Integrations Handbook for Partnerships Leaders

pandium

integrate all the things

pandium
integrate all the things

Table of Contents

Introduction

Technology partnerships are becoming increasingly vital to SaaS companies. Integrations with other SaaS products are no longer a nice to have, but vital to a company's survival and success.

However, most technology partnership leaders come from a business development background, not engineering. Sometimes product managers or marketers assigned to in-app integrations have technical backgrounds, but they just as often do not.

This primer is designed to give partnership leaders (and PMs and marketers) a clear explanation of all the tech concepts that are involved in building an in-app marketplace and product integrations.

Why does this matter? Why should a partnership or other business person try to understand the technical considerations around integrations?

Poorly built integrations and integration infrastructure can break a tech partnership program.

The ROI on tech partnerships can quickly disappear when customers have a bad user experience, partners don't like working with your API, and your engineers are spending most of their time on maintenance and support.

As a result of an ad hoc approach to building integrations, ShipBob was spending 80 percent of their engineers' time on integration maintenance and customer support. Read how they remedied this.

Why Partnerships and PM Leaders Should Read This Primer

So why not just trust your engineering team will get this done properly, and leave it at that?

Because unlike the core product, integrations are often an afterthought or secondary priority of the engineering team.

Engineers assigned to the task are often inexperienced with building integrations and integration infrastructure, and view it as a distraction from their core work.

The systemic and strategic approach to building the core product is often replaced by an ad hoc approach that results in varying quality and disparate systems.

Even when an engineering team is interested in and dedicated to integrations, there are business considerations and logic that they might overlook.

As a result, a tech partnerships person can't afford not to understand the basics of integrations and integration infrastructure.

If you understand what goes into building product integrations and in-app marketplaces, you can ask the right questions, have meaningful conversations with your product and engineering team, and help to ensure that your integrations and their infrastructure technology satisfies both your customers and your partners.

The commercial success of technology partnerships hinges heavily on the right integration infrastructure and processes being put in place.

As a partnership or business leader, this primer will help you to ensure that your organization is wisely spending their engineering resources, and building an infrastructure that allows you to drive more revenue from your partnerships.

This primer is sorted into subject matter chapters, with questions you should ask at the end of each chapter.

There is also an Index in the back where you can locate any terms alphabetically and click on the link to the page.

pandium
integrate all the things

1. All Things API

If you're new to tech partnerships, you might think an API is an integration. They are closely related, but you can build a SaaS product integration without an API. (And you might need to if you're dealing with legacy systems or an industry that isn't tech savvy.)

An API is a tool that is used to build an integration. In the vast majority of cases, APIs will be utilized to build a SaaS product integration.

What exactly is an API?

An API (application programming interface) is a computing interface that allows two or more software to communicate and interact with each other.

This interface abstracts away from the custom code of each application, helping the two applications to communicate without knowing the details of the other application.

Metaphorically, you can think of it as a person who only speaks French (SaaS application 1) using a translation app (the API) to communicate with someone who only speaks English (SaaS application 2).

With the translation app (the API), the person who speaks only French (application 1) does not need to know English to communicate and interact with the English speaker (application 2), and vice versa. And they can only communicate what the translate app is coded to translate, not every word.

Another common metaphor is to think of a drive thru restaurant. A driver (application 1) comes up and reads the menu (the API) and makes a request using the language of the API. The restaurant (application 2) then returns the item that was requested.

Having a standardized menu means that the driver (application 1) does not need to become familiar with all the specific ingredients and methods of the restaurant (application 2), which saves everyone time and complication.

Think of it as your product has custom and complex code, and so do all the other products. An API is a more abstract layer that means products can talk to each other without needing to see or access the other product's custom code.

This not only makes communicating easier (you don't have to learn the other system; you can just learn the API, which is more standardized), it is more secure (you don't need to be inside the other application's database).

What is a private, partner, public, and open API?

APIs are used both internally and externally.

Internal APIs connect various parts of a company's infrastructure. These APIs are private, meaning no external parties have access to them.

External APIs can be offered to technology partners, customers, or the public to use.

Some companies only offer their external API to customers. Usually this is because their customers have integration needs, but the company has not invested enough time or resources in the API to make it suitable for widespread use.

Of the **1000 fastest growing SaaS companies**, 88 percent offer their customers an API to use.

A Partner API is when a company opens up access to third parties whom they approve as partners. The key distinction between a Partner API and a Public API is that the company has a list of requirements and criteria that it applies to decide which companies can access the Partner API.

A Public API has public documentation, and is available for use either without registration, or (more commonly) through a simple registration process that is not designed to vet applicants, but only to track usage and enforce rate limits and security issues.

In general, a Public API will use standard protocols so more people can make use of them.

Open API is used interchangeably with Public API.

OpenAPI specification, on the other hand, is a description format for REST APIs that helps standardize and document APIs.

pandium
integrate all the things

What different protocols or architectural styles of APIs are used to build SaaS integrations?

At the moment, the most commonly used style of APIs for SaaS product integrations are REST APIs with a JSON response. If you're starting from scratch, most likely this is what you would use, but it depends on your product and use case. REST is an architectural style, not a protocol.

Prior to the proliferation of REST APIs, SOAP APIs were popular, and they are still used. Salesforce, for example, offers **both REST and SOAP APIs**.

SOAP is a protocol so it is more rigid than REST in its requirements, which include security features and strict rules around message formatting. It requires more bandwidth and is more complex than REST; as a result, it is slower.

SOAP's **advantages are** greater standardization and built-in security, while REST's **advantages are** speed, scalability, and flexibility. Practically, SOAP is most commonly used in legacy applications, in enterprises, or when partners need a rigid protocol to enforce.

A newer style that has gained some adoption is GraphQL. GraphQL allows the requesting application to define what data it wants, instead of the resource application defining what data will be sent, as is the case with REST. In addition, GraphQL allows you to see which fields and data clients are requesting, and that is extremely helpful to modifying the API.

