Elion

Buyer Review Transcript

Product Reviewed	Tray.io
Reviewer Tech Stack	
Other Products Considered	Workato, Awell, Dock
Date	August 22, 2023
Expert Role	Operations Consultant
Organization Type	Digital Health Consultant
Payment Model	Fee for Service, Cash Pay, SaaS

We're chatting about Tray.io. Before we jump into that, could you give a brief overview of your company and your role?

I run a consulting business where I offer advice and do project-based work. I specialize in helping early-stage digital health companies, particularly those aiming to establish virtual clinics or telehealth services. This includes everything from developing go-to-market strategies and selecting the right products to managing clinical operations and building scalable virtual clinics.

How long have you been using Tray?

Three years.

What caused you to look into purchasing Tray?

The company I was working with at the time didn't have an engineering team. It became clear that our manual processes weren't cutting it anymore, especially considering the scale we were operating at. We needed to move data between different systems, and we needed better reporting. We had all these needs, but we didn't have the engineering talent or the manpower to tackle them. Since we were dealing with health data, we couldn't use Zapier. I can't quite remember how I first heard about Tray, but I knew there were automation platforms out there. I probably started by searching for something like "HIPAA-compliant Zapier," and that's when I came across Tray. Initially, I was shocked by the price. As a small startup, this was one of the most expensive software purchases we were making, and I wasn't sure if it was responsible, but there were so many problems it could solve for us.

What problems does Tray solve for you?

I'll give you examples of five different workflows that I've built using Tray.

The first one is set up so that every time someone fills out a lead form, it automatically adds the information to our CRM system. It also sends a notification to a specific channel in Slack, so everyone on the team knows when a new lead comes in. Then it uses round-robin logic to assign the lead to a team member, who then gets a ping in Slack to let them know they need to deal with and triage the request.

Another example is a daily reminder for patients to check their blood pressure. It's integrated with Twilio, so every morning it pulls in all the patient contact information from our electronic health records (EHR) system. It then checks if the patient has given consent to receive text messages, based on a specific field in the EHR, and then goes through the list and sends each patient a customized message with their name, like "Good morning, Karen. Please remember to check your blood pressure today."

You can also use it for alerting purposes. For instance, if a value from a remote patient monitoring (RPM) device comes in and is significantly outside the normal range, you can automatically send a message to the patient that says, "Please check again. If your heart rate (or blood glucose) is still above or below this range, you should call 911." This type of alerting is quite useful.

Another way to utilize it is as an event bus, which involves receiving and deciding what to do with events from various parts of the organization. Currently, I'm working with a client where we are ingesting events from their custom-built application, CRM, EHR, and phone calling system, and taking action accordingly. For example, when a new signed chart note arrives from the EHR, it uses a complex set of logic to determine if certain conditions are marked on the chart note, and if so, we update the corresponding data in the CRM, send a text message, create a package, and order a device from our RPM vendor (Smart Meter) based on that package. As Smart Meter fulfills the order and provides us with the tracking number, it ingests those events and sends an email to the patient informing them that their order has shipped with the tracking number and a contact button.

And one of the main areas where we utilized it was for analytics. We needed to bring the data into Google Sheets and structure it effectively, since we didn't want to invest in a more robust ELT/Snowflake/Looker data pipeline. So we relied on Tray to handle the ingestion of events and input the data into Google Sheets. Tray would then automatically process and convert that data into charts that we could use to monitor the metrics that were important to us.

What does the actual product look like, and how do you use it?

Much like Zapier, Tray has the concept of workflows, which are referred to as zaps in Zapier. These workflows are based on triggers, and there are various types of triggers available. One

type of trigger is a manual trigger, where the workflow runs when you click a button. There are also app-based triggers. For instance, you can set up a trigger with Google Sheets to run a workflow whenever a new row is added to a Google sheet, or when a new contact is added to ActiveCampaign. The trigger that I use most frequently is a webhook-based trigger, which activates when data is posted to a webhook. An example would be that our custom app sends posts to a webhook when a patient logs into the app. Additionally, Tray provides the concept of callable workflows, which can be triggered by other workflows, and I find this feature very useful.

