finch

Build vs. Buy

Leveraging Employment Data via HRIS and Payroll Integrations

Whitepaper

2022

Introduction

If you're reading this, you already know that data connectivity has the power to transform every industry in the world—and few businesses are in the dark when it comes to understanding the importance of breaking down the data silos currently impeding their productivity and innovation. In turn, B2B applications are hurrying to bridge the chasms between the systems in their customers' tech stack. Their goal? To create superior user experiences and unlock automation and insights that set them apart from their competitors.

Increasingly, the basis of those bridges is API integrations, and one of the most important datasets to access is housed in employment systems like HRIS and payroll. Determining how to approach employment system integrations is a critical decision for any business, and different business scenarios dictate different considerations as you explore your integration options. Maybe:

- You are starting from ground zero and need to decide whether to buy or build in the first place.
- You already built a few key integrations in-house and are trying to determine if outsourcing remaining integrations and their maintenance would be a more prudent next move.
- You already have an adequate data sync solution, fueled by a flat-file feed and hodgepodge of APIs, that meets your basic needs and are wondering why you should entertain the idea of outsourcing integrations at all. Or perhaps you even recognize that digital transformation is ultimately necessary but wonder if it's an initiative you can keep internal.

In this guide, we explore all of these considerations and more to help you arrive at the best integration approach for your B2B application.

The Employment Data Opportunity

Despite being key to business operations, the employment systems (i.e., HR information systems and payroll platforms) employers use every day to manage their workforce are one of the last frontiers of the integrations race. These systems serve as sources of truth for all employee records, and unlocking the trove of data they hold is crucial to countless use cases across the B2B landscape.

Here are just some of the ways innovative applications are putting HRIS and payroll integrations to work:

- **Retirement plan/401(k) providers** automate participant onboarding and offboarding, deductions and contributions management, and recordkeeping.[1]
- **Health benefits providers** circumvent manual plan enrollment, eliminate CSV file uploads from their workflows, seamlessly change payroll deductions, and manage compliance requirements.[2]
- **Employee engagement solutions** streamline employee onboarding, map out organization charts to identify team members, track team retention, and pre-populate contact information to send recognition gifts to employees on their work anniversaries and birthdays.
- **Training and learning development platforms** automate account openings, import critical role hierarchy and job data, track training progress by employee, monitor retention, and surface important career development insights.[3]

1 https://blog.tryfinch.com/how-to-build-a-best-in-class-retirement-benefits-platform-with-finch/

2 https://blog.tryfinch.com/customer-story-lane-health/

3 https://blog.tryfinch.com/customer-story-trainual-the-scalable-playbook-for-smbs/

- **Financial planning and analysis providers** instantly integrate data from employment systems with other key business software (billing, ERP, CRM) to provide valuable forecasting tools for department costs and headcount planning.[4]
- **R&D tax credit platforms** retrieve the historical payroll information necessary to qualify and enroll businesses in critical federal and state tax break programs.[5]
- **Security compliance platforms** instantly access workforce census records to track employees' background checks and training requirements.[6]
- **Sustainability solutions** leverage employee and workplace location data to populate emissions calculations and quantify customers' carbon footprint.[7]
- **Insurance products** improve underwriting and unlock pay-as-you-go plans, utilizing up-to-date census and payroll data to actively assess risk and adjust premiums.
- **Commercial lenders** cross-reference organizational payroll numbers with data from across the employer business operations stack to augment underwriting.

While the exact functions employment system integrations enable differ from provider to provider, the benefits are consistent across the board: faster onboarding, automated workflows, superior user experiences, real-time insights, and incalculable time savings.

4 https://blog.tryfinch.com/mosaic-and-finch-a-strategic-partnership-powering-modern-business-finance/

5 https://blog.tryfinch.com/mainstreet-and-finch-a-match-made-in-tax-credit-heaven/

6 https://blog.tryfinch.com/secureframe-and-finch-scaling-up-security-compliance/

