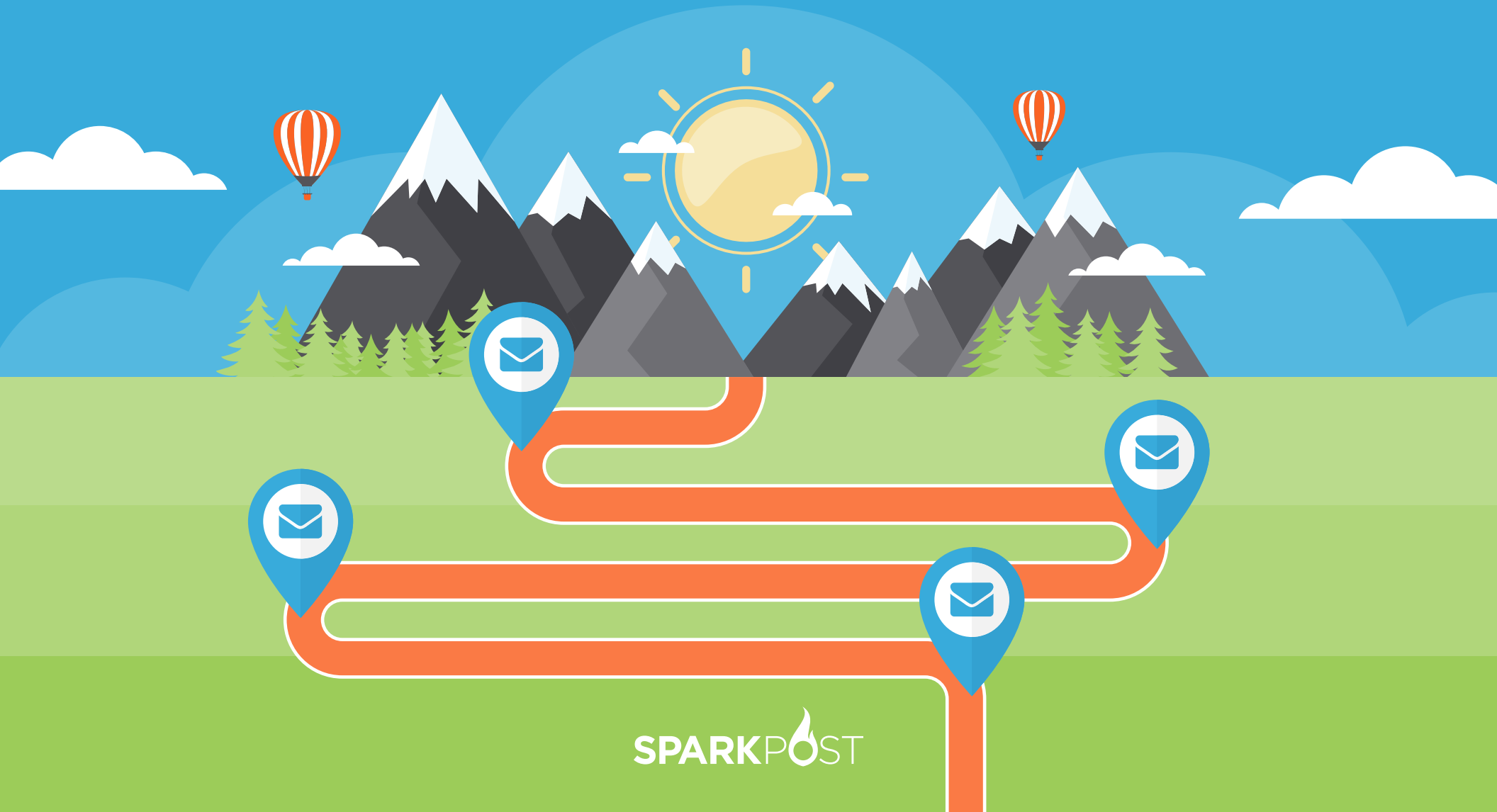
The Product Manager's Guide to Email

How to Build a Successful, Growing App with Email Notifications, Alerts, Transactional Messages, and Other Product Emails



SPARKPOST

1 Growing Your App with Email

Email is part of daily life that's easy to take for granted. In many ways, it's the backbone for all the information we receive.

Many of those emails are not interpersonal communication and correspondence, or even marketing messages. They're e-commerce receipts, event or airline tickets, social network notifications, and alerts from business apps. Mailboxes have become our de facto system of record, and one that continues to grow in importance.

As users, we appreciate how email notifications from SaaS apps and services help us keep tabs on our workflows and give us confidence in the security of our accounts.

