The Frigg Framework: Key Concepts & Considerations

An Overview for Prospective Adopters

Revised August 2022

About the Frigg Framework

Frigg is a software framework used by SaaS product leaders to build, deploy, maintain, and improve integrations faster through an open source model.

About this Overview:

This overview is written for the non-technical SaaS leader. It is a “plain English” explanation that explores why the Frigg Framework exists, and the context in which Frigg is a valuable solution to a common challenge: how to provide more and better integrations faster.

How to read this Overview:

We highly recommend reading this document linearly from start to finish.

Some sections include detailed examples that are indented and italicized.

Section 1A. Business Terms

The ISV world struggles without a common tongue to define integration scenarios and technologies. Before diving into Frigg and the problem it solves, we suggest reviewing the terminology that Left Hook uses:

B2B SaaS: Left Hook lumps a broad range of software products and companies under the "SaaS" shortcut. We intend to include any B2B software product with an API, even if it's not technically hosted or a subscription service.

Integration: This unhelpful word has several overlapping meanings. Left Hook narrowly defines an "integration" to mean "when a software user/customer is connecting two or more user accounts, each on a discrete, external software tool, usually via API."

Universal Integrations: A universal integration is a self-serve productized software tool configured via a user-friendly interface. To qualify as universal, the integration must be installable, configurable, and manageable by a non-technical admin. SaaS companies refer to these in their marketing as "integrations," "apps," "partners," or "connectors." Left Hook adds "universal" to distinguish productized, self-serve integrations from one-off "custom integrations," and/or externalized integrations provided by an IPaaS.

Custom integrations: These are typically built to serve a single organization's requirements and are not "productized." They are usually shipped without any UI, and the tool's developer remains in control of any configuration choices. The business customer is dependent on the developer for ongoing improvement or tweaks.

Despite the recent rise of universal integrations, one-off custom integrations remain a multi-billion dollar business for consultants and professional development organizations that are sometimes referred to as "System Integrators," or SIs.

Integration Platforms or IPaaS: This acronym ("integration platforms as-as-service") includes a range of software tools designed to solve a comparatively narrow range of use cases. By definition these are externalized services, hosted and managed outside of any core SaaS application. A subset of IPaaS providers brand and sell exclusively to line-of-business customers (i.e. Zapier). Many others maintain both a direct-to-lob model as well as an "embedded white-label" model. Read Appendix B for more Left Hook analysis on why SaaS leaders should embrace yet limit their reliance on IPaaS.

Marketplace & Ecosystem: A typical SaaS will aggregate its universal integrations into a "Marketplace," i.e. a gallery or directory. These listings are often exhibited inside the

product's UI, on an external marketing page, or both. Some name this an "App Store," though others have coined more distinctive monikers like SalesForce's "AppExchange."A SaaS company's "Ecosystem" is the sum total of its universal integrations and the business relationships ("technical partners") behind these tools.

(Note that some partner leaders might expand this "ecosystem" definition to include agency, consulting, training, and channel/referral/affiliate partners. Left Hook focuses on technology partnerships powered by APIs, which are usually at the heart of a SaaS company's ecosystem strategy.)

Section 1B. Integration Technology Terms

The following terms dive into more technical aspects of Frigg and its interaction with your core tech stack:

Authentication: Authentication is the control process by which a software user authorizes access to their account and API. Authentication is typically scoped to enforce either individual permissions and/or permissions that affect an entire organization's account (i.e. at the End Customer Admin level, described below). The process commonly involves passwords, API keys, or oauth login systems, among other potential authentication methods.

Configuration: Configuration is our broad term for the range of choices executable by an ECA (see below) through an integration's UI. After years of dependence on developers and expensive code tweaks, non-technical business leaders are now eager to take responsibility for managing integration-related choices.

End Customer Organization (ECO): The single *organization* paying to use a SaaS.