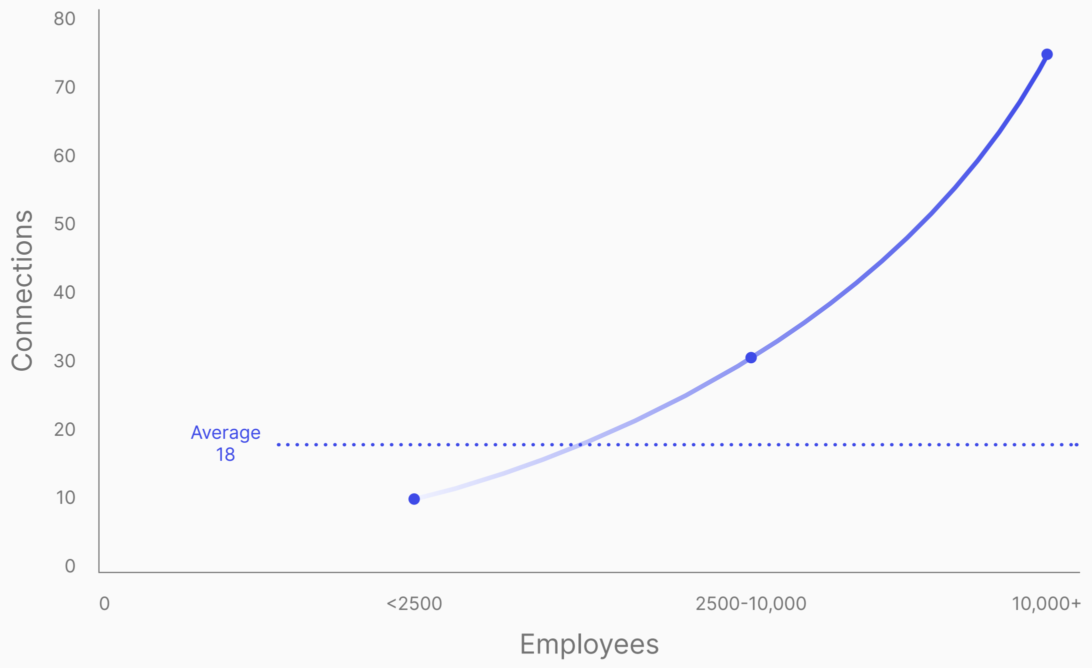
7 https://blog.tryfinch.com/green-places-customer-story/

A Competitive Imperative

These days, data connectivity isn't just a nice-to-have—it's a must-offer.

The reason is simple. Employers' tech stacks are increasingly unbundled and diverse, and the software industries that cater to them are increasingly fragmented. In fact, employers can have dozens of applications[8] connected to their core HR and payroll systems.

Average Data Connections to HR System



Source: Sierra-Cedar's 2018-2019 HR Systems Survey

8 https://www.shrm.org/resourcesandtools/hr-topics/technology/pages/why-data-sharing-remains-challenge-hr.aspx

As thousands of specialized, next-generation B2B applications proliferate the market, employers have the freedom and flexibility to configure highly tailored stacks at a fraction of the cost they used to pay monolithic vendors. With those advantages come new challenges, including a meteoric rise in data silos that threaten to impede productivity if not solved for.

Now, employers expect their chosen solutions to speak to one another, and B2B applications are heeding their call—so much so that you'll be hard pressed to find an employer willing to submit to drawn-out onboarding flows or data entry requirements when there are faster, more automated options to choose from.

For B2B applications, an equally pressing issue is one of differentiation. In a crowded market, employers have choice, and choice can lead to churn. To drive customer loyalty and build stickier relationships, B2B applications are rushing to build the features, products, and services that secure their permanent spot in customers' day-to-day operations. **Fueling those innovations is comprehensive, real-time data from across the employer tech stack—including employment systems like payroll and HRIS.**

Comprehensive and *real-time* are operative words. The innovations that set B2B applications apart aren't powered by static data. They require highly granular, high-fidelity data that comes from live data feeds. But with almost 6,000 employment systems on the market, accessing that data is no small feat. The gold standard—API integration—is a complicated undertaking that only grows more complex with every data source you add.

Where Are You in Your Integration Strategy?

The answer to navigating employment system integrations is a sound strategy, which you may already be in the process of implementing. Based on the stage of your company, how you're using integrations, your priorities in terms of where you want to allocate your resources, and the talent and capital at your disposal, you likely fall into one of three categories of integration implementation—each of which face different challenges and have specific needs when it comes to accelerating growth:

1. Launch

You are launching a new product or feature that leverages employment system integrations but don't have any of those integrations in place. If you're in this phase, getting to market quickly is paramount: You need to generate revenue and traction as fast as possible, so you can create a product feedback loop.

Your challenges:

- **Chicken and the egg**
  Acquiring customers requires employment system integrations, but to get those integrations, you need customers.
- **Divided attention**
  Your engineering team is focused on driving value and differentiation through core product development. Adding integrations to the roadmap slows everything down.
- **An obstacle to growth**
  Larger companies expect you to work with their employment systems; if you don't, they may see your product as too underdeveloped to take a chance on.