For product development teams, the messages their apps send are an indispensable tool for generating user interest, building trust, and nurturing long-term engagement. These product emails have a dramatic impact on customer conversion and retention.

The email experts at SparkPost created this guide to email notifications and other app-generated email messages to help teams building fast-growing services make the most of email in their products. It's vital reading for product management and development teams and anyone else looking for effective ways to engage customers and build a growing app.

In this guide, you'll discover:

- Key use cases and the kinds of emails users want to receive—and how they can be used to increase conversions, generate engagement, and reinforce trust and confidence in your app.
- How product teams can measure email performance and impact.
- The right way to implement emails for your app, as well as the technical challenges and risks you'll want to avoid.

Product emails and notifications have an outsized impact on engagement, conversion, and retention. Let's dig in.

2 Does My App Need Email? Yes—And Here's Why

Is your app sending email? If it's not, it should be.

Let's be clear what we're talking about: messages generated and sent by an application or web site in response to specific user action or other event. Common examples of these app-generated emails include activation and welcome messages, onboarding prompts, activity reminders, receipts and shipping notifications, account and security alerts, and even utilitarian functions like password resets.

These notifications serve an important purpose—alerting us when a post was shared on social media, reminding us to take action on a personal account, or asking us to approve payment for goods and services.

Beyond these purely functional needs, notifications also are a valuable (and often untapped) communication tool that enables product teams to directly engage with their customers. They're one of the most influential tools that product managers have to drive conversion, retention, and growth.

Consider some of the benefits to your app:

- Emails are a persuasive instrument for drawing users back to using apps that they might have forgotten about.
- Carefully designed, timed, and implemented notifications help deliver a great user experience.
- Emails reinforce trust in services and help to build long-lasting relationships between a SaaS business and its customers.

Let's look at the situations where these emails will deliver the most impact.

Key use cases for email in SaaS apps

Product emails are as diverse as the products that send them. The messages sent by a social network obviously will be very different from those sent by a service that helps airlines manage their fleets of aircraft, maintenance schedules, and work orders.

Nonetheless, there are several types of emails that nearly every SaaS product should send, customizing their functionality, style, and tone to the needs of the business and its users. **These key product email use cases include:**

Activation emails are sent as soon as a new user creates an account. It's the first email notification your users will receive, and is a critical path step towards user activation. For many services, users very literally cannot use the app until an account has been activated by clicking a link in the email.

Activation emails serve to verify that the email address the user provided is valid and working. They also remind users that they chose to sign-up for your product, an important step in making the difference between a genuine user and a "drive-by" signup.

Welcome and onboarding messages are sent once a user has verified their email address to activate a new account. Most often, a welcome message is sent as soon as an account has been activated. Welcome messages reinforce a service's core promises, set the tone for future interactions, and say thanks for joining.

One or more onboarding messages are sent to help users explore and get started with the product. Whether timed or triggered by specific actions, these emails provide information and prompt users to get up and running—and to accelerate their pace to become active, engaged users.

User invites and shares are are a core email for consumer and B2B apps alike. Both contribute to viral, word-of-mouth growth. Explicit invitations asking colleagues to join a project team are essential for many categories of services and are a notification example that product teams must consider implementing.

Although shares are very often associated with social networks and similar consumer services, they also have a place in many B2B contexts. Document sharing and other forms collaboration are common examples. It is worth noting that shares also play the role of an implicit invitation to join when sent to a new user.

Activity reminders and notifications are an important tool for alerting users to changes in workflows and for integrating in-app activity with many users' primary environment, their email inbox. They also are an effective way of reengaging passive users who might not be using the app on a regular basis.

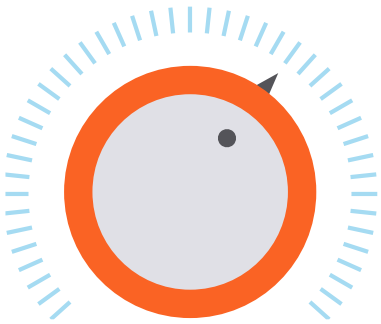
These messages can be triggered by explicit events or sent on a scheduled basis, depending upon the context. Typical examples include reminders to complete a task or a summary of missed social media notifications.

Reports and dashboards are especially important for B2B services. The information they contain is essential for users and their teams to keep tabs on the business processes they manage.

Like other forms of activity notifications, these messages reinforce engagement for users who may not be active in the app itself. And well-designed reports and dashboards are an effective way of explicitly reminding users and their executive decision-makers of a service's value.