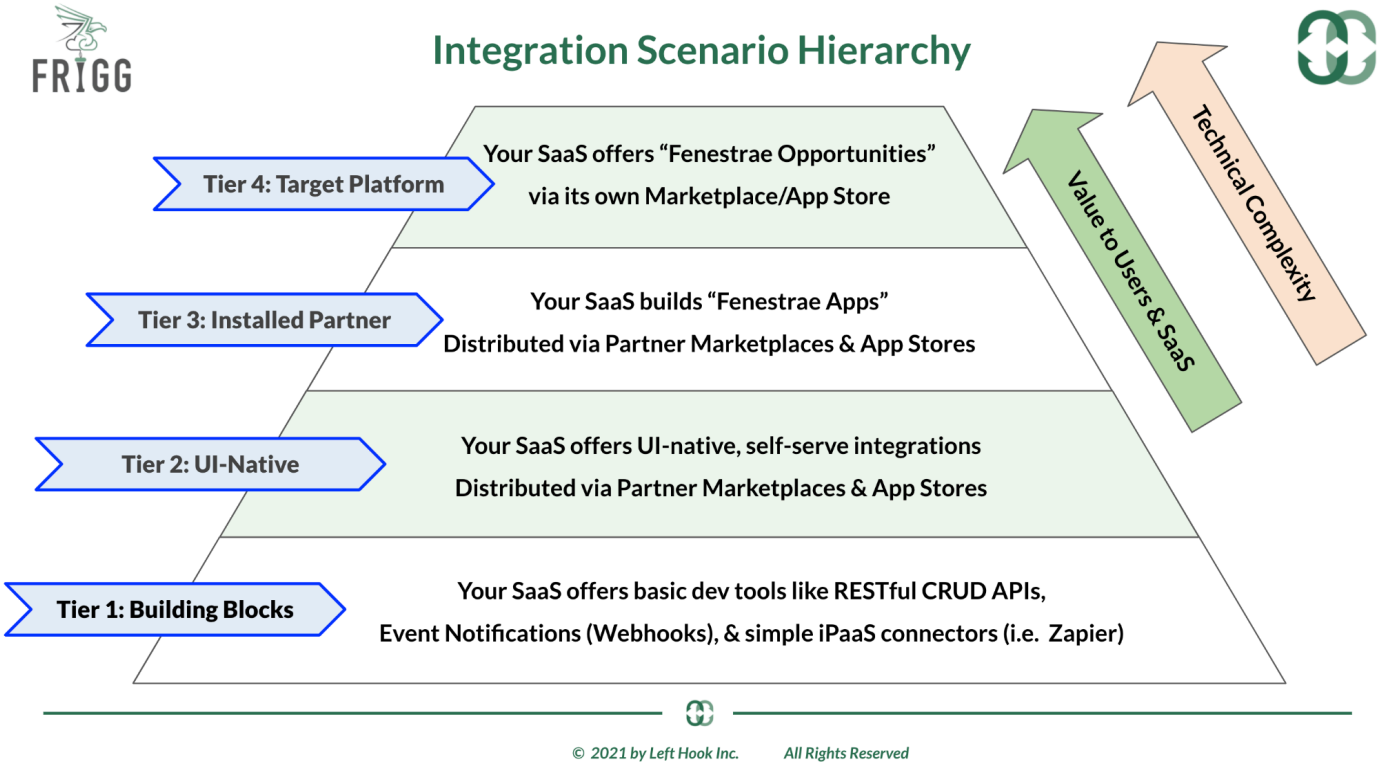
End Customer Admin (ECA): An ECA is an Individual User with elevated account permissions sufficient to authenticate and configure a universal integration. Since any universal integration often requires access to the entire organization's data, an ECA is typically a high-level user like "Account Owner" or "Administrator." In rare situations, an Individual User might be empowered to install a universal integration on their own user account.

Seamless Permissioning: After logging in to a SaaS product, users expect instant and ongoing access to all permitted resources, even if those resources are surreptitiously controlled by multiple and disparate authentication systems. Left Hook strives to help SaaS clients achieve seamless permissioning as a part of any Frigg implementation.

Section 2: Integration Scenario Hierarchy

The integration landscape is complex, confusing, dynamic, and *messy*. Whenever we pitch Frigg, SaaS leaders want to understand exactly how it can accelerate integration development.

While integrations come in many different shapes and sizes, they can be generally classified into four hierarchical tiers. Each is explained in detail below the graph below.



Tier 1: "Building Blocks"

Tier 1 identifies fundamental API features required to build integrations. While seemingly "table stakes" in 2022, a surprising number of SaaS companies still fail to offer a public API for customer use. And by our count, about 85% of all B2B SaaS products still lack a public Zapier app.