Tray groups these events and triggers specific workflows based on how they are categorized. In the Tray product, they visually represent this setup with small boxes called connectors, connected by lines. Let's say we have a trigger box. When you click on it, a configuration menu pops up. In this menu, you can set things like the authorization header and which authorization to use. (We manage authorizations separately in another part of the product.) So, in this example, let's say you want to use a Slack connector. You would first choose the Slack account you want to use from the available authorizations. Then you would input the channel ID, the user to tag, and the message body. You can even create a channel programmatically if you need to. And each connector has various configuration options to tailor it to your needs.

They also have a lot of default connectors, like code blocks or JSON parsers. In cases where a default connector is unavailable, they have an HTTP connector that allows for API requests. I find myself using the HTTP connector often, especially when working with small healthcare software companies that have not been integrated with yet. This applies to Healthie as well, although Healthie is not considered small anymore. Healthie's APIs are usually comprehensive, so I frequently use Tray's GraphQL connector to make raw API requests. If you need to perform more complex tasks beyond the capabilities of Tray's default connectors, you can write custom code using a code block in your workflow. I'm not an engineer, but I can write some basic Python, so this option allows for more advanced functionality.

As you were thinking about these solutions, what were the key requirements that you were evaluating these products against?

The biggest competitor I've looked at is a company called Workato, which offers a very similar product.

The first thing I looked at was the difference in the number of integrations available. The main idea behind these products is to save engineers' time. Just because I can write Python or perform raw API requests doesn't mean that every operations person will be capable of doing so. Therefore, the number of default connectors offered by the platform is quite important.

Price is obviously another factor to consider. We discovered that Tray scaled a lot better than Workato in terms of cost, especially with their event-based pricing model.

Another aspect we considered is ease of use and error handling. One of the things I really like about Tray is how easy it is to go back and replay something within the UI if there's a failure.

Let's say an engineer writes a bug in your code and something goes wrong. It can be complicated to fix and replay, but with Tray, you have a log of all the automation runs that you can filter by failed runs. You can make changes to the automation, and then, for example, replay the webhook trigger as if it was the first time. This makes it easy to debug and ensure that it performs exactly as desired. Additionally, every workflow can post to an alerting workflow when it fails. For example, I have an automation alerts channel in Slack. Every time an alert fails, it posts a message that includes details like the workflow that failed, the step where it failed, the link to the log for that workflow, and the alert message. So Tray handles debugging and replaying of missed or misconfigured workflows really well.

I've checked out some other healthcare-specific platforms, like Awell, and they claim to do a lot of this. And Dock also handles workflow tasks, but I haven't found any healthcare-specific options that can really compete with what Tray offers.

Have you looked deeply into Awell's capabilities?

I sat down with Awell and explained to them the tasks I wanted to do. They quickly realized that their product wasn't suitable for my needs, so we didn't get into a deeper discussion. I had prepared a list of automations I wanted to create, but they made it clear that their product wasn't designed for that purpose.

So the main competitor you reviewed was Workato?

Yeah, I think Workato would have been just as good as Tray. But the pricing was better with Tray, and at the time I was comparing them, I was more familiar with Tray, so that seemed like a good enough reason to choose Tray. But I really liked Workato's product. Their representatives were great too. I think they would have been a good company to go with. I just have less firsthand experience with them, except for the demos.

What other differences did you note between Tray and Workato?

I would say the primary factor was the price. Personally, I found Tray a bit easier to use. However, one weakness of Tray is that they don't have as many built-in integrations compared to Workato. So, when selecting a vendor like this, it's important to think broadly about the software you might actually use and then check the integration lists. Also, I want to give a word of caution – just because the website lists a company they integrate with, it doesn't necessarily mean they have a default connector. Sometimes they create those pages when they can use their default HTTP connector to call the company's API. So, make sure to dive into the details to understand what the supported connectors actually do.

Another thing I'd like to mention is that the more connectors a platform has, the harder it becomes to keep them updated with new features. Take ActiveCampaign, for example. Tray has

a built-in connector for ActiveCampaign. It allows you to write to custom fields for contacts, but not for deals. So sometimes they don't have the exact functionality you expect. In such cases, I can use an API request to achieve the same outcome, but it's more complicated and prone to errors compared to having that functionality built into the connector.