2. Scale

You have a few integrations that you built in-house, or you're working from a semi-automated solution that's been keeping you viable. But now, you're getting larger, and you're running into more complex data requirements and new systems that customers are requesting compatibility with. You need to expand your connectivity without investing more talent resources.

Your challenges:

- **Lost business**
  Potential customers expect a seamless experience and are opting to not complete your onboarding process because you're not compatible with their employment system.
- **Delayed implementation**
  Due to the volume and complexities of edge cases and unsupported systems, you're using an operations team to help create a data sync, which can take weeks to confirm.
- **Support constraints**
  Your team is flooded with employment-data questions, and you have to dedicate hours per customer to make sure they have a smooth onboarding experience.
- **Ongoing maintenance**
  You spend excessive time and effort keeping existing integrations running smoothly and troubleshooting the inevitable problems that arise as you run into new edge cases.

3. Status Quo

You have a model for integrating with a majority of employment systems, and your large customer base has grown accustomed to the SFTP, flat files, and existing integrations you have. But you've found that new leads have sophisticated requirements, and now, you're looking to drive digital transformation, set a higher bar for the customer experience, and unlock efficiencies across your practices.

Your challenges:

- **Slow digital transformation**
  Most efforts to modernize processes and systems require major overhauls and don't offer a clear ROI for years to come.
- **Non-differentiating support costs**
  Old integrations and file transfer systems eat up a lot of your support resources. As you scale, the costs add up quickly without adding value, because your customers view support as table stakes.
- **"Good enough" coverage**
  You've gotten to a point where you're not supporting newer employment systems because it doesn't make economic sense to build for the long-tail, which puts constraints on your total addressable market.
- **Opportunity costs**
  Due to the variation in data syncing processes, the time to get an employer up and running can take months—diverting critical engineering resources away from core product enhancements and features.
- **Endless upkeep**
  Maintaining your existing integrations is a thankless responsibility that drains valuable internal resources in perpetuity.

No matter which phase you're in or set of challenges you face, your bottom line and overall user experience are at stake. B2B applications looking to gain market share and improve customer retention need to continue investing in their integration strategy to match the evolving expectations of employers.

A Tale of Two Integration Approaches

Trainual—a SaaS platform that helps employers scale their operating procedures and employee training protocols—had a data challenge. When a business signs up for Trainual, every employee in that organization needs to be entered into the system in order to properly map their roles and responsibilities.

To deliver a seamless experience, Trainual knew it had to integrate with the systems its customers use to manage their employee records. The alternative—asking customers to manually enter or upload employee data—was a nonstarter for the tech-first company.

At first, Trainual built point-to-point integrations with individual employment systems. As its tech team added more integrations, the undertaking became more complicated and time-consuming, leaving them with a decision to make: Should they continue their course of building one-off integrations? Should they build an API? Or should they outsource integrations to a third-party provider?

For the fast-moving startup, the decision was clear: partnering with an integrations provider would allow it to meet customers' expectations in a much shorter time frame.

With Finch's unified API, Trainual immediately grew its integration coverage by 137% but only had to build to one centralized point instead of hundreds, saving weeks of time and effort.

Combined with the effects of outsourcing integration maintenance to Finch, Trainual realized a 75% reduction in development costs.

Unpacking the Build-Versus-Buy Debate

When you undertake an integrations initiative, there are essentially two paths forward: build in-house or outsource the work to a dedicated partner.

Some B2B applications want to retain the work because it's important to them to have tight control over the end-to-end process. In other cases, an application might only anticipate building one or two integrations and believes it will be easy enough for its development team to take on.

Other B2B applications understand integrations to be a sophisticated but auxiliary technology outside of their wheelhouse and recognize the merits of outsourcing, which allows them to focus their internal resources on the proprietary technologies that define their core product.

Both approaches are worth parsing.

Let's say you decide to build in-house

Where should you begin?

To clearly understand the scope of the work in front of you, you'll want to kick off your efforts with a discovery phase to account for considerations like:

The data sources you will need to integrate with to provide adequate coverage for your customer base
One of the reasons integrating with employment systems is so challenging is the highly fragmented nature of the payroll processing and HR management software industries. There are over 5,700 providers in the U.S. alone, and the top three systems only have a 44% share of the market. That leaves a huge long-tail, with each other provider accounting for no more than 4% of the market. Learn more in Finch's Guide to the U.S. Payroll Landscape.[9]

9 https://tryfinch.com/resources/whitepapers

Unless you're targeting a very niche, homogeneous audience, covering your customer base in a meaningful way means integrating with potentially dozens and dozens of systems. The exact number can vary, so it can help to perform a study of the exact employment systems used by your target market. Our data shows that even targeting a subset of 1,000 SMB and mid-market employers may require compatibility with up to 34 systems.