PayPal **explained why** they moved their PayPal Checkout from REST to GraphQL.

pandium
integrate all the things

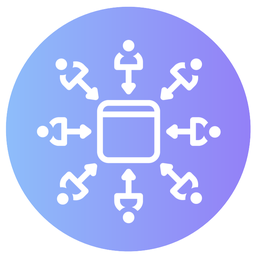
Most Commonly Used External APIs

SOAP

Most Used By: Legacy applications, enterprises, whenever rigid protocol and robust security is needed

Benefits: Standardized due to more requirements, security

Drawbacks: Slow, complex, rigid, less familiar to newer engineers



REST

Most Used By: Modern SaaS companies; REST with a JSON response is the most commonly used external API style for SaaS companies

Benefits: Commonly used, speed, scalability, flexibility

Drawbacks: Less standardized so can have different variations, security not built in

GraphQL

Most Used By: Companies that need speed and process lots of data

Benefits: Efficient and faster than REST, flexibility, easier to update

Drawbacks: Less established, fewer engineers have used, caching not built in, monitoring more difficult

pandium
integrate all the things

What is API documentation?

API documentation provides a detailed guide to a company's API, including authentication, data schema, endpoints, functions, and more. It is for third party developers (whether customers or tech partners) who are interested in building integrations with the API.

API documentation is crucial to encouraging third party developers to build integrations to your product. Your documentation should be thorough and detailed. If you have poor or hard to find documentation, you will lose potential technology partners as well as provide a frustrating customer experience.

Your documentation should clearly map out: your authentication protocol, endpoints, actions, code samples, guides to building with the API, and a changelog. Dropbox, for example, does **an excellent job** documenting their API.

In **our study of the 1000 fastest growing SaaS companies**, we found that 57% of companies have public documentation of their API.

What is API versioning?

APIs will inevitably change as the underlying product changes, the use cases change, and companies learn more about what their customers need integrations for.

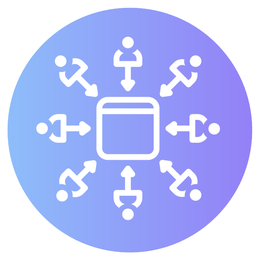
Some changes can be made to an API without needing to release a new version of the API. For example, you can usually add a new endpoint to an API without it causing a breaking change. In this case, you'd simply want to document this change and communicate it to third party developers.

A “breaking change” is exactly what it sounds like - a change that means part of the old API will no longer work. **For example**, changing an existing field’s data type, or removing an endpoint would be a breaking change. An integration that was designed to move registrants from one application to another will no longer work if the ‘registrants’ endpoint is deleted.

When there is a breaking change, companies usually introduce a new version of the API. This allows third parties and customers to continue to use the old version if they want. Often, a company will communicate how long the older version of the API will be supported, giving third parties time to update their integrations to work with the new version.

You should build a way to track and manage API versions so that you can provide the proper support, and ensure your customers and partners do not end up trying to use a deprecated version of your API.

What is an API call, endpoint, method, and parameter?



An API call is what it sounds like - when the API is queried with a request. So if Mindy’s Marketing Automation platform asks the Twitter API to return any tweets made in the last week from @samjohnson4765 that would be 1 call.

APIs are made of endpoints that define where particular data is. For example, users, tweets, campaigns, or audiences might all be endpoints that a request can be sent to.

The method is often next to the endpoint in a request, and the method defines the action to take on the data. The HTTP methods used in REST are GET, POST, PUT, DELETE, and PATCH.

GET /users, for example, would fetch a list of the users in the application.

API parameters add further details to a request. For example, if you wanted to return users who were active in the last month, and have the list of their names but not emails returned in alphabetical order, that might be something you could put as a parameter in the request.

What is an SDK?

An SDK is a software development kit, which is essentially a kit of developer tools and code that makes it easier for developers to build on a particular platform. Often SDKs will include APIs.

Especially on larger SaaS platforms, SDKs are often offered as a language-specific shortcut for building integrations with the platform. Shopify, for example, **offers a Javascript SDK**.

What are client libraries?

Client libraries are blocks of code written in particular coding languages that can execute the API. They save developers the time from having to write all the code themselves when using the API.

Larger SaaS companies offer more client libraries in a wider variety of languages to encourage use of their APIs. Here are some of **Google's client libraries**, for example.

What is a synchronous and asynchronous API call?

A call to an API is a request. A synchronous call means the application will wait for a response from the API to execute the next step. An asynchronous call will instead proceed even without a response.

pandium
integrate all the things

Why Your API Matters

A well-designed and well-documented API is a necessary tool for successful tech partnership programs.

If your API is hard to work with or does not meet the customers' needs, it is going to take more time to build integrations to your product for a worse result.

It's like trying to build a skyscraper with a rickety crane that no one is sure how to work. It can be done, but it's going to be a lot more expensive, time-consuming, and will likely lead to flaws in the building.

In contrast, an API that is well-designed and well-documented will lead to more third party developers wanting to build into your product, and it will make building integrations that meet customers' needs easier and more fun.

pandium
integrate all the things

QUESTIONS TO ASK ENGINEERS: APIs

- [ ] **Why did you choose this style of API?** 1
- [ ] **Is there anything irregular or unusual about the API, and if so, why?** 2
- [ ] **Did you consult with product, customer success, and customers to design the endpoints, methods, and parameters?** 3
- [ ] **Is the API documentation clear and thorough and easy to locate?** 4
- [ ] **Is there any reason not to make this a Public API?** 5
- [ ] **What is our system for tracking versions, and how often do you anticipate having to release a new version?** 6

pandium
integrate all the things

While an API is an interface that makes it easier to communicate with an application, an integration is what actually moves the data between two or more applications.

Companies often build integrations to connect their own systems. For example, Eddy's Enterprise might build an integration from their Salesforce account to their homegrown ERP, or between their homegrown ERP and their homegrown BI platform, or even between their Salesforce account and their Hubspot account.

Sometimes, Eddy's Enterprise might use internal APIs or third party APIs for this purpose; other times they might build connections directly between their own systems.

As you can imagine, there is a big difference between the apps built amongst one company's internal systems, and connections that are offered to hundreds or thousands of customers.

A product integration is built between two systems so that any users of both systems (on the appropriate plan) can move data between their accounts in each system.

What is a product integration?

When you form a technology partnership, you typically want to build a product integration. As covered, an API is not the integration, but the likely interface which will be used to build an app that will send and retrieve data back and forth.

While an integration is a built connection between two systems that enables data to move, a product integration is a type of integration that enables all customers of a SaaS company to move the data from that account to their account with a different SaaS application.

So, for example, if Busy Bee's Soap wants to move their list of Zoom webinar attendees into Hubspot, they could download the CSV file of the attendees from Zoom, re-format it, and then upload the list into their Hubspot account.

This is not an integration because Hubspot and Zoom are not talking to one another. And, of course, it takes a significant amount of time to manually move data like this, and it opens the door for human errors.