Password resets and two-factor authentication are examples of utilitarian notifications that every app must implement reliably. Like activation emails, they're quite literally the sort of product email can make or break a user's ability to use a service.

Security and account alerts are an essential aspect of helping users to protect their accounts. They're a crucial bulwark against fraud, misuse, and the theft of sensitive information and credentials.



Where emails move the needle

Users value notifications when they're relevant, but quickly learn to ignore, delete, or even flag as spam messages that are repetitive or don't add value to how they interact with an app.

But that still leaves a question for product teams. Deciding where to begin with product emails can be daunting. It's important to make strategic choices about which to send, and how often.

Identifying your product's key "make-or-break" moment are powerful places to begin. Make-or-break moments are the junctures where user activity (or lack thereof) can mean the difference between success and failure. They're also where product teams have an opportunity exert the most dramatic leverage to improve the user experience and directly improve core business metrics.

The right email notification at these moments can help keep a user engaged and productive. On the other hand, a missing or ineffective email may well result in a missed opportunity and even customer churn.

Consider two very literal examples of this sort of critical message: activation emails and password resets. Both require direct, immediate action from a user to continue their usage of an app. But if the email is not effective or is delayed or even lost, the consequence may well be a permanent loss of that user.

Beyond this sort of blunt scenario, there may be more subtle moments that have outsized impact on your users' success. Starting with the common use cases we described earlier, identifying and addressing them is crucial to your success. They're ideal candidates for the right sort of product email.

So, what are yours?

3 It Takes More Than a Send Button to Deliver App Email

We've seen how crucial emails like onboarding messages, email notifications, security alerts, and other product emails are to the success of SaaS applications. But when emails are ineffective, lost, or delayed, the result may well be user churn.

Sending email from an app isn't quite like opening Gmail and pressing the send button.

In fact, operating infrastructure for generating, sending, and measuring email is a surprisingly complex challenge—especially at the scale and performance a SaaS product requires.

Why delivering SaaS email is challenging

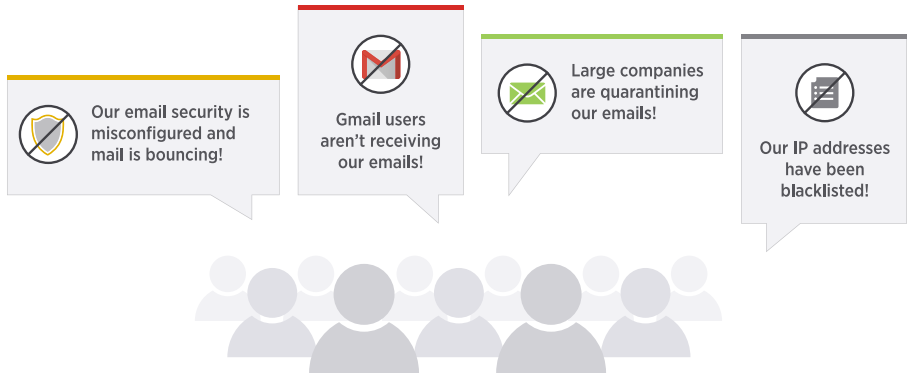
In its basic form, email seems so simple; sending an email message to your user takes nothing more than pushing "send" (or making the equivalent call in the programming language of your choice).

For SaaS email notifications, your team's developer might assemble a few lines of a message in their code, add a subject line and a sender address, and call the send function. The message shows up in their inbox when they test it, it looks like what they assembled, and they move on to the next part of their code.

The difficult challenges come when things start to scale. What the developer didn't realize is that while mail delivery is a trivial task when sending one message here or there, the Internet service providers (ISPs) who host most consumer mailboxes—services such as Gmail, Microsoft, and AOL—become much more picky when you send thousands of messages at a time. In order to protect their customers from spam, they'll often divert unusual, high-volume bursts of traffic from the inbox to the spam folder.

So consider what happens when a SaaS product begins to grow. The volume of email notifications suddenly increases, and ISPs react by slowing or even blocking delivery of the new messages. When those key alerts such as welcome emails or account verifications never arrive in the inbox, the user experience suffers and introduces roadblocks to continued growth.

Fortunately, there's a better way. Read on to learn how.