Your team's level of expertise in the HR and payroll space
Building the connection between systems is just one component of integrations; there is also the issue of data mapping. Essentially, every data source you integrate with will use a different data model, and each source value you retrieve from them has to be mapped to a target data field to make it operable for your system. If your team doesn't have a deep understanding of the nuances and terminology specific to the HR and payroll domain, accurate mapping can prove challenging.

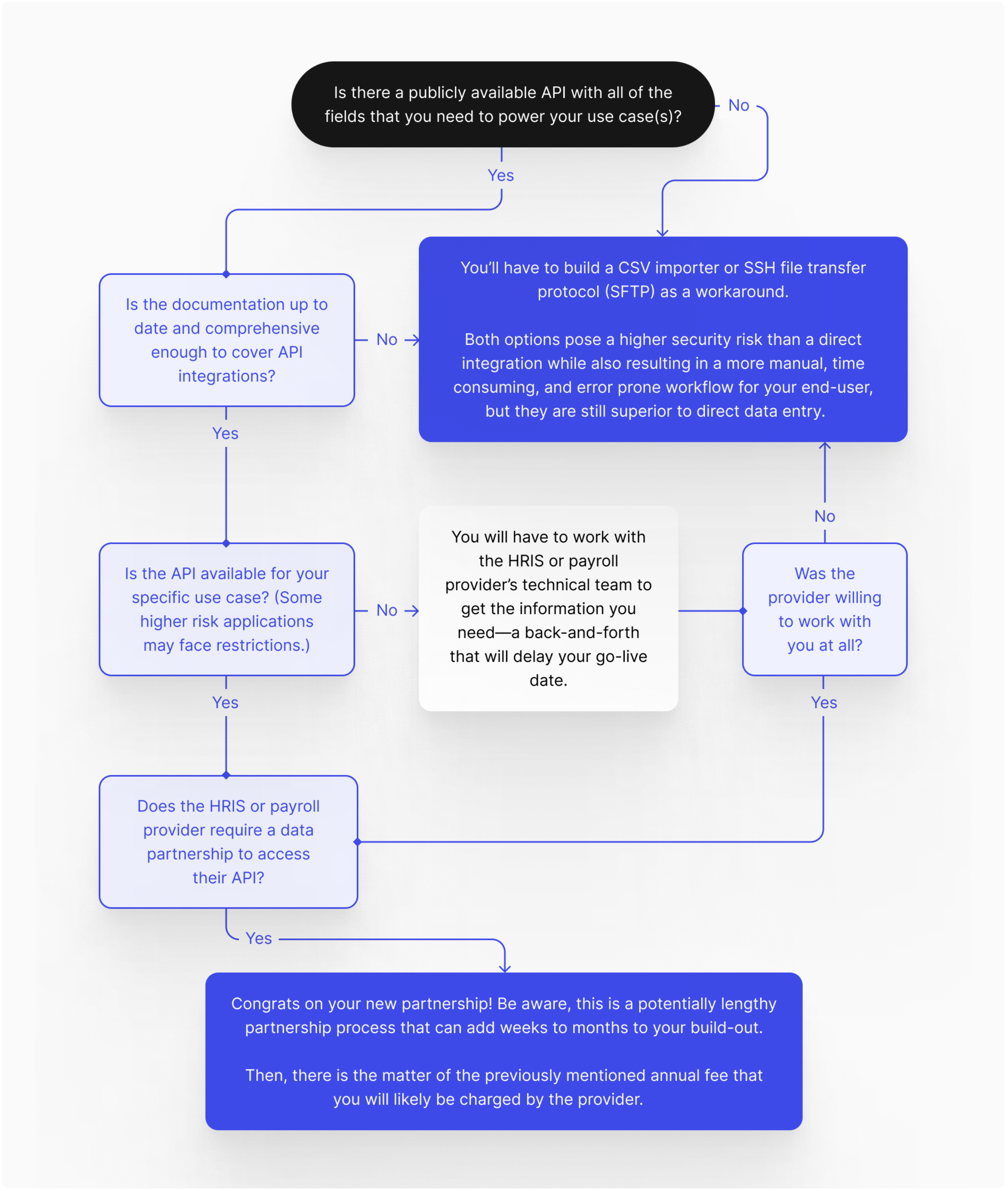
Your team's bandwidth and priorities
Integrating with an employment system takes a minimum of three months for a basic employee census data pull and anywhere from six to nine months to achieve 360° read-and-write access. Before you set out to build, you'll want to make sure you have the capacity to take on the work as well as the flexibility in your timeline, keeping in mind that time spent on integrations is time you could otherwise devote to perfecting and evolving your core product.