Sidenote: Our inclusion of integration platforms (particularly Zapier) in Tier 1 could be confusing, as they necessarily provide a "no-code, self-serve" user experience. However, because their UI is external to the SaaS product's UI and out of its control, we view these more as "connectors" and API extensions, rather than "UI-Native" integrations (described below).

Adopting a third party integration platform like Zapier drives additional cost and requires significant orientation and learning effort from the ECO. In our experience, ECAs generally prefer "UI-Native" integrations (described below).

Tier 2: "UI-Native"

"The next big thing is the one that makes the last big thing usable."

-Blake Ross, Co-creator of Mozilla Firefox

Tier 2 describes when one of the two SaaS products (the "**Ringmaster**" or primary partner) agrees to provide the necessary authentication and configuration screens from behind its product's authentication wall, and thus *within* its own product UI.

The second SaaS product (the "**Wing**" or "Target") engages via API only, and doesn't contribute substantial engineering effort to the partnership. Note that some UI-Native integrations are built without the Wing ever becoming aware of the Ringmaster's accomplishments. This dynamic is typical with giant companies (i.e. Google) that don't invest in proactive partnerships with smaller SaaS partners.

UI-Native Integration: Detailed Example

SaaS Company "A" (CoA) is the Ringmaster. SaaS Company B (CoB) is the "Wing."

Imagine that an ECA for "BizExample1" ("Biz1") logs into their CoA account and clicks into the "Integrations" page to find a gallery of direct/native integrations.

Biz1 clicks on the logo for CoB to initiate the authentication and configuration flow, i.e. a modal window opens and asks Biz1 to enter CoB authentication credentials.

Once authenticated, the ECA makes some configuration decisions. When finished, the ECA returns to CoA's primary UI. The integration is now live and working, without the ECA ever accessing CoB's UI.

Note that some SaaS veterans refer to UI-Native integrations as "direct" and/or "native." These terms are helpful but often confused; "UI-Native" is more precise.

Sophisticated customers often assess the breadth and depth of a prospective ISV's UI-Native library as a signal for overall robustness. A healthy roster of well-designed UI-Native integrations is the gold standard for successful SaaS companies. (We'll further discuss the importance of Tier 2 in a section ahead.)

Tiers 3 & 4: Fenestra "Installed Partners" & "Target Platforms"

As "table stakes" as Tier 1 and Tier 2 are to SaaS success, they are required but not sufficient. The next frontier entices ECOs, confronts SaaS product leaders, and raises the technical bar. Until now, the SaaS world had little language to describe these Tier 3 scenarios. To help SaaS leaders better understand, Left Hook is coining new terminology centered around the unusual word "Fenestra(e)."

Recalling your high school Latin, "Fenestra" (singular) means both "window" and "opportunity." It provides an efficient label to encompass the many integration scenarios already out in the wild that fit the general pattern.

A Fenestra Opportunity Explained:

1. One SaaS product (the "Target Platform" or TP) enables another SaaS product ("Installed Partner" or "IP") to display data and/or features inside the Target Platform's UI. Each enabled location is a "Surface."

2. The Installed Partner builds its "Fenestra App" according to the TP's requirements. When ready, the IP submits the app to the TP for acceptance review.

3. Once approved, the TP distributes and promotes the app via its marketplace; for qualified partners, it co-markets and co-sells as well.

4. ECOs discover the app in the marketplace and self-initiate installation. Authentication and configuration occur in either UI, depending on TP's requirements.

5. TP markets its "Fenestra Opportunities" to other prospective IPs and encourages them to "build to us."

A detailed example might also help your understanding:

> *Consider the integration between FreshBooks (FB) and HubSpot (HS), built by Left Hook. FreshBooks is an invoicing and accounting tool; its typical ECO is a small business like the fictional "Biz1." HubSpot is a CRM tool also used by small businesses.*
>
> *To be more concrete, let's stipulate that Biz1 sells laminate flooring to homeowners. Biz1's team includes a part-time bookkeeper/invoicing manager, a part-time IT admin (the "End Customer Admin"), and six account reps who handle customer relationships (the "Individual Users").*

Biz1 uses HubSpot's CRM as its activity tracker and "system of record" for all homeowner contact information and communication. Its six account reps spend their entire day with HubSpot open on their screens, logging information about each customer call or email inside the HubSpot UI. Rather than jump between several browser windows, Biz1's six reps would prefer to use HubSpot as a "single pane of glass" for their workday.