Instead, Busy Bee's Soap's developers might build an integration for themselves using Hubspot's and Zoom's APIs. The Busy Bee's developers could program their integration so the attendees' name and emails are automatically sent to Hubspot, where they become a new contact or are noted in an existing contact as being a webinar attendee.

That would be an integration built by Busy Bee's, and the advantage is, within the constraints of the Hubspot's and Zoom's APIs, Busy Bee's can customize this integration to their particular business needs.

In contrast, a product integration would be if Hubspot builds an integration to Zoom (or vice versa) so their joint customers can access the functionality outlined in the paragraph above.

In this case, Busy Bee's Soap - as well as every other Hubspot/Zoom customer on the appropriate plan - doesn't have to build anything; they can simply follow the instructions from inside Hubspot or Zoom to install and run the product integration.

The plus side of this is Busy Bee's developers do not have to be involved. The potential downside is that the Busy Bee's marketer is dependent on whatever options Hubspot designed for the integration. If Hubspot required, for example, that someone who registers on Zoom enters their name, email, and company to be added as a contact in Hubspot, that might not align with the marketers' goal to require fewer fields.

Product integrations can be relatively simple - say sending the emails and names of people who registered for the Zoom webinar into a contact field in Hubspot - or rather deep and complicated - sending data back and forth between systems where the underlying fields are not the same.

Other things being equal, the more complicated the integrations are, the longer they take to build.

Components of a Product Integration

The important parts are authorization and authentication, the configurations, the UI/UX, monitoring, scheduling, data management and performance, and maintenance.

From a business perspective, it is important to understand that a poorly designed integration is worse than no integration at all. A poorly designed integration will provide a poor user experience, and not serve the business objectives of the user. It might even allow for security breaches.

pandium
integrate all the things

In addition, maintenance on integrations can get very expensive. With a good initial design, you will minimize the maintenance costs. With a poor design, your engineering resources will quickly be diverted from building new integrations to maintaining old ones.

Important Components of a Product Integration

1. Authentication and authorization protocols
2. Configurations
3. User interface and UX
4. Data management and performance
5. Scheduling
6. Tracking and logging
7. Responding to API or product updates

We will cover the different components of a product integration in later chapters. But there are other concepts related to building product integrations that will frequently come up.

What is a webhook?

A webhook is when a HTTP request is made to a URL upon a defined event. So, for example, a webhook might be set up so that when someone is tagged on Twitter, the tweet appears in Slack.

The advantage of webhook is that it only sends requests when there is relevant data. For example, if API calls were being used to move data from Twitter and Slack, the API would be routinely calling to see if anyone was tagging the user in Twitter. If they aren't being tagged very much, it's a lot of wasted calls.

In addition, the webook will send the request immediately. With an API call, you'd have to wait for the next sync.

Potential downsides with webhooks is you won't necessarily know if the other system goes offline, and you may be hit with an overwhelming amount of data. They are also better for less complicated use cases, as they do not empower as much functionality (such as deleting or modifying date) or specifications as APIs can.

pandium
integrate all the things

What is ETL and ELT?

ETL stands for extract, transform, and load. It refers to extracting data from multiple sources, transforming the data, and then loading it into another source, usually a data warehouse, and then into a visual form that users can access. ELT is a related process that moves the data into the second source, and then transforms it there.

Normally ETL refers to a company's internal process for moving all their data into a business intelligence platform where data insights can be leveraged by the business users and data scientists in the company.

What are sandbox and production environments?

A sandbox is a testing environment that developers can use to test out new code without having it go live. In the case of integrations, it means developers can test what they build before customers' access it, and avoid wasting API calls on tests. The production environment is where the code runs live.

Larger SaaS platforms, like **Marketo** and **Salesforce**, often offer their technology partners sandbox environments where they can test their integrations.

pandium
integrate all the things

What is an iPaaS?

An iPaaS, or integration platform as a service, is designed to help users integrate software.

Traditional iPaaS's are designed for internal workflow automation, not for building product integrations. Zapier, for example, is an iPaaS that your application's customer might use to create internal workflows amongst their various systems. (And Tray.io, Boomi, and Mulesoft perform the same function for larger businesses.)

Traditional iPaaS's contain a GUI where business users can set up workflows amongst applications. They typically require coding on top of this interface unless the use cases are fairly simple.

Pandium is an iPaaS specifically designed for building product integrations. Unlike a traditional iPaaS, it has a white labeled front-end that applications can customize, and has visibility into errors and usages with an Integration Admin Dashboard.

How do iPaaS's charge?

Traditional iPaaS's charge on a combination of the connectors being built, number of API calls, the number of steps in a customer's workflows, and the number of integrations that are utilized.

Pandium has simplified pricing, as it charges solely based on the number of tenants. A tenant is one customer running one integration.

pandium
integrate all the things

What is a connector?

A connector can refer to the built connection between an iPaaS and a particular system (like Salesforce), a built connection between two systems on an iPaaS (say the connection between Salesforce and Hubspot on Izzy's iPaaS), or the built connection between two systems (like Salesforce and Hubspot).

An iPaaS will build connectors for all the systems that run on their platform. Izzy's iPaaS, for example, will build a connector between Izzy's IPaaS and Hubspot, and a connector between Izzy's iPaaS and Salesforce.

This allows users of Izzy's iPaaS to use the Hubspot connector and the Salesforce connector to create workflows between the two systems, without needing to work with Hubspot's or Salesforce's APIs (instead they use the connectors on Izzy's iPaaS).

Connectors can be lighter or heavier. Traditional iPaaS's have heavy connectors where they authenticate between the systems but also define the business configurations.

With heavy connectors, you lose control over what the integrations configurations are, and you also lose the ability to update them on your own timeframe. Because the connector belongs to the iPaaS, if your application or a partnering application updates their product or API, you cannot immediately take advantage of these upgrades because you have to wait for the iPaaS to update their connector.

While the downsides of thick connectors are loss of control, customizations, and flexibility, the plus side is you don't have to code the configurations yourself.

Pandium has lighter connectors that take care of all the authentications, but allows your developers to control the business configurations in the language of their choice, and change them as needed. This gives you control and flexibility, and also means you own the code for the configurations of the integration. With a traditional iPaaS the coding around the configurations is typically not visible to the user, and is owned by the iPaaS.

What is a native integration?

A native integration is one that is built to live inside one application and connects directly to another application. This is in contrast to an integration that runs through a third party like Zapier. In the latter case, the user is directed to sign up with Zapier, and then use Zapier to connect your system with other systems.