The hard and soft costs of building in-house
Integrations come with a heavy price tag when you consider the average engineer's salary and the hours they spend establishing data connections. If you assume that three engineers will work on an integration for three months, that puts the soft costs of just your initial buildout in the ballpark of $200,000. Then, there are the hard, ongoing costs to consider, like the fees many systems charge to use their API. For access to a legacy system like ADP, the hard and soft costs combined can run in excess of $1 million.

There are also opportunity costs to consider; how much more could you accomplish and how much faster could you get to market if your team's attention wasn't split between your product and the ancillary technologies that support it?

You’ve chosen a system to integrate with. What’s next?

Before you begin building, consider the nuances of the specific HRIS or payroll system you’re targeting. For starters:


Is there a publicly available API with all of the fields that you need to power your use case(s)?
No
Yes
You’ll have to build a CSV importer or SSH file transfer protocol (SFTP) as a workaround.
Both options pose a higher security risk than a direct integration while also resulting in a more manual, time consuming, and error prone workflow for your end-user, but they are still superior to direct data entry.
Is the documentation up to date and comprehensive enough to cover API integrations?
No
Yes
Is the API available for your specific use case? (Some higher risk applications may face restrictions.)
No
You will have to work with the HRIS or payroll provider’s technical team to get the information you need—a back-and-forth that will delay your go-live date.
Was the provider willing to work with you at all?
No
Yes
Yes
Does the HRIS or payroll provider require a data partnership to access their API?
Yes
Congrats on your new partnership! Be aware, this is a potentially lengthy partnership process that can add weeks to months to your build-out.
Then, there is the matter of the previously mentioned annual fee that you will likely be charged by the provider.

The ins and outs of API partnerships

In many cases, the HRIS or payroll system you're integrating with makes formal partnership a stipulation to accessing some or all of its API. If you find yourself in this situation, you will want to be prepared to go through a potentially lengthy partnership process that can include:

- Creating an account with the provider
- Applying for partnership
- Understanding and adhering to the provider's rules and standards for accessing its API
- Submitting your security artifacts and undergoing a review of your technical requirements and data security protocols, especially with regard to personally identifiable information
- Agreeing to regular audits of how you handle the data accessed
- Legal review and contract negotiations

In total, partnership can add weeks to months to your build-out. Then, there is the matter of the previously mentioned annual fee that you will likely be charged by the provider (see "The hard and soft costs of building in-house" on **page 12**).

"We work closely with employment systems, not just as data partners but as key stakeholders that help us shape the future of employment data. Together is the only way we're going to power the next wave of innovation in our space. For B2B applications looking to leverage employment data, building bespoke, one-off integrations to each system requires a lot of engineering time that they either don't have or would rather spend on their core product.

Finch solves for this specific problem while also opening opportunities for applications on our platform to establish marketing and distribution partnerships with employment systems. We are only just scratching the surface of what's possible as we continue to align applications, employment systems, and employers towards a connected future."

Runae Lee
Head of Partnerships at Finch

Authorization models

The provider you are integrating with will also stipulate an authorization model that you will need to conform to in order to ensure that the user making the API request has proper permissions to access or change the data in question. If you are integrating with multiple providers, you will run into a variety of approaches, including basic auth, OAuth 1.0, OAuth 2.0, and private API keys.

- **Basic auth** entails sending an base-64 encoded username and password with each request. The credentials are encrypted via an HTTP transport protocol to safeguard private information.
- **OAuth 1.0** forgoes usernames and passwords, and the potential security risks they pose. In their place, temporary and access tokens are exchanged between your application, the system you're integrating with, and your user.
- **OAuth 2.0** differs from 1.0 in that it employs an authentication server to communicate with the API server. In this method, flows can take many different forms, including the exchange of authorization codes and access tokens. One common example of OAuth 2.0 is when web applications prompt users to log in via third-party systems like Twitter, Google, or Facebook.
- **Private API** keys are a less secure option that work by identifying the calling application (rather than the calling user) but allow the system to more closely track how its data is being used. The mechanism at play is a unique string of randomly generated characters, unique to your application, which must be included in the request header. This server-to-server method restricts API access to registered applications.

You built an integration. Mission accomplished?