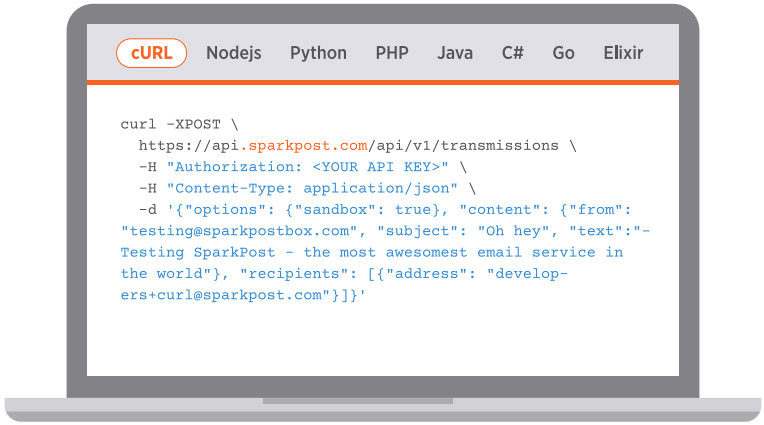
Why an email API?

An email API gives apps access to the functionality available in an email platform, such as generating and sending transactional emails, manipulating templates, and enabling access to email metrics.

API-generated email is an easy fit for how most product and development teams already work. That alone might be reason enough for you to choose this approach, but there are additional reasons as well. In particular, sending email at volume and with the performance characteristics SaaS products require is a challenge most teams shouldn't take on internally.

Traditional email infrastructure doesn't scale well, and as we've discussed, managing issues with ISP mailbox providers like Gmail and Outlook is a specialized expertise for which very few teams are prepared. Troubleshooting email server uptime and delivery headaches is a distracting and expensive task for engineering and ops teams to take on.

That's why most teams building SaaS applications implement their email notifications and other messages by relying a cloud email API.



Using an email API to send email from your app

Email notifications and transactional emails are a perfect use case for an email API. In general, the process of generating and sending a notification or transactional email with an email API looks something like this:

A user takes an action or some other event occurs. Perhaps it's something the user did explicitly like creating an account or resetting a password. Or it could be triggered indirectly, such as a change in a workflow process or a summary of activity by other users.

In either case, you want the user to take action to accomplish a task or reengage with your app. You've specified that an email be sent with relevant information to help the user accomplish that goal. **The process looks something like this:**

1. Your app calls the email API and provides such information as the customer email address, the message content, and other details.

2. The email service creates a message with that info, using a template that has already been established for that specific purpose. (For example, you probably want an onboarding email to look and read differently from a password reset email, given what a customer likely expects in each situation.)

3. The email service transmits the message, negotiating the protocols required to ensure the email is delivered into the customer's inbox.

4. The email service captures data about the delivery of the message, if it was opened, if the recipient clicked any of the links in the email, and more.

There's no email expertise required—your team can continue to focus on building your app, and you can work with the results and data to understand and drive your user engagement.

There's more to email than sending

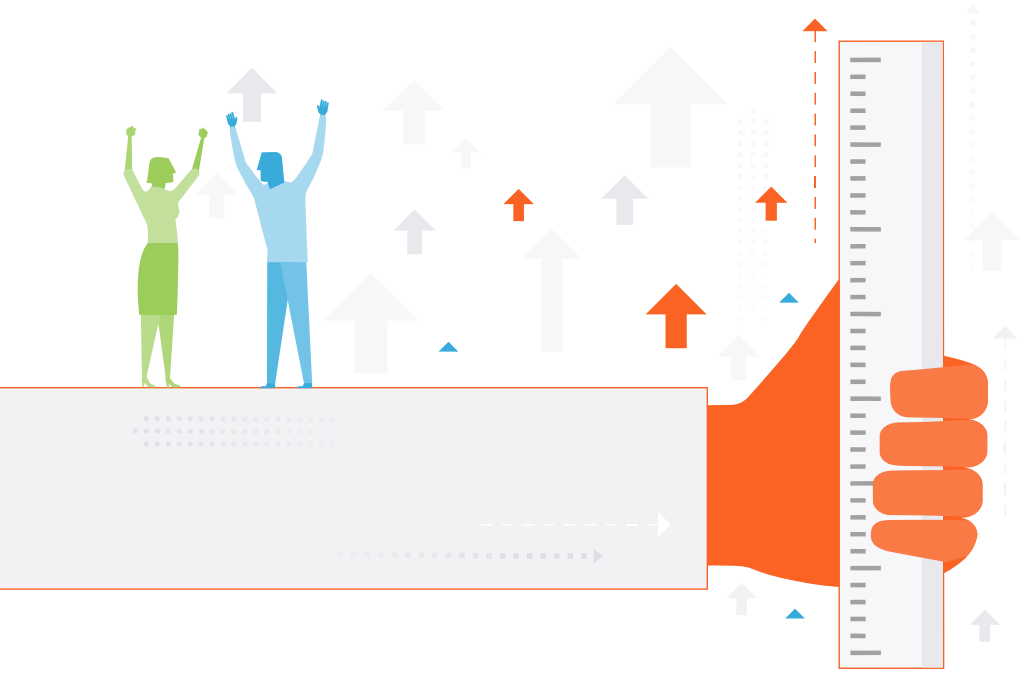
In fact, that data-driven quality is a really important part of working with an email API. You'll want access to a dashboard for summary reporting, but programmatic access to important metrics is just as important.

