80,000	$80,000	$80,000
Implementation/development cost	$108,000					
Integration maintenance cost		$1,850	$1,850	$1,850	$1,850	$1,850
Total costs	**$108,000**	**$81,850**	**$81,850**	**$81,850**	**$81,850**	**$81,850**

The cost assumptions include 120 hours of initial development time for two integrations, two hours per month of maintenance and support, a fully loaded engineering cost of $90 per hour, and a shared engineering and customer support cost of $77 per hour.

Integration returns - incremental revenue

The first category of returns that we'll discuss are the specific revenue gains associated with integration features. We can now put our monetization strategy into play. For this example, let's assume that one integration is priced at $1,500 per year, and the second is positioned as a higher-value feature (integration with an enterprise system used by large companies) and priced at $2,500 per year. Revenue returns would look like this:

Revenue	Initial	Year 1	Year 2	Year 3	Year 4	Year 5
Total customers		300	350	439	555	708
Integration 1 purchased		30	5	9	12	15
Integration 2 purchased		12	2	4	5	6
Integration 1 revenue		$45,000	$52,452	$65,802	$83,300	$106,182
Integration 2 revenue		$30,000	$34,968	$43,868	$25,533	$40,380
Total revenue		**$75,000**	**$87,420**	**$109,670**	**$108,833**	**$146,562**

For our calculations, we're assuming a customer base growing 22% per year, an attach rate for the first integration of 10%, and an attach rate for the second integration of 4%. Using our assumptions, we already see a positive overall ROI for the integrations, but there are several other returns to factor into the analysis.

Integration returns - churn reduction

One of the direct benefits of integration features is that they help to reduce customer churn. A customer that has an active integration running is using data from your product as part of broader team processes or workflows. As a result, the "stickiness" of your product increases as customers will have to address stakeholder and system dependencies if customers decide to stop using your product. In fact, some companies see as much as 36% higher retention for customers that use integrations with an embedded solution.

Updating our model, let's assume that the current customer churn rate is 6%, and our average deal size is $25,000. If every customer that adopts an integration has a 20% lower churn rate, we'd see the following incremental revenue:

Churn reduction	**Initial**	**Year 1**	**Year 2**	**Year 3**	**Year 4**	**Year 5**
Total customers (no integrations)		300	350	439	555	708
Revenue (no integrations)		$7,500,000	$8,742,000	$10,966,980	$13,883,311	$17,696,968
Total customers (with integrations)		300	351	440	558	712
Revenue (with integrations)		$7,500,000	$8,769,489	$11,012,467	$13,950,958	$17,792,298
Revenue gain		**$ -**	**$27,489**	**$67,647**	**$67,647**	**$95,330**

Our example doesn't show a massive lift, but even a small improvement in retention rate driven by integrations would result in more than $200,000 additional revenue over five years.

Integration returns - increased win-rate

One additional place where companies can realize benefits from integrations is in new business win rates. In the same way that integrations can improve customer satisfaction and retention, they can also make your product more attractive during the sales cycle. While some integrations may seem like basic requirements, certain integrations are "must-have" features for prospective customers. Vendors that cannot provide must-have integrations typically lose out to competitors that can. Losing deals is one of the most common sources of internal friction around integrations as sales teams press their counterparts in product management to address a deal-killing lack of integrations.

While sales deals are a common driver for integration requests, they aren't always part of calculating the potential return on an integration investment. Even a small uptick in win rates can be an important addition to any analysis. Continuing our example from the previous section, let's assume a baseline sales win rate of 25%. If the addition of these two integration features were to drive a slight one-point increase in win rates, the revenue gains would look like this:

Win rate increase	**Initial**	**Year 1**	**Year 2**	**Year 3**	**Year 4**	**Year 5**
Total customers (no integrations)		300	350	439	555	708
Revenue (no integrations)		$7,500,000	$8,742,000	$10,966,980	$13,883,311	$17,696,968
Total customers (with integrations)		300	353	446	568	727
Revenue (with integrations)		$7,500,000	$8,826,600	$11,156,484	$14,204,419	$18,184,675
Revenue gain		**$ -**	**$84,600**	**$189,504**	**$321,108**	**$487,707**

A small improvement in win rate can have an outsized impact on revenue. To summarize our analysis, we'd see a total expenditure of $512,000 over five years, $520,000 in net new revenue from integration monetization, and $1,319,000 in new revenue from increased customer retention and sales win rates. There is clearly a significant return on investment associated with the integration features.

Now our model, like all models, uses a series of assumptions. Specific circumstances vary, and there will definitely be instances where there isn't a positive return on an integration opportunity. In either case, understanding monetization strategy and following a clear ROI framework will help you better assess opportunities, manage requests, and prioritize the integrations that will provide the most value to your business.

"There is clearly a significant return on investment associated with the integration features."

A Winning Monetization Strategy

Integrations aren't always the difference-maker for any SaaS application, but they are becoming increasingly important. Customers will rapidly lose patience with offerings that can't fully operate as a part of their increasingly complex tech stack. For product teams, adopting an integration monetization mindset isn't just a good business opportunity, it's essential for product success.

The good news is that integrations aren't just a necessary, costly initiative that product teams must grudgingly support. They are opportunities to drive new revenue—either through direct productization, or through increasing win rates and customer retention. With a clear understanding of monetization, you will be positioned to capitalize on the opportunity that integrations provide.

While there are some general guidelines and best practices to consider around integration monetization, there is no one perfect way. The best way to monetize integrations is the one that's aligned with how you go to market, which may vary across integrations. Our above example assumed a similar pricing and deployment model for a company's integrations, but it would be perfectly reasonable to have one integration that's part of your base package, another that's sold as an add-on, and a third that's delivered as a custom service offering. As with all pricing and packaging decisions, integration monetization should optimize for customer segments, how they value integration capabilities, and how your team delivers integrations.

A BEGINNER'S GUIDE

Integration Monetization

+1.415.418.3570 | WEBSITE | BLOG

About this guide

Written by

Peter Zavlaris, Sr. Product Marketing Manager, Tray.io

BEGINNER'S GUIDE TO INTEGRATION MONETIZATION
VERSION 1.0 | 04.04.21