Not quite. Laying the rails is really just the beginning. There are many ongoing efforts you have to account for:

Authorization in the Wild: ADP

Here's just one example of an authorlzation model you might have to accommodate. For access to its data, ADP stipulates the following:

1. A user accesses your application.
2. Your application redirects the user to the ADP authorization endpoint.
3. ADP authenticates the user's ADP credentials and obtains their permission to access the data requested by your application.
4. ADP redirects the user to the redirect URI pre-registered with ADP. In turn, your application receives an authorization code.
5. Your application contacts the ADP token endpoint to exchange the authorization code for an access token.
6. ADP authenticates your application, verifies the authorization code, and provides you with an access token.
7. If your application needs additional context to identify the user, your application uses the access token to leverage the userinfo API and retrieve the user's identity profile.

Making sense of the noise
First, there's the issue of standardizing the data you retrieve so that it can be aggregated with other sources, operated on by your application, and understood by your users. The solution is data mapping (i.e., matching data fields from the provider's database to your own to ensure a common data structure).

The kicker is, as you add more integrations and have to standardize additional data models from disparate sources, data mapping only becomes more complicated, and if mapping is not done correctly, it can precipitate a lot of manual interventions that practically negate the benefits of an API integration in the first place.

For example, let's take a look at what should be an intuitive data point to map that is not standardized across systems. Zenefits has four employment types and five subtypes for contractors, which is also complemented by a comp type to indicate salaried versus hourly workers. Conversely, Rippling only provides six employment types with no subtypes that also include comp types in the name.

	zenefits	RIPPLING
Employment Type	full_time	salaried_ft
	part_time	salaried_pt
	intern	hourly_ft
	contract	hourly_ft
		temp
		contractor
Employment Subtype for Contract	agency_paid_temp	
	company_paid_temp	
	independent_contractor	
	vendor_employee	
	volunteer	
Comp Type	salary	
	hourly	

Building a normalized data model that's compatible with just two systems is no easy task, and challenges compound further when incorporating the various data structures of more complex systems like ADP WorkforceNow or Workday.

Beyond mapping, there is the issue of data enrichment, which is the process of adding value to the data you retrieve. Data enrichment may involve rectifying missing or ambiguous fields or extracting additional meaning from the raw data you pull. It's a sophisticated layer of data processing that requires specialized expertise in the HR/payroll domain, general data analysis, and your particular use case. Without it, the data you're retrieving is likely usable but isn't reaching its full potential.

Keeping up with the upkeep
Then, there's maintenance. Integrations are far from "set it, and forget it," and once built, they require endless support:

- **Refreshing your cache**
  Without a continually fresh cache, your data pull isn't much better than a static, analog spreadsheet. To reap the benefits of streaming, real-time data, you will have to manage the process of data synchronization, including choosing the frequency at which to make API calls—a decision not as straightforward as it might seem. When setting your synchronization protocols, one thing you'll have to keep in mind is the rate limit imposed by the HRIS or payroll provider you're working with. Exceeding your rate limit—that is, the number of API calls your application is permitted to send in a given time period—can result in the suppression or failure of subsequent API calls.

  ADP, for example, limits calls to under 300 times in a 60 second period, with no more than 50 concurrent requests in any time. When the rate limit is exceeded, ADP first throttles additional requests before returning a 429 error.

  You'll also want to plan around your data source's scheduled maintenance windows and account for peak server traffic, among other considerations.

- **Ensuring proper storage and access**
  Because you'll be handling sensitive data like PII and payroll information, the provider you're connecting with will require or recommend that you uphold certain requirements around the storage of the data you retrieve.

  To start, you'll need to create a classification model for different types of data that may be more sensitive and ensure you're allocating appropriate resources to keep it secure. Then, you'll need to make use of TLS 1.2 to encrypt data in transit and use AES-256 bit encryption for data at rest. Of course, you'll also need to implement a strict data segregation and data access policy to complement the infrastructure protocols to protect against possible data leaks.

- **Rolling with the updates**
  The APIs that you build to will issue updates, and you will have to adapt your integration as necessary to keep things running smoothly for your users. Some providers update their APIs very frequently, others less so, but it will be critical for you to have point persons charged with monitoring your data sources for new releases.

  Meanwhile, when an API has outlived its relevance, become nonfunctional, or been superseded by a newer, better version, it reaches a point in its lifecycle referred to as deprecation. When that happens, support for the API officially ceases, and you will need to overhaul your integration plan accordingly.

- **Efficient error handling**
  Just as APIs will inevitably deprecate, integrations are bound to fail at one point or another, whether due to a system change or something more preventable like exceeding your rate limit. When they do, your customers will run into errors that they expect you to address quickly and transparently. That puts the onus on you to create a routing system for error handling and protocols for maintaining open lines of communication with your customers throughout the fix. It can also mean spending undue time on the phone with providers' technical teams to troubleshoot more complex issues.