HubSpot is neither an invoicing nor an accounting tool; Biz1's bookkeeper prefers to use FreshBooks to issue invoices and track payment activity. Biz1's part-time bookkeeper doesn't have enough hours to communicate about the lifecycle of each invoice, and it would be problematic to give the six reps access to Biz1's accounting software.

Yet it's still important for the reps to know when an invoice is sent, viewed, or paid, as a rep with instant situational awareness can accelerate cash flow and make Biz1 "look good" by knowing invoice details.

To solve this challenge, the IT admin searches HubSpot's "App Marketplace." She clicks "Get Integration" from the FreshBooks integration directory and launches a new browser tab, which prompts her to authenticate into both Biz1's FreshBooks and HubSpot accounts.

(Two sidenotes: First, FreshBook's authentication flow is an example of a lesser but workable alternative to "seamless permissioning," should your SaaS need to pursue one. Second, this particular integration also includes a bulk data syncing of eligible HubSpot customers into FreshBooks, and a corresponding sync of FreshBooks clients into HubSpot. This contact syncing saves the bookkeeper from manually entering each new customer profile in FreshBooks.)

Now that the app is enabled, contextual status data about each invoice is viewable inside the HubSpot UI. Alongside each contact record HubSpot provides a right-most column for its partners to display "CRM Extension Cards," a series of small rectangles stacked vertically. One group of CRM cards is branded with FreshBooks' logo and displays status updates (i.e. Draft, Sent, Viewed, Paid) for the five most recent invoices associated with the contact's email address.

The Activity Timeline will also display a log entry for any invoices created or payments made.

Fenestra in the Wild:

Once you grok our language and common themes, you'll begin to recognize the Fenestra pattern as it is practiced in some high-profile SaaS ecosystems. Examples:

Salesforce “Lighting Apps”	**Notion “Blocks”**
Trello “Power-Ups”	**Front “Plugins”**
Airtable “Blocks”	**Coda “Packs”**
Miro “Web Plugins”	**Pipedrive “App Panels”**

After analyzing more than 60+ Fenestra Opportunities, Left Hook has identified two helpful subtypes:

Subtype 1: Feature Hooks
A Target Platform (TP) designs its “Feature Hooks” to allow an Installed Partner to expose its data in a limited and structured way via constrained surfaces, and almost entirely via API. While the TP dictates most display attributes (i.e. location and style), the Installed Partner chooses what data is displayed.

> *In the FreshBook/Hubspot example, both Timeline Events and CRM Cards are Feature Hooks. The Installed Partner (FreshBooks) defined the unique events it wanted inserted into the user’s timeline, and stipulated the specific data.*
>
> *The Installed Partner can also add actionable elements like a button to open a modal or another tab. However, the look/feel of the timeline events are controlled by HubSpot, as are any UI action elements (i.e. no FreshBooks’ blue buttons.)*

Subtype 2: Open Canvas
A growing number of Target Platforms offer an “Open Canvas” in which the Installed Partner can display whatever external URL or HTML it chooses, while still leveraging data from the surrounding record to establish context.

Again HubSpot is a useful example. As a feature of its CRM Extension “cards,” an app can include a button that initiates a modal window. Inside this window’s four walls the partner can render any URL, from a simple iframed page to a true Single Page App. If triggered while viewing a Deal record, deal attributes become parameters to programmatically change the modal’s display.

For example, Stack Global (a Left Hook client) is software that generates customizable “playbooks” for sales reps to follow a predictable, measurable sales process. Because many sales reps live inside the HubSpot CRM all day, Stack wanted to display a sales playbook inside their “single pane of glass.” The modal’s specified view is set by the deal stage field managed in HubSpot.

Stack's app offers not just right-column "CRM cards," but presents a button to trigger an Open Canvas modal. Inside the modal, the Stack app displays relevant best practices and prompts the sales rep to follow each prescribed step of the sales cycle.

Achieving Tier 4 and *Becoming* a Target Platform

Many SaaS leaders dream about their product becoming a *Target Platform* and asking (requiring?) their partners to "build to us." Relatively few companies have achieved this yet; most are unicorns with +$50 million in ARR (or gobs of VC funding). Building an app platform requires high-quality engineers and designers capable of designing bespoke systems to handle partner apps at scale.