Understanding the data generated by email engagement is a crucial part of ensuring messages like email notifications are effective. Read on to learn how to measure the performance of product emails like these.

4 Are Your App's Emails Working?

We've discussed several key use cases for product emails. But simply sending those isn't enough—knowing how users interact with them is a crucial. As we've learned, when product emails aren't effective, an app's users are likely to churn.

So, accurately measuring how notifications affect an app's performance is a must-do. Unfortunately, it's all too common for product teams to have limited visibility into what happens once a message is sent. Data about something so critical to your product's success shouldn't be left to chance.



Assessing the impact of product emails should be considered with two lenses:

- Did the email immediately drive a specific, desired action?
- Did the email contribute to user engagement, conversion and retention over time?

These questions both are important, but they require different sorts of data and analysis. Fortunately, email is both measurable and flexible.

Basic product email metrics

Email is usually rich with data. Consider, for example, how many messages were delivered, rejected, opened, read, and clicked. With well-designed email infrastructure, these and many more metrics can be retrieved on a per-message basis, reported in aggregate analytics, or even sent in a structured data stream for more sophisticated analysis.

There are dozens of notification data points product teams can track (the SparkPost email API offers more than 35 individual metrics). **Some of the most fundamental metrics for product emails include:**

	Comments	Goals for Email Notifications
Bounces	High bounce rates are a red flag to ISPs and have a significant effect on how they treat your messages.	1%
Latency	What matters most is your user's context: did the message arrive when they expected it and could act upon it?	< 1 minute
Inbox placement	Each email that doesn't reach the inbox is a risk factor for customer disengagement and churn.	98%
Open rate	If you're seeing fewer than 1 in 3 of your notifications opened, then you're risking user fatigue by sending unwanted or unengaging messages. Dial it back and think about how to improve them.	> 50%
Click-to-open rate (CTO)	Ideal CTO rates vary. They depend upon the nature of the notification (e.g., is it informational or are you seeking to drive action)? But if the rate is consistently low, it's time to rethink the content and call to action of your notifications.	40%
Unsubscribes	Some of your users will want to opt out of email notifications. Don't fight it—make it easy for your users to take control.	< 10%

With the right data, your teams can capture and monitor both technical performance (i.e., how well the email infrastructure is operating) and engagement results (i.e., how users respond to the message). And you can act on that data to improve app engagement and performance.

Understanding product email performance

Let's look at a typical SaaS application email. What's the basic reason your app is sending an email? It's because your user has done something, and you need them to take an action to complete the task.

Account signup is a common example:

1. A user signs up for an account
2. You need the user to confirm her or his email address to complete the signup and activate the account

It seems a straightforward task. Certainly, it it is among the make-or-break moments we described earlier, and making sure that the email driving it works is essential to helping your user succeed… not churn.

But the journey this message must traverse isn't always as simple as it seems. **Consider the multiple checkpoints at which something could go off the rails:**

- **Triggering event.** Something happened in your app or an external service to trigger a notification event. Did you capture that moment and uniquely identify it for future analytics?
- **Generation.** Did your app or a dedicated service (sometimes called an "email API") create a properly formed message, with correct and personalized content? Does the message contain the unique identifiers you'll need to connect subsequent actions back to the original event?
- **Submission.** At the stage, the message officially has become an email and placed in the care of a mail transfer agent (MTA)—more colloquially known as a mail server. An MTA well-architected for the needs of product emails—that is to say, one that's designed for API-driven email—will introduce almost no latency to the process, but mail servers designed for passing messages from traditional email clients or marketing lists very often queue messages for later transmission. Depending upon the load, that delay might be substantial.

- **Transmission.** The actual sending of an email can be affected by network conditions, email infrastructure capacity, and a variety of other technical factors. But the most visible issue for most of us is whether the message "bounced," that is to say was rejected by the destination mail server, or was accepted. Bounces happen for a variety of reasons, but among the most common is an invalid email address.