A native integration provides the most seamless user experience as the user does not have to leave the application to connect their systems, and can execute their workflows from within the application. The application is also the one in charge of defining and updating the integration configurations that are vital to the integration's success with customers.

Most iPaaS's are not designed to build a native integration experience. Though some, like Pandium, are. These type of iPaaS's are sometimes referred to as embedded iPaaS's.

pandium
integrate all the things

QUESTIONS TO ASK ENGINEERS: Product Integrations

- [ ] **Are we using webhooks? Why or why not?** 1
- [ ] **Have we considered using an iPaaS for building integrations?** 2
- [ ] **If we are going to use an iPaaS, does it have logging? Will we own the configuration code?** 3
- [ ] **Are we using an embedded iPaaS that offers a native integration experience?** 4
- [ ] **Are we providing a native integration experience?** 5
- [ ] **If we aren't using an iPaaS, have we compared the total cost of ownership of building in-house, and the time to market for new integrations?** 6

pandium
integrate all the things

- [ ] **What visibility do we have into the integrations our partners are building?** 7

- [ ] **Can we offer partner developers a sandbox or any other resources to make building integrations into our product easier?** 8

3. Authentication and Authorizations

pandium
integrate all the things

Authentication and authorizations are essential components of a product integration. When you open up your data to hundreds or thousands of users, and to dozens or more partners, designing authentication and authorization processes well is key to both data security and a good user experience for customers and partners.

What is authentication and authorization?

Authentication is establishing the user and application is who they say they are, and authorization is establishing that the user and application has the authority to perform certain actions.

For example, when you go through airport security, your identity is usually authenticated by you providing government-issued photo identification.

Once you enter the airport, your ticket from NYC to Oakland, California, will authorize you to board a particular plane. And you might have a keycard to a VIP lounge that authorizes you to enter the lounge.

Authentication and authorization are related but different concepts. A product integration must have secure processes for both.

From a business perspective, when you establish your authorization and authentication process, you want to meet two goals: a seamless, easy experience for your customers and partners, and keeping data secure.

The industry gold standard for authorization of integrations is OAuth v2.0. If you're a SaaS company building today, this is most likely the protocol you should be using.

However, you should be aware many companies are using different standards for a variety of reasons, and when you build an integration, a big part of the lift will be figuring out to adhere to a secure way to authorize and authenticate the user accounts from two applications.

If your authorization and authentication protocols are designed poorly or if they are unusual, you will encounter some of the following problems: potential partners who do not wish to build an integration to your product, users who cannot install the integration, users who keep getting disconnected from the integration, and security issues from users or third parties accessing functionality or data they were not meant to.

What is the industry standard for authorization, OAuth v2?

OAuth v2.0 is the industry standard for an authorization protocol. Some also refer to it as a pseudo-authentication, though it does not define a particular way of authenticating.

A common analogy to explain what OAuth v2.0 does is to think of a keycard in a hotel room. The front desk issues you a keycard that can get you into the hotel room for a limited time period. The keycard authorizes you to go into the hotel room, but it doesn't know who you are or carry your identifying information.

OAuth v2.0 leaves it to the application that is allowing access (in the analogy, the hotel via their front desk clerk) to authenticate who you are. And different hotels will authenticate in different ways.

The easiest way to understand this is to walk through an example. Imagine that you (the user) want to authorize your Hubspot account (an application) to post on your Facebook account (another application).

With OAuth v2.0, the first step would be Hubspot registering with Facebook, and receiving a client ID and client secret, which enables Facebook to know who Hubspot is and that it really is Hubspot.

When an integration runs securely, the application (Facebook) who is giving access to their data needs to be sure the other application (Hubpot) and the user (you) are who they say they are. The client ID and secret is to authenticate the application (Hubpsot), not the particular user (you).

Next, you as the user would tell Hubspot you want to authorize Hubspot to post on your Facebook account. Hubspot would then redirect you to Facebook, who will ask you to authorize this access. This is where Facebook authenticates you (the user)'s identity. How Facebook does this is up to them.

If you click YES to the authorization, Facebook's Authentication Server will issue your Hubspot account an authorization code (also known as an authorization code grant).

By Facebook issuing an authorization code to you, Hubspot does not need to know your Facebook credentials, making this process much more secure for the user (you) and the resource application (Facebook).

Hubspot will then take this authorization code (along with their client ID and secret, which proves who they are) back to Facebook, and request an access token and (usually) a refresh token.

Assuming Facebook deems the client ID, client secret, and the authorization code valid, Facebook will give Hubspot an access token and (usually) a refresh token that enables your Hubspot account to access its resource server (Facebook's application which includes your Facebook account) in the ways specified by the access token.

pandium
integrate all the things

OAuth v2 Flow

- Cathy’s CRM registers with Edna’s Emails and gets a client ID number and secret
- Sue, a marketing VP at Kikky’s Hardware, logs into Cathy’s CRM and clicks to install the integration with Edna’s Emails
- Cathy’s CRM sends a request to Edna’s authorization endpoint with their ID and secret, asking for access to Sue’s account
- Edna’s returns a screen with a series of options for authorization, such as ‘access audiences,’ ‘access names’ and ‘access emails’
- Edna’s Authentication Server authenticates who Sue is
- Sue clicks YES to authorize the data and capabilities Cathy’s CRM can exercise with her Edna’s Emails account

pandium
integrate all the things

OAuth v2 Flow

- Edna's Emails' Authorization Server issues an authorization code to Cathy's CRM
- Cathy's CRM returns this authorization code (with their client ID and secret) to Edna's Emails' Authorization Server's token endpoint and receives an access token and a refresh token
- Cathy's CRM uses the access token to request access to the relevant data in from Edna's Resource Server
- Edna's Resource Server validates the (self-contained) access token (or, if it is an identifier-based access token, Edna's Resource Server asks Edna's Authorization Server to validate)
- Once the token is validated, Edna's Resource Server returns the data (also known as the resource) to Sue's Cathy's CRM account
- When the access token is going to expire, Cathy's CRM uses the refresh token to request a new access token and new refresh token from Edna's Emails' Authorization Sever

pandium
integrate all the things

What are the tech terms used in OAuth v2?