To date and to our knowledge, there are no open source tools or frameworks to accelerate this transition. (Yet).

In Left Hook's experience, most SaaS companies aren't actually ready to become a Target Platform. Beyond the technical hurdles, many products just aren't a great fit.

Some ISVs are best suited to being Installed Partners that thrive through marketplace distribution (i.e. Clearbit or Twilio). Other ISVs are targeted at narrow audiences (i.e. bookkeeping or payment systems) that are purposefully walled off from a broader user base.

However, Left Hook believes that more and more SaaS products will choose to become a Target Platform over the next decade, especially those that provide a "single pane of glass" experience to teams needing restricted views into their employer's data (I.E. CRMs and CX platforms.)

If you think your SaaS product has the right stuff to become a Target Platform but can't yet afford the talent to build one, we have good news: Frigg is on its way to becoming that open source framework the industry needs to *accelerate* Fenestra Opportunities.

(Let us know if you're interested in exploring the possibilities together.)

How the Hierarchy Informs your Strategy

The hierarchy provides a yardstick your SaaS can use to measure how your product stacks up against others in its category. Here is Left Hook's general advice about how the hierarchy should inform your own integration strategy:

1. If your product doesn't yet offer a public & RESTful API, focus there first. Custom integration development opportunities will always be relevant to enterprise customers, particularly those in the Fortune 5000 and heavily regulated industries. A good API leads to good universal integrations too.

2. Publish a Zapier app (or ask Left Hook to help). Pursue other IPaaS connectors that match the needs of your customer base.
3. Verify the quality of each “integration” in your competition's library. Get hands-on and determine which ones are a true UI-Native integration and which are a Zap Template in masquerade. This assessment will help you decide what you’re *actually* competing against; their customers eventually recognize the difference with frustration.
4. Plan and build the Top 15 UI-Native integrations for your category. Be prepared to serve as both Ringmaster and the Wing depending on the partner.
5. Identify five Target Platforms that will distribute and expose your product to the right audiences; keep in mind common use cases and the potential for reusable code/UI.

Section 3. Technology & Implementation Process

If you’ve finished Sections 1 and 2 above, you understand the role Frigg might play in accelerating your many integration opportunities, in the larger context of how this technology partnership stuff works.

Section 3 explains the framework’s technology, and answers common questions received from product teams who have evaluated Frigg to date.

Frigg is <u>not</u>:

- A hosted SaaS-like service that bills on a subscription
- An embedded iPaaS (Cyclr, Workato, Pandium, Tray.io, etc.)
- A customer-facing iPaaS (Zapier, Integromat, Celigo, etc)
- Limited to data syncing or workflow integrations

Frigg is a software framework, which means it:

- Is code your SaaS “owns” & controls on your own cloud infrastructure
- Is an open source development tool (i.e. like Wordpress or React.Js)
- Is extensible with custom code, plug-ins, or community additions

- Can be customized by competent developers who work in Frigg's native stack
- Accelerates Tier 2 UI-Native integration development
- Accelerates Tier 3 "Installed Partner" app development
- COMING SOON: Accelerates an ISV's journey to becoming a platform, and asking other partners to "build to us."

Left Hook is:

- Architect and originator of the Frigg Framework
- Frigg's lead developer
- Professional services team to implement, configure, customize, & deploy Frigg
- Expert at +200 APIs, Fenestra dev, and marketplace strategies
- Working on an Integration Opportunity (IO) SaaS Platform, a suite of self-serve tools to help SaaS leaders discover, design, generate, deploy, support, monitor, market, monetize, improve, measure, and analyze how tech partnerships drive SaaS success (coming soon)

Requirements for Adopting Frigg (as of August 2022)

1. Use of the Serverless.com framework
2. Hosting via a compliant cloud provider
3. Use of Node.JS
4. Use of a GIT provider

Strongly Recommended

1. Clear internal process and criteria for selecting technical partners
2. An understanding of your technical partner's users, use cases, and features (i.e. product management)
3. Internal capacity to design and develop your integration frontend

4. Internal capacity to understand Frigg's technical requirements and the long-term responsibility that comes with "owning" Frigg code
5. Use of Mongo, AWS, Redis, Seed.run, and Upstash

Internal Players Key to Successful Adoption

1. Senior Product Manager who understands integrations and your partner's product
2. Senior UI Engineer
3. Back End Engineer
4. Deployment/CI/CD/Infrastructure Lead
5. Partnerships Leader
6. C-level approval, usually CTO

Important Frigg-Specific Terms

Frigg (Goddess): Frigg is Odin's wife and the Norse goddess of marriage and partnerships. She was said to ride a falcon across the skies while weaving the clouds.