- **Delivery to the inbox.** Not bouncing doesn't necessarily mean it's sitting in your user's inbox. Most email providers like Gmail or Outlook run additional, proprietary algorithmic checks to predict how recipients react to your message before deciding if it deserves to be in the inbox, some other folder or tab, or even the spam folder.

- **Open.** Did your user open the message? Details like the wording of a subject line or the return address can make a major difference in whether a user responds, ignores—or even flags a message as spam. Empirical approaches such as A/B testing are the best way to optimize this engagement.

- **Click.** Once opened, did the user take action? A direct click on a properly tagged link will make that connection explicit. As with subject lines, testing of message body design, images, content, and calls to action are essential.

- **Resolution.** Did the user actually complete the task at hand? That's what really matters. A closed-loop connection from initial triggering event to resulting action provides product teams definitive information about the effectiveness of their notifications.

Where to improve performance

The wealth of data provided by email APIs and similar infrastructure provides visibility into a product email's journey. It also helps product teams to identify where to optimize its performance.

- **Review how messages are assembled.** Closely monitoring the performance of your code is a good way to improve the latency of your message assembly and submission system.

- **Choosing performant email infrastructure.** Additionally, using an email API system can boost your performance. By offloading the message generation from your own systems, this method can help to reduce delivery time and maximize your efficiency.

- **Focus on user actions.** Did the notification drive the user actions that a product team needs, specifically improving engagement? By monitoring which actions a user has performed, product teams can gauge which are most attractive to users and adjust their strategy accordingly.

- **Look at email technical best practices.** To ensure that an email actually makes it to an inbox, product teams should focus on a checklist of items such as sender reputation, IP address warming, feedback loops and subject lines.

5 Keeping Your Users Safe

You already know how essential robust security is for your SaaS application. Your team has followed security best practices: training, code reviews, third-party penetration testing, secured your infrastructure, certification, and more.

And yet your efforts mean little if your user's credentials are stolen. One common approach bad guys take is to impersonate your email: a phishing attack. A malicious third party sends your users an email that looks like it came from your application. A link in the email takes users to a website resembling your own and requests their username and password to log in. User information like passwords or account numbers can be stolen—and attackers have access to your application.

It's clear that phishing attacks and similar abuses of your app could result in customers losing trust in your service.

What can you do to combat this? It's a complex challenge, but user education is certainly part of the answer; by following best practices for security notifications, you'll help your users be alert to unusual situations. In addition, properly implementing technical standards that help ensure your users never receive faked notifications is an important part of the solution.

Let's look at each in turn.

Best practices for security notifications

As with so many aspects of user engagement, user attitudes and behaviors are critical to the security of a SaaS product. Security alerts and notifications are one important part of developing and reinforcing user trust.

To be effective, product teams developing security notifications for their apps should consider how their alerts can (1) help users make good decisions about security-related issues and (2) convey the information users need to have confidence in the product or service.

Here are five best practices to consider when developing effective security alerts and notifications.

Give users clear, helpful information. When it comes to security alerts, don't beat around the bush. Be direct with your messaging and clear about why the alert was sent—avoid explicit marketing or brand messaging. The last thing a user needs is to feel like they're being sent an advertisement when they're worried about a sensitive subject. And be sure to include links to relevant support resources on your web site.

Quickly notify users about unusual account activity. You've probably used services that add extra layers of security that go beyond a simple login screen when a user logs in from a new device or location—perhaps you've been asked for a second factor or verification code. And even when a login is successful, many high-risk services (such as social networks) will immediately sends post-login emails to all addresses associated with the account, along with in-app notifications. That's a practice more SaaS apps should emulate.

Take extra care with password resets. What can you do to ensure password resets aren't compromised? First, don't allow password resets without secondary confirmation in the form of an email or other alert. It's not foolproof, but that extra step can eliminate one group of lazy attackers.

However, that practice is one reason password resets and similar alerts are a prime target for phishing attacks. It's here that technical best practices such as email authentication (see below) are essential. Certain design considerations, such as providing clearly recognizable URLs can also reduce the risk.

Be mindful of actions that make users nervous. Even the savviest Internet users can hesitate when making a change to how they conduct business online. For example, if your service offers a new option that will make transactions more convenient for customers, they may still be a little nervous about the implications of that change.

Providing transparency about what's being asked of the user, giving them advance notice well before the actual change, and providing in-app explanations can all go a long ways towards reducing apprehension about change.

Treat security as an ongoing part of the user experience. Onboarding messages and product updates that highlight security features can be helpful to simply let users know you're proactive about this important area of concern. They help reinforce the overall trust a user has for a service.