- **Model modifications**
  By the time you're in the integration maintenance phase, you've already developed a comprehensive data model for organizing all incoming values, which should define elements like field names and how those fields relate to each other. This model is based on your use cases and the data models used by your data sources, which means changes to either will require updating your application's own model.

  Let's say an employment system updates their API to include another level of granularity into deduction codes and deprecates more generalized codes that you were previously relying on. Your standard model may not have those new codes mapped, especially since they're different across every system. In order to become compatible again, you'll need to review the edge cases where this impacts your users, identify a new deduction code categorization, and test its compatibility with other systems as well. As API changes happen across multiple systems, you may find yourself maintaining system-specific data models filled with edge cases that are not transferable across your customer base.

Forget APIs—SFTP and CSVs Are Enough for Me

We hear you. It can be tempting to stick to a flat-file solution that gets you by, rather than dedicate bandwidth to digital transformation. Or maybe you are tempted to build a flat-file feed in-house to cope with data sources that don't support APIs. But before you resign your B2B application to good-enough status, it's important to consider the short- and long-term implications of that decision.

First and foremost, you should acknowledge the direction your target customers are heading in as it pertains to their own digital transformation. By and large, most companies you want to work with are prioritizing streamlined experiences internally and expecting the same from their vendors. In turn, you'll want to think about the UX of your end-to-end workflow and how your flat-file feed may be causing undue friction:

- **Not self-serve**
  Setting up a file for transfer requires your customers to build out a report, which they may not compose correctly for your purposes. In those cases, you'll have to jump on the phone with them to walk them through the process step by step.
- **Too many cooks**
  Many times, your customers will have to call their payroll or HRIS rep to set up the SFTP, which can take days or even weeks to complete successfully and adds yet another task to your customers' plate.
- **Pay to play**
  Your customers may have to pay a fee to their payroll or HRIS provider to enable SFTP.
- **Prohibitively technical**
  Depending on the protocol, which is different for every system, your customers may have to input host domains, keys, and other server-specific information into their system to establish a scheduled sync. This can be burdensome for a non-technical user.

There are other technical challenges to consider as well:

- **Deviations are the norm**
  Each HRIS and payroll system has its own CSV format that may not conform to your data model. You'll have to accurately map to each one—a time-consuming and potentially challenging process depending on your team's familiarity with employment data's idiosyncrasies. If you're also dealing with custom reports, you'll have to map to those separately.
- **Laying the groundwork**
  You'll have to configure an SFTP server and exchange profiles—and maintain them—to ensure you can upload data from various sources.
- **Security concerns**
  If you're providing credentials or keys to set up SFTP feeds, you'll need to manage them per employer to ensure there's minimal cross-contamination of data between your customers.
- **Far from real-time**
  Depending on your sync frequency, you may miss critical data updates that are necessary for your product experience.
- **Proper storage**
  By nature of an SFTP, you'll be handling and storing large volumes of personal data. Your customers and their HRIS and payroll providers will expect ironclad security protocols are in place to protect sensitive information. You can be held responsible in the event of a breach, especially if proper security measures aren't in place.

Finally, there's the matter of costs:

- **Support-intensive**
  Expect to devote 2 to 3 hours of support time per employer to set up an SFTP and ensure a sync is established.
- **Development drain**
  Your engineering team will devote countless hours mapping and validating data from each system—not to mention handling one-off cases involving custom reports and incorrectly labeled data.
- **Server spend**
  The server you'll be compelled to configure and maintain will cost tens of thousands of dollars or more, depending on the amount of data you transfer. As you grow, those costs will grow with you.
- **Lagging implementation**
  Getting an employer onboarded via a flat-file feed can take weeks, prolonging fulfillment of your product's performance obligation and, ultimately, delaying revenue recognition.

In short, while forgoing an API solution might initially seem easier, it's likely the less prudent option for your business.

Or you could take a buy approach

What does it look like to outsource integrations?

As we have seen, in-house integrations are complex and resource-intensive. To get the benefits of integrations without the pitfalls, many applications turn to outsourcing.

In a fragmented industry with 5,700+ employment systems, outsourcing your employment system integrations to a provider like Finch saves time and money. Instead of building integrations with dozens of HRIS and payroll systems yourself, Finch does the work for you while also providing ongoing support and maintenance. This outsourced infrastructure lets your team focus on what they do best: fine-tuning your product, executing your roadmap, and creating the best user experience possible.