Resource Owner

The user who has an account (i.e. Sue of Kikky's Hardware)

Client

The application requesting data from the other application

Authorization Server

The server of the application that is trying to be accessed that authorizes the user and requesting application

Resource Server

The application's database that is trying to be accessed

Scopes

These specify different levels of access that the resource owner can usually reject or approve; common categories are the ability to read, write, or delete certain fields of data

pandium
integrate all the things

Authorization code

The Authorization Server provides this code to the client once the resource owner clicks YES to approve access

Client ID and secret

A unique identifier and password that the requesting application (client) originally obtained from the application they are requesting from (resource) so that they can identify themselves

Access token

The Authorization Server issues this to the application once it validates the authorization code, client ID, and secret. It contains the scopes of what data and actions can be performed, and the expiration date. It might contain more info depending on how it is set up. JWT tokens are often used

Refresh Token

Normally, the access token will expire, and the client must turn in the refresh token to get a new access token and new refresh token

ID Tokens

OpenID Connect is a common way to authenticate the user when using the OAuth v2 flow; the ID token is a JWT token that is issued to the requesting application (the client) when a user is authenticated by the server, and it carries information about the user

pandium
integrate all the things

What are API keys?

API keys are generally used to authenticate an application or project calling an API. They are primarily used to identify an application as opposed to identifying a particular user.

Organizations define “API keys” somewhat differently. LinkedIn, for example, **uses it interchangeably** with “client ID” that identifies the application. Google, on the other hand, **specifies** that an API key is a unique application identifier that can only be used when the application is trying to access publicly available data (or certain Google data) that does not require user information.

In general, API keys are used when the resource application simply wants to identify which application is making calls against the API. As API keys do not identify the particular user, they aren't a good mechanism for granting different users different levels of access, and they don’t include multi-factor authentication.

Because API keys require only one level of security (whoever gets ahold of an API key can use it), it is safest when used only for reading a database. If it is used, it is recommended to send it in the authorization header of a request.

pandium
integrate all the things

What other terms might I see around authentication and authorization?

Basic Authentication

This is insecure so rarely recommended, but this is a method of authentication where the user puts their username and password in the header of the request and it is encoded with Base64. (An API key can also be used as the username, with no password.)

SAML

This is **an older standard** for both authorization and authentication. It entails the user requesting access to a service provider, and an identity provider validating the user, and returning that information along with their access in an assertion (sometimes referred to as a token). This is **most often used** for enterprises to enable their employees to use single sign on in order to access applications the enterprise is using.

OpenID Connect

This works with OAuth v2 to authenticate the user. It is used for authentication, not authorization. OpenID Connect allows a user to login into an OpenID provider (such as Google) and use those credentials to authenticate on another application (say Yes Accounting Software).

Tokens

Depending on how they are written, tokens might identify a user or application, their authorizations, URL restrictions, how long the token is valid, and more. OAuth v2 uses tokens in its protocol, but you can also use tokens without following OAuth v2. JSON web tokens (JWT) are most often used for APIs with OAuth v2.

pandium
integrate all the things

What are best practices around defining scopes?

In general, scopes specify what the particular user and application are authorized to do with the data in the resource server. In some cases, an application will only want to allow another application to read their data; in other cases, they might want to allow the application to delete or add data.

GitHub's scopes, for example, offer many different levels and types of access. Scopes are a combination of the allowed action (like read, write, or delete) and the data field (like contacts, emails).

Scopes should be as concise and narrow as possible. Good security practice requires giving applications and users the least amount of access possible.

From a UX perspective, it is important not to offer a user too many options; in such cases, many users will just allow all in order to move to the next screen. Instead, carefully identify what different users are looking to use the data for, and specify those options as concisely as possible.

With OAuth v2.0, the scopes are in the authorization code and then passed to the access token.

Ultimately, scopes should be defined by business objectives. You should ensure that you understand what your customers want to do with the two applications, and how they need to be connected to accomplish their goals.

One important distinction is between an admin of the application and a User. Sometimes, companies want to require that an Admin approve an integration installation, and that needs to be built into the authorizations.

Imagine Busy Bee's Soap has 700 employees and they use Slack, for example. Busy Bee's might not want every employee to be able to install Slack integrations, some of which might be security risks or simply at odds with Busy Bee's workflow and standards. Busy Bee's might want Slack to give them a way to require a user be an admin to install integrations on their account.

Why does authentication and authorization matter?

Proper authentication and authorization protocols are required for keeping data secure. And if you have unusual or insecure protocols, many tech partners will not want to build an integration to your platform because they will take time to learn and maintain, and may pose security or compliance issues.

In addition, the user experience can be significantly affected by how the authentication and authorization workflow and permissions are set up. Users want to be able to authorize the data and functions they are interested in, not a blanket permission to read, write or delete all their data. And in some applications, admin users of an account need to have more control than regular users.

When you're building an integration into another application, you might have limited control over the authentication and authorization protocols for their application. In that case, assuming there is no security risk, the goal would be to provide the best user experience for your customers while adhering to the protocols of the partner application.

pandium
integrate all the things

QUESTIONS TO ASK ENGINEERS: Authentication and Authorization

- [ ] Are we using OAuth v2, and if not, what is the reason for that? 1
- [ ] Is there anything irregular or unusual about our authentication and authorization protocols, and if so, why? 2
- [ ] Are our scopes defined as narrowly as possible while still enabling the user to complete their business objectives? 3
- [ ] Do we have regular and admin users of our product, and do they need different authorizations? 4
- [ ] How are we authenticating partner applications and users? How might his pose a security risk? 5
- [ ] How are we handling legacy protocols used by partners? 6

The integration configurations refer to the business logic of the integration. The available options will depend on how the respective APIs and the integration itself are written.

In terms of tech partnerships being successful, the configurations are vitally important: if the integration enables the user to pull names from the other system, for example, but not emails, the business user might not be able to meet their business objectives. And if the integration doesn't meet their business objectives, they won't use it.

What are integration configurations?

Integration configurations represent the business logic of an integration. Whoever is building the integration has to decide which fields of data can be moved between systems, and what actions can be performed on that data.

The best way to identify the right configurations is by interviewing customers, product, sales, and customer success and support to set up a flow of your customer's work process, and nail down how your product will interact with the other product they are using.

Before building the integration, confirm with customers that you haven't missed an important use case, and that your flow makes sense to them.

For example, consider an integration between Mindy's Marketing Automation and Wendy's Webinars. What is the marketer's workflow and how does she need these systems to interact?