Email authentication basics

In addition to effective alerts and notifications, SaaS teams should implement key email authentication standards to help keep users safe.

Email authentication is a technical solution to proving that an email is not forged. In other words, it provides a way to verify that an email comes from who it claims to be from. Email authentication is most often used to block harmful or fraudulent uses of email such as phishing and spam.

There are several different approaches to email authentication, each with its own advantages and disadvantages. **Although the specific technical implementation varies from approach to approach, in general, the process works something like this:**

1. A business or organization that sends email establishes a policy that defines the rules by which email from its domain name can be authenticated.

2. The email sender configures its mail servers and other technical infrastructure to implement and publish these rules.

3. A mail server that receives email authenticates the messages it receives by checking details about an incoming email message against the rules defined by the domain owner.

4. The receiving mail server acts upon the results of this authentication to deliver, flag, or even reject the message.

As these steps make clear, in order for this process to work, the sender and the receiver both must participate. That's why technical standards for email authentication are so important: they define a common approach to defining the rules for email authentication that a SaaS provider (or any other organization) can implement.

Properly configuring email authentication standards like SPF, DKIM, and DMARC is one of the most important steps you can take to safeguard your app's reputation. (You can learn more about these important standards on our web site.)

Taking proper email authentication measures also have another benefit: they can help improve your ability to get notifications and other critical product emails to your users' inboxes. That's because email authentication can help make it more likely that the IP addresses and sending domains for your email will be trusted by receiving mail servers.

Email authentication protects your SaaS product's brand and domain reputation from spammers and spoofers. It also improves the likelihood your users will see the messages your app sends. That's a win-win for you and for your users.

6 Meet Your Email Team

The emails your app sends—notifications and the like—are a core responsibility of SaaS product teams. These emails are an extension of the product and should be treated as such.

In the case of app-generated email notifications, you'll find your team looks a lot like a classic, three-legged stool. In fact, these roles probably already make up the core of your product team.

Product Manager. That's you! It's your job to define the requirements for a product email team for a simple reason: the most successful SaaS businesses treat email notifications as a core product feature.

Larger, complex products and platforms may well have a product manager dedicated to developing a shared app notification service that includes email alongside other forms of communication like SMS or push.

Technical Lead. Perhaps a developer or an architect, the tech lead works with the product manager to design and build the services and technical foundation that will trigger, generate, and process data from email notifications.

Working with a team of developers, a notifications teach lead's core responsibilities include ensuring that reliability, scalability, and perhaps security considerations for product emails are addressed.

Growth or Product Marketer. As with other aspects of building a growing SaaS product, growth and product marketers increasingly work closely with product leadership to optimize user engagement. These marketers can help fine-tune the messaging, design, and pace of app emails. They should also use rich data from the emails to better understand and nurture user actions and engagement.

If you're building notifications with the help of an email API, these roles might provide the skillsets you need to get started.