Software Framework: From Wikipedia, "In computer programming, a software framework [provides] generic functionality, can be selectively changed by additional user-written code, thus providing application-specific software. It provides a standard way to build and deploy applications and is a universal, reusable software environment…" Examples of other successful open source frameworks include React.js, Django, and Bootstrap.

API Modules: Reusable, shareable code chunks that wrap a particular partner's API to facilitate its interaction with Frigg. While akin to "connectors" in IPaaS-speak, API Modules extend beyond basic RESTful functions. They include templated code that accelerates development of Fenestra Opportunities offered by the partner's target platform. Frigg comes with a library of 25+ API modules that include Salesforce, HubSpot, Stripe, Salesloft, monday.com, Slack, Google Calendar, and many more commonly integrated B2B SaaS products.

Logic Blocks: Reusable, shareable code chunks that determine how Frigg orchestrates specific use cases with data moving in/out via API Modules. Logic built for one set of use cases can be shared and extended.

Integration Manager: In Frigg's architecture, Integration Managers are where use cases are orchestrated using different Logic Blocks.

Base Management API: Each Frigg instance includes an API that manages authentication credentials and stores various configuration options.

Section 4: Working with Left Hook

As of August 2022, Frigg is now a fully released and developable open source tool. Left Hook's professional services are not strictly required to use Frigg'; in fact, we encourage and support any developer willing to give it a try.

However, if your ISV needs external support and development resources, Left Hook is happy to work with you to make Frigg a central part of your complete tech partnership program.

To get started, we offer "Phase 1 Consulting" to address a list of critical scoping questions. A senior consultant will help your team think through the technical nuances and requirements of adopting Frigg. Our team will also perform research and API testing to guide use case design and selection.

The end result is an actionable build plan that identifies "who does what" between Left Hook and your product team, tasks with timelines, and a budget for Left Hook's further involvement.

Once your implementation is planned, Left Hook can provide a range of development services necessary to configure, deploy, and customize Frigg to your specific requirements. We can do all the work, some of the work, or be a guide/consultant to your product team.

Scoping Questions

Even our most sophisticated clients hesitate to answer some of these questions without first engaging Left Hook to clarify and unpack them. While we attempt to provide background information and context upfront and in writing, we expect your product/engineering team will need multiple conversations before arriving at well-considered decisions.

Question Set 1: Potential Partners, Use Cases, and API Functionality

1. *A year from now looking back, what does a successful Frigg implementation look like to internal stakeholders?*
2. *Which potential partners are you targeting and why?*
3. *If you knew a potential partner would refuse any marketing/sales cooperation, would you still pay to integrate with them?*
4. *What are the specific use cases and workflows that your common ECAs want in*

a universal integration?

5. *Are any use cases applicable to multiple integrations/partners, i.e. "syncing people/contacts from several CRMs."*
6. *Does your existing API support all desired use cases? If not, is your API product team ready to make improvements soon?*

Question Set 2-User Interface (UI)

7. *Where is the ECA going to authenticate and configure the integration? Inside your product's UI? From your external integration directory page? Or ?*
8. *Who is responsible for building any required Auth/Config UI?*
9. *How will the UI leverage Frigg's Management API?*
10. *How will Frigg identify your ECOs, ECAs, or Individual Users?*
11. *How will Frigg authenticate ongoing API requests?*

Question Set 3-Infrastructure and CI/CD

12. *Which compliant cloud provider will you use?*
13. *Which default CI/CD tools/services do you prefer?*
14. *Who will manage deployments, and what process will we follow together?*

FAQs

Can Left Hook develop your auth/config UI, too?