In this case, a marketer might want to:

* Build a landing page in the marketing automation platform, have people register for the webinar on the page, and have the email, name, and company fields automatically uploaded as contacts in both systems

* Use the landing page in the webinar software and have the registrants automatically uploaded to the marketing automation software

* Send out invites from Wendy's Webinars from a list of contacts created in Mindy's Marketing Automation

* Have registrants removed from the Wendy's Webinars' list if they were blacklisted in Mindy's Marketing Automation

* Have whether a person attended the webinar, how long they watched, and whether they asked a question automatically uploaded as a new field into Mindy's Marketing Automation

What are user stories and acceptance criteria?

User stories are how product managers and developers often map how a user will interact with the product and other systems. Acceptance criteria lay out what needs to happen in order for the user story to occur.

When it comes to integrations, you should lay out how the user (or users) will want to use the integration and then ensure that the integration is built in a way that makes this possible by defining the exact functionality required for the user to walk through the whole story.

pandium
integrate all the things

User Stories

Sample user stories for a marketer integrating a marketing automation platform with a CRM.

As a marketer, I need to send my landing page leads to my sales people by having them entered in our CRM.

As a marketer, I need to send my rankings of my leads to my sales team by having the rankings entered into our CRM.

As a marketer, I need the sales team's intel on leads in my marketing automation platform so I an send out personalized and segmented emails.

As a marketer, I need to calculate the ROI on my marketing efforts by pulling in data from our CRM.

pandium
integrate all the things

Acceptance Criteria

Sample acceptance criteria for a marketer integrating a marketing automation platform with a CRM.

As a marketer, I need to send my landing page leads to my sales people by having them entered in our CRM.

- Automatically send my landing page leads to the CRM
- Removing a lead or its duplicate updates the CRM list
- Click to send different landing page leads to lists in the CRM

As a marketer, I need to send my rankings of my leads to my sales team by having the rankings entered into our CRM.

- Click and select the fields and matching rankings I want to automatically send to the CRM
- Rank a lead and have it automatically update in the CRM

As a marketer, I need the sales team's intel on leads in my marketing automation platform so I can send out personalized and segmented emails.

- Click and select the fields I want from sales to be added to my lists
- When a sales person adds new relevant intel, it automatically updates my lists
- Click and remove CRM fields

As a marketer, I need to calculate the ROI on my marketing efforts by pulling in data from our CRM.

- Click and select the fields from the numbers of the sales team's funnel to automatically come in
- Be able to tag deals in the sales funnel to marketing efforts

Once all your user stories and acceptance criteria are written, you are ready to work with your engineering team to get them built.

What other terms might I see around the configurations?

Request and response schema

The request schema is all the different requests the application can make to the resource, and the response schema is all the different replies the resource can deliver. It is important these align with the user stories identified.

Parameters

As covered, parameters include additional information and options that can be included with a request to one application.

Naming convention

This simply refers to how elements and data are labeled (for example, are data elements case sensitive? Is a customer a user or a customer?), and ensuring that the integration reconciles any discrepancies in the APIs or systems.

Methods, and resources

An integration does not need to offer the user every possible function and data field available in the APIs. The functions or methods refer to the actions that the application can perform - usually, the options include being able to retrieve, push, modify, or delete data.

Resources identify a particular category of data, and can be endless and are ultimately a reference to the underlying options in the application. Users, locations, patients, doctors, articles, for example, may all be resources.

The integration should utilize only the resources that the user needs to meet their business objectives.

pandium
integrate all the things

Why do the configurations matter?

Setting up the right configurations are vital to giving customers a good user experience, and enabling them to meet their business objectives.

If the options are too narrow, then users won't be able to use the integration. If there are too many choices, then users will be dissatisfied with the experience, and others will choose not to use the integration instead of trying to wade through an overly complicated setup process.

pandium
integrate all the things

QUESTIONS TO ASK ENGINEERS: Configurations

- [ ] **Did we consult with product, CS, and marketing on user stories? Did we ask customers for their feedback?**

1

- [ ] **Did we identify all the different user types and their workflows for this integration?**

2

- [ ] **What actions can the user perform with this integration? Do these actions meet all the acceptance criteria?**

3

- [ ] **Which calls are asynchronous and which are synchronous and why?**

4

- [ ] **Did we include additionally functionality or resource options in the integration that the user does not need?**

5

- [ ] **Have we QAed and bete tested this to ensure we aren’t missing any needed parameters or functionality?**

6

5. UI and UX

A key element of a successful product integrations is ensuring that customers have a good experience using them. This means an easy to understand and intuitive workflow and a user interface that is well-designed.

What is the UI of the product integration?

The UI of the integration is the interface that the user interacts with to install, run, modify, monitor, and disconnect the integration.

A front-end developer needs to design this UI. An ideal experience starts in an in-marketplace, but even if there is no marketplace, navigating the integration should be branded to your product.

The integration would also display the logo of the other application, and potentially adopt some of their branding materials in any content.

The UI might also sit in your partner's marketplace, in which case it will have to comply with their branding and rules, as well as reflect your own branding.

In some cases, the integration might live or have UI elements in both applications. The goal is to identify both brands where relevant, have it align visually with where it sits, and to seamlessly facilitate the user's business objectives.

As a user installs and uses the integration, the interface should adhere to the general standards of your product, and best design practices.

The newest elements of the UI for an integration might relate to the configuration setup, and any logging or monitoring the user has access to.

An important part of the UI is leaving space to include any relevant instructions that the user can follow to self-serve. This reduces the burden on customer support, and provides a better user experience.

What is the integration UX?

The integration UX is exactly what it sounds like - the user's experience of installing and using the integration.

Unless there are only a handful of product integrations, the best UX includes an in-app marketplace, where the user can easily discover and install integrations from one place.

They will be covered in more detail in the next chapter, but a user should be able to search and filter an in-app marketplace to find the integrations they would like. Once they click on the tile for a particular integration, they should be able to read about what exactly it does in a nicely designed user interface.

What are the important stages of a product integration UX?

Discoverability

This will be covered in more detail in the next chapter, but unless there are only a handful of product integrations, a user should be able to search and filter in an in-app marketplace to find the integrations they would like.

Understanding

Once they click on the tile for a particular integration, the user should be able to read about what exactly the partner app is, what the integration does, and what is required to install it.

Installation

The ideal installation UX is the user clicks install, is presented with a page to log into the other application, and is then offered the scopes for their approval. Once they click yes, the integration should be installed.

Monitoring

The user should have an easy way to see if the integration is connected, and what activity has occurred, including any failed syncs or errors. In addition, in most cases, the user should be able to run a sync.

Modifying

The user should be able to easily change the integration configurations they first chose.