Regarding weaknesses of Tray, I think the UI is a bit clunky. It feels like it freezes sometimes or is not as responsive as you'd want it to be. So the UI is not ideal, but it has never restricted me from achieving what I wanted to do with Tray.

Another thing you'll want to look at: onboarding and training hours, as well as any professional services included in the contract. With Tray, for example, you have access to 10 hours with a solutions engineer within the first six months. I highly recommend utilizing those hours. I sometimes built workflows in a very complicated way, but the solutions engineer showed me a simpler approach using just one connector, eliminating the need for the 10 different connectors I had built. So when reviewing the contract, make sure to take into account the number of onboarding and training hours you'll receive.

You said some of the integrations aren't up to date. Are they responsive to feedback there?

Yes, though I've always relied on using API requests. I reported the issues to the relevant people, and their response was, "Okay, we'll add it to our list." However, I don't have an idea of the length of that list or how likely it is for them to address those issues, so I don't really have much visibility into that process.

How did you find the sales process with Tray?

Every time I've worked with Tray, I've had excellent sales reps who were highly responsive. The sales cycle for me has become pretty short because I just present the deal I know I can secure. I simply ask them to approach their director of finance for approval, and then we aim to sign the contract within the week.The sales team at Tray was really accommodating and made an effort to work with me.

How did you find the onboarding and setup?

I tend to be the kind of person who dives in and figures things out on my own, disregarding the smoother, more traditional software onboarding processes. So I might not be the best person to ask about their very traditional onboarding methods. However, I can say that they have been responsive to my requests. The people I have worked with are knowledgeable and helpful. They definitely have an onboarding process in place, and it is what I would use if I were more disciplined.

What are some other use cases that Tray offers?

Tray is an incredibly flexible tool that allows me to do pretty much anything I want. The only limitations I've come across are mainly related to writing HTTP requests. As long as you're comfortable with that, there's very little you can't achieve. Some of the default connectors have their quirks, though.

As an example, say you have a record pushed from a CRM to Tray, and you want to create a row in a Google Sheet where all the information is posted in the respective columns. Well, that proved to be a bit tricky because not all fields were present, and the order in which they came from Freshdesk or Freshsales was inconsistent. So I had to do some matching based on the field names. The thing is, you can't use a default connector in Tray to take the data and put it in the right columns of the Google Sheet, or create a new column if needed. So I had to resort to writing a script using Google Apps Script and setting it up within the spreadsheet to handle the parsing.

Using loops and lists can be challenging at times, so I highly recommend reaching out to their solutions engineers and implementation support for assistance. This is especially important if you're not an engineer, and it's likely most of your users won't be engineers.

However, I will say that our engineering team has also found great value in using Tray. To provide some context, one of the key benefits of using a solution like Tray is that you don't block operations on engineering, or you need very minimal support from engineering.

And it also allows engineers to automate various processes much more quickly than they otherwise would. For example, Tray makes it easy to transfer items between different parts of a product. It can be used to run repetitive tasks, even for engineers, making it easier to maintain.

One of its major benefits is scalability. Tray automatically adjusts its resources based on the number of tasks being executed, including memory and compute, sparing users from having to handle these details when building their own applications. Even engineers have found it highly valuable, as they tend to have a better instinct for utilizing connectors, lists, and loops than operations staff. However, fully maximizing the potential of Tray requires a certain technical aptitude, which can be a learning curve for individuals who come from non-technical backgrounds.

I would also say the documentation is actually quite good. It provides explanations for every connector, and I found it to be up to date and helpful. However, it lacks information on more complex concepts. This is where solutions engineers can come in and offer their assistance.

In terms of features and functionality, can you highlight some strengths and weaknesses?

As far as strengths, Tray does exactly what you configure it to do. Unlike other platforms that sometimes fail to perform as expected or encounter issues, I've never experienced any problems with Tray. It simply executes the tasks you assign to it. Any bugs that arise are usually due to user error, such as forgetting to re-enable the automation after replacing the trigger. There is some complexity involved that may not be immediately apparent, so it's probably best to set up event hub workflows and then utilize callable workflows. This way, you can avoid tampering with the event hub workflows frequently and reduce the chances of accidentally disabling them.