Clients often ask if Left Hook can perform the required auth/config UI development. Our answer is "maybe, but we don't recommend it." Here's why:

1. Seamless Permissioning should be your goal. To achieve it, an internal frontend leader needs to get deeply involved, the earlier the better.
2. ECAs overwhelmingly prefer integrations launched from within a native, trusted UI. While Left Hook will work closely with your team, co-developing UI elements inside an existing stack is very challenging. In the long run, your users are best served by a single unified UI "owned" by your product team.

3. Left Hook's front end development skills are limited to React.JS. According to the most recent Stack Overflow Developer Survey, 60% of developers use a front end technology other than React.
4. Left Hook is not a graphic design or user experience shop. Our backend/ data/API skills will always be our primary focus.

However, Left Hook does provide UI development under limited conditions:

- Using Frigg's library of React.JS components is a requirement
- Client must provide a UI design from a graphic/UX designer
- Elements will likely render in modals or tabs, and not your in-page UI

How much internal effort is required to develop an Auth/Config UI?

There's no quick answer to this question. One driving factor will be your existing UI. Is there an existing "Integration Page" with an established auth/config pattern? Is this UI ready to leverage Frigg? Are your front end developers skilled at working with a RESTful API?

Another critical factor is the complexity of the use cases, as more configuration options raise the execution bar.

All told, answering this question requires an extensive discussion between Left Hook and your product team undertaken in the "Phase 1 Consulting" previously described.

Can Left Hook help your SaaS become an "Installed Partner?"

Absolutely! Building Fenestra Apps is one of our core competencies. We can build Installed Apps for dozens of marketplaces, including all the obvious ones (HubSpot Marketplace, Salesforce AppExchange, Google Workspace Marketplace) and many more.

Given the previous UI-related FAQs, you might be wondering, "How is it that Left Hook can build out Fenestra apps but not our Auth/Config UI?" The answer lies in where an Open Canvas UI element will exist (often outside your core UI and stack), and how it can be built and served externally (likely using React.JS). These two conditions allow for Left Hook's deep involvement in this particular UI development.

Can Frigg power other integration scenarios besides Fenestra? Yes!

Fenestra Opportunities are novel, exciting, and highly valuable investments. But in Left Hook's recent experience, more traditional Tier 2 data syncing and workflow

automations remain highly valued by SaaS customers. This is why Frigg enables use cases across all of our hierarchy tiers.

Frigg outcompetes embedded IPaaS on total cost of ownership and long-term control. While its Fenestra capabilities help distinguish it as well, Frigg performs and competes well across the entire integration scenario hierarchy.

What long-term costs and responsibilities come with adopting Frigg?

The good news about Frigg is that your company will end up "owning" the code. The bad news is that your company will end up "owning" the code. Frigg is not turn-key hosted software; your product team needs to be deeply involved in its implementation, operation, and ongoing product management.

However, Frigg's open source system will encourage qualified service providers to provide outsourced development. Frigg's developer experience (documentation etc), user community, and service provider network will support Frigg adopters over the long term- with or without Left Hook's long-term involvement.

Is Frigg right for your SaaS?

Ask your product team if Frigg is right for you- after they've had a chance to read this overview and our documentation.

Appendix A: “Integrations: A Brief History of User Expectations”

Back in the earliest days of B2B SaaS (circa 1995-2005), customers seeking integration or automation between systems were expected to pay for API access AND hire expensive custom developers, on top of their subscription fees. This “custom development expectation” was status quo until well into the 2010s, and persists in some verticals even today.

The paradigm began to change in 2005 thanks to Salesforce and its AppExchange. Over the past 15 years, a huge audience of B2B customers learned to love no-code, self-serve, “app-ified” alternatives. From 20,000 installs in 2005 to 9 million in 2021, Salesforce convinced a massive number of B2B software buyers that integrations should be universal, easy to use, and ubiquitous.

The AppExchange's popularity compelled many CRM competitors to build their own universal integration marketplaces. Many savvy SaaS leaders begin to recognize universal integrations as a competitive advantage, and copied Salesforce’s playbook.

Within a decade after its launch, hundreds of SaaS companies offered their own “App Store '' or “Integration Marketplace” filled with thousands of self-serve, no-code tools. B2B buyers began asking about the size and depth of these libraries during the purchasing cycle, with many deals tipping in favor of the SaaS product that appeared to be better integrated.