Let's take a closer look at the impact of outsourcing your integrations on several key areas:

- **Speed to market**
  Once you decide integrations are key to the success of your product, you want to get them live as soon as possible—making three to nine months for a single in-house build seem particularly daunting. With Finch's unified API offering read and write (360°) integrations into an ever-growing list of HR and payroll systems, all of the time-consuming technical and legal legwork is done for you, shaving weeks to months off of your initial build-out. In fact, most Finch customers are up and running in less than a month.

  What's more, because Finch connects you to dozens and dozens of systems that can be turned on with practically a flip of a switch, you'll cumulatively save months to years of additional development time down the road as you expand your coverage.

  By allowing your team to recalibrate their focus on your core technology, outsourced integrations permit you to get other elements of your product live faster, putting you at an advantage over your competitors.

+

Customer Story

Finch enabled Mosaic to establish connections with employment systems **94% faster** than if they had built them all in-house.

Read more at tryfinch.com →

- **Operational efficiencies**
  Speaking of your team's focus, an integration partner like Finch relieves them of distracting burdens at every stage of an integration's lifecycle—not just build-out—reducing instances of context switching that drain productivity. From day one through long-term maintenance, Finch assumes responsibility for error handling, troubleshooting, API updates, and more, all with a commitment to high-touch support.
- **Hard and soft costs**
  Operational efficiencies lower your overhead and reduce the soft costs of integrations by potentially hundreds of thousands of dollars with each system connection you turn on.
- **Coverage**
  When you build on top of Finch's single, unified API, you enable data connectivity to dozens and dozens of HRIS and payroll systems in one fell swoop—and our network is constantly expanding. Critically, we pay particular attention to the employment domain's (extremely long) long-tail, so you can be confident that a majority of your customer base is covered. Finch currently supports over 160 integrations and covers 88% of U.S. employers.[10]
- **Monetization opportunities**
  With Finch, it only takes a moment for your customers to authorize you to access their HRIS and/or payroll system and turn on all the automations that connection enables. Compared to uploading CSV files, your customers are looking at hours of time savings—something you can charge a premium for. For many B2B applications, that turns into a new revenue stream with the potential to offset the cost to integrate in the first place.

Customer Story

Lane Health saves its customers **8 to 12 hours of admin time per month** with Finch

Read more at tryfinch.com →

10 https://tryfinch.com/developers/integrations

“Go long” with Assisted API Flow

To solve for the long-tail of employment systems, Finch takes a comprehensive, two-pronged approach. Our automated API integrations already cover dozens of (and counting) HRIS and payroll providers, with all the myriad of edge cases. In cases where you need to integrate with extra-niche employment systems, we also offer a semi-automated product called Assisted API Flow.

Assisted API Flow takes on the security, build, and maintenance burdens of connecting with smaller providers that present more barriers to integration, unlocking their data in as little as two weeks. Some providers, for example, require you to submit a form each time you want to pull or refresh data. We do that for you, delivering comprehensive and up-to-date data to you in regular intervals, so you can deliver an exceptional experience to your customers without being saddled with perpetual admin work. The best part? You’re still interacting with the same unified API so there’s no need for you to change your process or build anything new.

That means, on the off chance you’re trying to land a customer that uses a system not yet covered by Finch, we have a gameplan in place to quickly expand coverage so you don’t lose a sale. The best part is that we are always working to turn assisted integrations into fully automated ones, so that, in time, even the most obscure employment systems are in the Finch fold.

“As a new request for an integration comes in from one of our customers or we see a new provider that’s coming up, we’ve had a great experience going to the Finch team and asking, “Is this something that you have on your roadmap?” And so far, every single one we’ve recommended has been added within about a week. As people request it, Finch is adding integrations quickly.”

Taylor Sell
Director of Product at Trainual

About Finch

Finch does the hard work of integrating with HRIS and payroll providers to facilitate the secure, permissioned flow of critical business data. Our dynamic, unified API offers read-and-write access and abstracts away inconsistencies across systems for optimal usability no matter the source.

Finch is quickly becoming the API of choice for employment system integrations because we are:

- **Developer-friendly**
  We focus on developers and empower them to create world-class solutions
- **Reliable**
  Our API does the best job of interacting with and maintaining connectivity with employment systems
- **Secure**
  Finch is a pass-through system, and is SOC2 Type 2, CCPA, and GDPR compliant
- **Efficient**
  Finch connects you with 160+ HRIS and payroll systems through one API, 4x more than any other platform
- **Enterprise-ready**
  Our technology is built for large-scale synchronization with thousands of employers

Reach out to our team to explore ways to innovate on employment data together, or directly partner with Finch to empower the employment data ecosystem.

To learn more, visit tryfinch.com

finch