Alerting

The user should quickly be alerted if their data failed to sync, either directly or because your customer support team was alerted and then alerted the customer.

Disconnecting

The user should be able to easily disconnect the integration, usually on an Integrations page or tab where all their integrations are listed. This might also sit where the integration is used in the product.

Why does the UI and UX matter?

Integrations often fail not because the user wouldn't find them valuable, but because they were too difficult to install or use. The UI and UX are both key to making sure that an organization sees good ROI on their investment in integrations.

Consider, for example, if the user is not altered when their integration disconnected and it is failing to sync their data (which can happen solely because the partner's API went down). Depending on how long that passes without the user knowing, any number of problems could accrue, from marketing emails that did not go out, to updates in employee salaries not being logged, to sales meetings missed.

Or consider if a user cannot figure out how to modify their configurations when they need to update their workflow.

Bad integration UX can lead to real frustration, and take what should have been a value add and transform it into a dissatisfied customer. While a well-designed UI and UX will increase brand loyalty and customer retention.

pandium
integrate all the things

QUESTIONS TO ASK ENGINEERS: UI and UX

- [ ] **Does the UI match our product branding and our partners' as makes sense given the UX?** 1
- [ ] **Does the user have an easy way to discover and then learn about the integration?** 2
- [ ] **Is the installation process seamlessly, or does the user have to toggle back and forth, find API keys or other identifiers, and change multiple settings?** 3
- [ ] **Can the user view their data syncs, and sync their data if they need to? Can they see any errors?** 4
- [ ] **Is the user quickly alerted if the integration fails? Can the user easily disconnect and reconnect the integration?** 5
- [ ] **Can the user find the integration in an integration section, or in a place that makes intuitive sense given its function?** 6

6. In-App Marketplaces

Depending on the company, an In-App Integration Marketplace might be called an Integration Center, Apps and Integrations, App Directory, or App Store.

No matter the name, it is a page within an application where users can search and browse the available integrations and install them.

In most marketplaces, there will be listings for integrations built by the company, and third party partners. They also might have widgets or extensions that are often features built on top of the product rather than true integrations.

Most companies also have a public or brochure webpage that mirrors the in-app page of the marketplace. This is so prospects and customers can browse the in-app marketplace without being logged into the app.

In-App Marketplaces allow users to discover new apps and new functionality.

They also give partners an incentive to build integrations, as once the integration is in the marketplace, they can get more leads and new customers for their own application.

Typically, a company will build an in-app marketplace when they have around a dozen product integrations, though this varies significantly.

pandium
integrate all the things

What are the important components of an in-app marketplace?

User Facing

Integration Tiles

Each tile has the logo of the application it is connecting to, sometimes with a brief description of the product. When it is clicked, it goes to a page explaining the integration and product.

Search

A user can search for a particular integration by name. More sophisticated search might search the content of the tiles.

Filter

The user might be able to filter by any of the following: product category, integration tier, user, industry, who built the integration, user reviews, popularity (number of installs), new, price, or featured.

Waitlist Feature

The user might be able to add their name to integrations that the company is building or thinking about building.

Outlinking

The marketplace might be able to link out to external pages for integrations that were built by third parties or need to be installed within a third party.

pandium
integrate all the things

Payment

The marketplace **might be set up** to accept payment and then pass the payment minus a percentage to the tech partner.

Installation, Modification, and Monitoring

The user might be able to install, modify, monitor, and disconnect their integrations on a page or section that shows active integrations.

Internal Facing

Admin Dashboard for Logging and Monitoring

An admin dashboard for logging and monitoring usage would give customer support visibility into any errors, and the usage of the integrations.

Business Analytics

The marketplace might be able to track who visited which tile, who tried to install the tile and failed, and who successfully installed it and how much they are using it.

Account Provisioning

Developers need a way to securely manage all the users and permissions for all the integrations.

Alerting

Developers need to design a way for customer support and IT to be alerted if there is an integration sync failure or disconnect.

pandium
integrate all the things

Hosting

Developers need to ensure the integration data remains available and is not delayed, without causing untenable costs.

Partner Facing

Developer Sandbox, and Resources

Larger marketplaces offer their partners' developers a sandbox to test out their integration builds on. Other marketplaces will have a clear process for approval, and documentation, as well as developer support.

Marketing Page

Marketplaces will have a process for partners to submit their marketing materials about an integration they built or are part of.

Business Analytics

The marketplace might chose to offer their partner analytics on who visited their tile, who tried to install the tile and failed, and who successfully installed it and how much they are using it.

Logging and Support

The marketplace might choose to offer their partner visibility into integration activity, or alert them when there is an issue.

pandium
integrate all the things

Why does an In-App Marketplace matter?

A well-designed and properly running in-app marketplace can pour fuel on a tech partnership program. The right setup can mean partners have a much bigger incentive to build integrations as the build process is easier, and being in the marketplace will provide valuable leads.

And customers have a better user experience with a marketplace, and are able to more easily find and discover integrations they would find useful.

Building a successful in-app marketplace requires devoting engineering resources to the front-end and the infrastructure. Once it is running, it can dramatically reduce the need for customer support, and can greatly increase the number of integrations that are built and used.

In addition, it can provide leads and referrals for partners who show interest in the integration, and the public page can provide warm leads for a company's marketing department.

pandium
integrate all the things

QUESTIONS TO ASK ENGINEERS: In-App Marketplace

- [ ] **Does the front end of our marketplace have places for business users to upload marketing content from our team and from our partners teams?** 1

- [ ] **Do users have an easy way to search and filter the marketplace listings?** 2

- [ ] **Does the marketplace have a waitlist feature, and outlinking?** 3

- [ ] **Do users have an easy way to install, modify, and disconnect their integrations? Are their active integrations in one place?** 4

- [ ] **Do we have an internal Admin Dashboard for business users to track usage and errors? Do we have alerting for integration failures?** 5

- [ ] **Do we have a secure way to provision new accounts at scale?** 6

pandium
integrate all the things

- [ ] **Do we have alerting set up so we are notified if an integration fails? How do we notify users and partners?** 7

- [ ] **Do we collect business analytics on clicks, failed installs, and usage for our team, and our partners?** 8

- [ ] **Can we offer partner developers a sandbox or any other resources to make building integrations into our product easier?** 9

- [ ] **Do we have clear technical standards of review for partners?** 10

- [ ] **Is our hosting set up to remain high performing and affordable even at scale?** 11

pandium
integrate all the things

Once product integrations and an in-app marketplace are built, they need to be monitored and maintained.