This AppExchange-inspired arms race was a key driver for why SaaS “ate the world” in the 2010s. The more interoperable and easily integrated a SaaS became, the quicker it could outcompete older stovepipe systems.

Appendix B: iPaaS vs UI-Native

Sometimes these services are known by their acronym “IPaaS.” We prefer to call them “integration platforms” to avoid confusing acronyms.

To illustrate how an integration platform works, let’s recall one of the examples used in Section 2, Tier 2.

> *Imagine that both CoA and CoB are enabled as “public apps” on Zapier. The ECA for Biz1 logs into Zapier (and not either CoA or CoB) to begin the process of building a “Zap” i.e. a single workflow instance. After authenticating their CoA account, they choose to send data from CoA to CoB whenever a specific event is triggered by any/all of CoA’s Individual Users inside the CoA UI, i.e. “whenever a Contact’s phone number is added or changed.” The Zapier UI provides several configuration options as dropdowns and selectable fields.*

Many SaaS leaders wonder: is it really “as easy as Zapier” to outsource their integration headaches? What’s the catch?

SaaS leaders enjoy externalizing the cost and hassle of building a self-serve UI and infrastructure. This costshifting is a rational and efficient choice. But as there’s no free lunch, the platform expects something in return. For example, Zapier compels most of its SaaS partners to embed its heavily branded Zap Builder inside their UI. Customers must register for and pay for a separate Zapier subscription.

In Left Hook’s experience, a *minority* of B2B SaaS customers prefer using external integration platforms. They find that external integration platforms offer flexibility, speed, and the ability to to chain workflows together across multiple steps and tools.

However, the *majority* prefer and *expect* “UI-Native” integrations that are included in their subscriptions and configurable from within a familiar and trusted UI.

Many customers resent paying for an additional integration platform subscription. When compared to a simple, well-designed UI-Native integration, even Zapier feels like a “yet another software” burden for the user to learn and manage.

Instead, these same customers would rather pay *more* for their software subscriptions to get the UI-Native integrations they want. (Our friends at Profitwell analyzed integration “willingness to pay” and concluded as much.)

All things considered, Left Hook’s experience has convinced us that as more “UI-Native” integrations are promoted via in-app directories and thriving marketplaces, the more customers will prefer them over an external integration platform. Our

considerable investment in Frigg reflects our conviction that customers want what it offers.

Frigg vs Embedded White Label IPaaS

A growing number of technology vendors sell "embedded, white label IPaaS." These solutions are usage-based subscription tools that provide cloud infrastructure, reusable logic, and a shared pre-built connector library.

Left Hook is familiar with multiple embedded & white label IPaaS vendors; we've built dozens of the connectors, which gives us unusual insight into their features and business models. Several embedded iPaaS vendors make a compelling case, given their large pre-built connector libraries and "pay as you go" incremental cost.

However, we think all embedded IPaaS competes poorly with the Frigg approach, for these many reasons:

	Typical Embedded iPaaS	Frigg
Business Model & Pricing	Monthly fixed fee plus usage-based transaction fees; multi-year contracts and no clear transition off-ramp	Framework is free. Relatively high upfront implementation costs, but long-term "total cost of ownership" is much lower
Integrations: Breadth and Depth	Broad range of "shallow" connectors capable of lower and mid-tier hierarchy use cases; new connectors and use cases usually built by the embedded iPaaS provider	Small number of API Modules to start, but each is built to enable the full hierarchy of integration use cases; Left Hook is an API Module factory at our core, and can build fast
UX Capabilities	Implementation typically requires significant frontend dev services not provided by the iPaaS (though some offer limited "out of the box" packaged solutions)	Total flexibility, from 100% handled by your team, to 100% handled by Left Hook's team; library of React components speeds development; we support "UX surface" opportunities that embedded iPaaS can't provide
Self-Manageability	Some providers offer workflow templating and enablement implemented by the SaaS owner; core code and infrastructure mostly off-limits to customer's internal devs; some allow for self-dev of connectors; many	Frigg allows an ISV to manage, customize, or extend as it wishes. All code is accessible to your devs and written in common frameworks (React, Node.js)

	providers require time-expensive DIY implementation and mastering of their platform	
Integration Scenarios Enabled	Tier 2: UI-Native	Tier 2: UI-Native Tier 3: Fenestra Apps Tier 4: Become a Target Platform

###