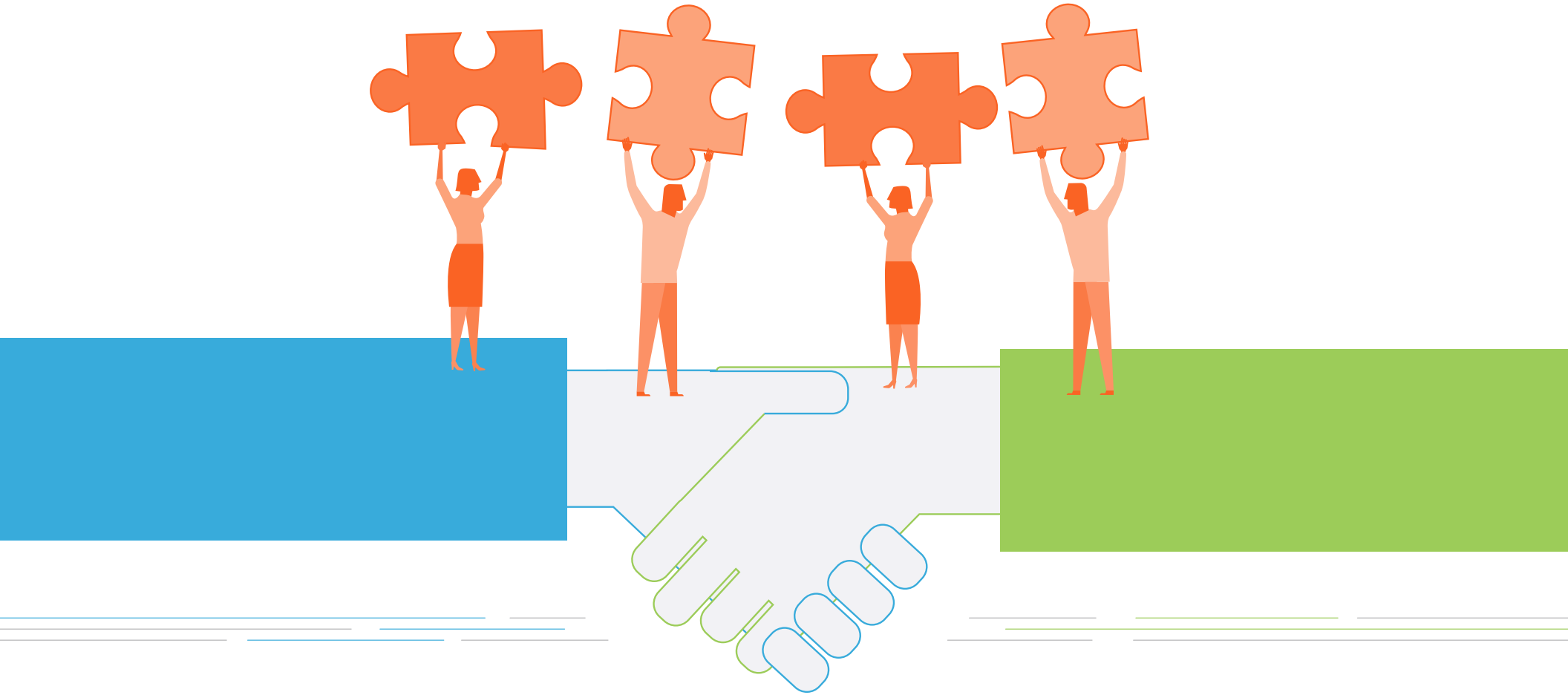
Some larger-scale services that send a lot of email probably need to fill additional roles. And if your business still operates its own infrastructure and email servers, you'll definitely need to think about the extra hands you'll need on deck.

Infrastructure Admin, Ops, and Security. Email servers are especially susceptible to scalability, security, and downtime challenges—even more so when dealing with the performance characteristics needed for product email notifications.

That's why a product email team understands the trade-offs and make an informed choice about managing infrastructure themselves versus relying on an email API. It also goes without saying that a good DevOps and security team will help you ensure your apps' emails aren't putting you at risk of phishing and fraud.

Deliverability Manager. Email deliverability managers are specialized professionals who understand the ins and outs of working with email and mailbox providers like Gmail. Deliverability pros are great resources to help you plan the implementation of an email notifications strategy—and what it takes to get email to the inbox on time.

In fact, if you're looking for help getting started, a good email delivery service will have great email deliverability pros to provide advice and extra help at the get-go.



7 It's Time to Get Started

As a product leader, you know how crucial user engagement is to driving conversion and retention and reducing churn. We've shown how onboarding messages educate users and help them get started using apps. Notifications and alerts earn attention and drive action. Account and security alerts build confidence and trust.

But email to deliver on those promises, product teams must approach email as a strategic initiative that's fully integrated into a product plan, not an afterthought. It requires care in finding use cases with maximum leverage and impact. It needs a performant infrastructure that's suited to the unique requirements of app-generated email. It entails ensuring that technical configuration and security needs are properly addressed. And it works best when a product team has the support it needs.

We hope this guide has helped you think about how effective emails are for supporting a great user experience and helping to drive the sort of engagement that will help you deliver a growing, successful application and business.

Learn More and Build a Better App with Email

Further questions about SaaS and email? Visit the SparkPost Academy, with resources that go deep on email technical best practices, user engagement, cloud development practices, and customer growth. It's valuable information that'll help you build a great product and your career. Dig in at **sparkpost.com/academy**

About SparkPost

Development and product teams want to build great apps, not manage email servers. SparkPost's email API delivers fast, flexible email and analytics integration for any application or website. SparkPost delivers your application's emails on time and to the inbox.

SparkPost is the most performant email delivery service—whether your app or web site sends hundreds, or billions, of messages. Our customers send over 3 trillion messages a year—more than 25 percent of the world's non-spam email.

Talk with our team **@SparkPost** or
go to **sparkpost.com** to learn more.

SPARKPOST

SparkPost.com • 301 Howard St., Suite 1330 • San Francisco, CA 94105 • tel +1 415-578-5222 • toll free usa 877-887-3031