An integration can fail because your infrastructure went down, or your code had a bug, but it can also fail when your partner has an issue with their infrastructure or code.

Integrations can disconnect, fail to sync, or break. They can also cease to be useful if one of the products changes in a way that makes the original integration design useless for the new functionality.

In addition - and this depends heavily on how well they were designed, and what the UX was - users can fail to install or use the integrations properly, and the user can turn to your or your partners' team for support.

Once a product integration is built, what maintenance does it require?

Maintenance of integrations include: hosting the data, updating the configurations, updating the UI and documentation, and monitoring usage and errors and responding to any alerts and errors.

It is important to remember that APIs and products are regularly updated, and integrations need to be updated accordingly. Otherwise, customers will soon find the integration does not meet their business objectives.

In addition, depending on the requirements you established for entering your in-app marketplace, integrations that were built by third party might cease to be maintained or to work.

pandium
integrate all the things

What is logging?

Logging provides visibility and a record of the runs and usage. As an organization, you want to know who is using the integration, and how they are using it. This is important for customer support and for evaluating the success of the integration, and refining what your customers are looking for in integrations.

API logging is a record of activity for the API. It can record when a request and a response was made. Developers can set up logging in different ways, but the logs should be respectful of data privacy, and make it easy for customer support to understand the activity and see when an error has occured.

What is alerting?

In this context, alerting refers to your customer support or IT team being automatically notified when integrations are not working. You don't want to wait until hundreds or thousands of customers are complaining their integration went down to respond to an issue. Your developers can set up criteria that merit alerting customer support or the IT team.

What is API versioning?

APIs will inevitably change as the underlying product changes, the use cases change, and companies learn more about what their customers need integrations for.

Some changes can be made to an API without needing to release a new version of the API. For example, you can usually add a new endpoint to an API without it causing a breaking change. In this case, you'd simply want to document this change and communicate it to third party developers.

A “breaking change” is exactly what it sounds like - a change that means part of the old API will no longer work. For example, changing an existing field’s data type, or removing an endpoint would be a breaking change. An integration that was designed to move registrants from one application to another will no longer work if the ‘registrants’ endpoint is deleted.

When there is a breaking change, companies usually introduce a new version of the API. This allows third parties and customers to continue to use the old version if they want. Often, a company will communicate how long the older version of the API will be supported, giving third parties time to update their integrations to work with the new version.

You should build a way to track and manage API versions so that you can provide the proper support, and ensure your customers and partners do not end up trying to use a deprecated version of your API.

What is hosting?

Hosting is the costs associated with renting or reserving a computer that can run your code. An integration's data need to run in the cloud and that requires relying on a particular cloud infrastructure and paying the cloud provider's costs.

Depending on how the integration is designed and who built it, the costs and setup of hosting the integration and its calls will fall to you or your tech partner (or a customers who builds an integration).

Like with hosting generally, the objective is to keep the costs affordable at scale and minimize any downtime. Companies may limit or charge for API calls as a way to keep fast performance and high availability.

Especially if you do not have an effective alerting system, downtime on an integration can result in key business failures. As a result, it is important that whoever is hosting the integration ensures that it can perform at the appropriate scale without breaking the bank.

pandium
integrate all the things

What are API calls and limits?

An API call is what it sounds like - when the API is queried with a request. So if Mindy's Marketing Automation platform asks the Twitter API to return any tweets from @samjohnson that would be 1 call.

Most applications set up API call limits to avoid system overload, and to keep high availability for all users. As a result, developers must design and update any integration to stay within these limits.

You should always ensure that customers can use the integration as intended without going over the limits. You don't want to put your customer in a position where they expect to be able to sync their data and they can't because they go over the limit on calls. (Or in a position where someone is hit with a huge surcharge.)

The most common way calls are limited is by limiting how many calls you can make in a given time period, i.e. the total number of times you can make a request to the server in a given time period.

Here are some examples of the ways API calls are limited by applications:

* Limiting the number of **long calls running at the same time**

* Limiting the number of **requests per second**

* Limiting the number of **data points requested**

* Limiting the number of **failed requested** number of failed requests

* Limiting the **execution time of all requests**

Integrations should be designed to use the fewest number of calls as possible. For example, applications often have a Bulk API where large amounts of data can be retrieved in one call. If this is possible given the business objective, this option should be utilized to reduce the number of calls.

From a maintenance perspective, developers must ensure that the integration continues to hew to these limitations while still enabling users to meet their business objectives.

What is API polling?

API polling is essentially putting in a call to the API. Short polling calls the API on a set schedule, while long polling calls the server and waits for there to be relevant data before returning it to the requesting application.

Why should we keep abreast of product and API updates?

As covered, either you or your partner might have a “breaking change” in the API that means the integration as designed may cease to work.

But even if there is no breaking change, API and product updates might merit updating the integration. When an API and product adds new functionality, how and why the user is utilizing the product might change. If the integration ceases to meet business objectives, it will provide a bad user experience.

Staying abreast of significant API and product changes of your partners, and letting partners know about yours, helps to ensure integrations will remain useful to customers.

pandium
integrate all the things

Why does monitoring and maintenance matter?

No matter how well-designed your initial integrations and marketplace, they have to be maintained and monitored to continue to meet the users' needs.

Because product integrations involve third parties (and at scale can involve dozens or even hundreds of third parties) it is vital to set up logging and alerting so your team is immediately aware when third party technologies cause errors that affect your customers.

In addition, ensuring your team has a systematic way to stay abreast of API and product changes of your partner (as well as your own product changes). These changes can break integrations, or render them useless even when they remain functional.

And however the integrations and calls are hosted, you should ensure they offer reliability and performance without huge spikes in costs.

pandium
integrate all the things

QUESTIONS TO ASK ENGINEERS: Maintenance and Monitoring

- [ ] **Do we have logging and alerting on our integrations for our customer support and IT teams?** 1

- [ ] **Do we have a process for staying abreast of API changes and product changes of our partners?** 2

- [ ] **Do we have a system for notifying third party developers when we make relevant changes to our product or API?** 3

- [ ] **Do we have limits on third parties calling our API?** 4

- [ ] **Do our integrations serve their business purpose while staying within any API limits of our tech partners?** 5

- [ ] **Does our hosting plan for integrations ensure high performance and low latency at scale?** 6

pandium
integrate all the things

Index

Page numbers are links.

pandium
integrate all the things

Index

Page numbers are links.