One major problem I faced was hitting API limits. For example, if you send around 1,000 requests to one of the Tray webhooks, which gathers all the requests, and then you want to store them in Google Sheets, you'll quickly reach the Google Sheet API limit. And in an individual workflow, you can't add a delay. If you add a one-second delay, it applies to all 1,000 requests at once, causing them all to wait one second and process simultaneously. In my initial implementation with Tray, I addressed this issue by setting up an SQS queue for most scenarios where an API limit was expected. I pushed events to the SQS queue and then called a workflow to retrieve items from the queue, process them in batches, and finally delete them. I haven't used it yet, but I'm excited that Tray now offers some built-in queuing functionality, so it seems like Tray is closing those gaps in their product.

Within the context of healthcare specifically, who do you think this product is a good fit for? And if they're not currently using something like Tray, how do you find they're typically solving these problems?

That's a great question. I believe that most tech-enabled, early stage companies would greatly benefit from using Tray. It's a cross-functional tool. Although my background is in operations, I have seen it being used by our care operations team, business operations team, product team, and engineers. Everyone has found some real value in it.

I think a product like Tray is especially important for companies that either don't have an engineering team or have an engineering team that needs to focus on the patient-facing experience. I've worked with companies where a significant amount of time was spent on figuring out how to split engineers' time between patient-facing work and internal tooling. But Tray has essentially eliminated the need for engineers to balance that split, at least until much later.

In terms of cost, I know the product can seem expensive at first glance. But considering the time it saves and the cost of engineers' salaries, I guarantee that every company has recouped its investment within four months.

What are companies that don't use a product like Tray currently doing? Well, they're either having people do these tasks manually, or they're having engineers develop internal tools. In some cases, they simply aren't doing these tasks at all. My main selling point is that Tray can help you accomplish things that were previously unattainable. And it can do so in a much

shorter time frame. You can perform analytics and engage with patients in ways that were previously out of reach. And the best part is, setting it up doesn't require an engineering background. Operations folks can handle it. This saves a tremendous amount of time that is often wasted on manually moving and ordering data. So the short answer is, there are three possibilities: they aren't doing it, they have engineers building it, or they're doing it manually through brute force.

Have you run into any scalability issues with Tray?

These products are used by huge companies on a massive scale, something that the majority of startups can only dream of. And these companies are finding success with them. So I would say that, unless you're a really big public company, you shouldn't be concerned about reaching that level of scale.

How have you found their support in general?

Good. They're responsive. They're knowledgeable. Personally, I don't use their support often, as I prefer figuring things out on my own, but when I have reached out to them, they have been pretty helpful. However, they do have some of those annoying aspects that come with being a big company. For example, if you need assistance with building a workflow, it would be great if you could just have a quick five-minute conversation with someone. But instead, you have to submit a ticket, and they categorize the severity of the issue for you, regardless of how severe you think it is. So their traditional support is what you would expect from a large company catering to enterprises – a 24-hour response time that you can rely on.

The most successful experiences I've had with them were when I built a relationship with my customer success representative. They can really enable things and provide more in-depth support when it's urgent. And Tray also offers professional service hours for a fee. After the initial six-month onboarding period, if you need some additional assistance, you can pay a couple hundred dollars and have someone help you.

Are there any growth areas that you see for Tray?

Yes, I think the UI is a bit wonky. It would be great if they could do a better job of keeping the connectors and integrations updated and provide more transparency about their roadmap and what might be missing from the documentation. It would be helpful to know if certain features are not yet supported, instead of having to search around to figure it out. Also, the searching and sorting of logs could use a better filter. Plus, when it comes to replayability, you can retry a failed step but not a successful one, so more flexibility in that area would be nice. One of the biggest issues for me was queueing, which they've apparently improved, although I haven't had a chance to test it yet.

As for organizing workflows, they've tried different things to organize a large number of workflows and how they interrelate. However, they don't give much clear guidance on the best way to organize and split workflows. It would be helpful if they could provide more direction on the most effective way to do that.

Any advice for buyers who are looking at selecting this type of product?

Negotiate hard on pricing. You can often secure a much better deal than what is initially offered.

If you don't have people who can readily make API calls or if you don't have engineers available to support your operations team, then it's crucial to request a demo account and thoroughly test whether you can achieve everything you desire using their default connectors.