BROUGHT TO YOU
BY SVIX

The State of Webhooks

2023

Table of Contents

CONTENTS

INTRODUCTION

This report presents research on webhook implementations. The goal is to show which best practices are adopted, how often they are adopted, and how those best practices impact actual webhook deliveries in practice.

100 API providers[1] were examined as well as internal data at Svix where we deliver billions of webhooks on behalf of our customers.

Currently, the adoption rate of webhooks is accelerating but implementations are still fragmented, with each provider reinventing the wheel.

This report quantifies how different webhook implementations are from one product to another, and how often best practices are followed.

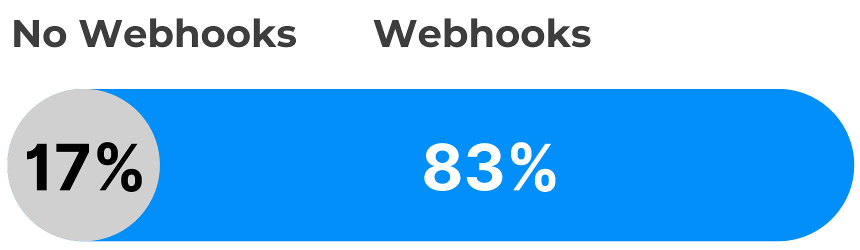
Hopefully this report will be a catalyst for the adoption of webhook best practices and improve the developer experience around webhooks across the industry.

IMPLEMENTATION

This section is based on the analysis of 100 API providers and their webhook documentation. The analysis focuses on their adoption of webhooks and webhooks best practices.

WEBHOOK ADOPTION

83% of the APIs researched offer a webhook service. This clearly demonstrates high adoption of webhooks.

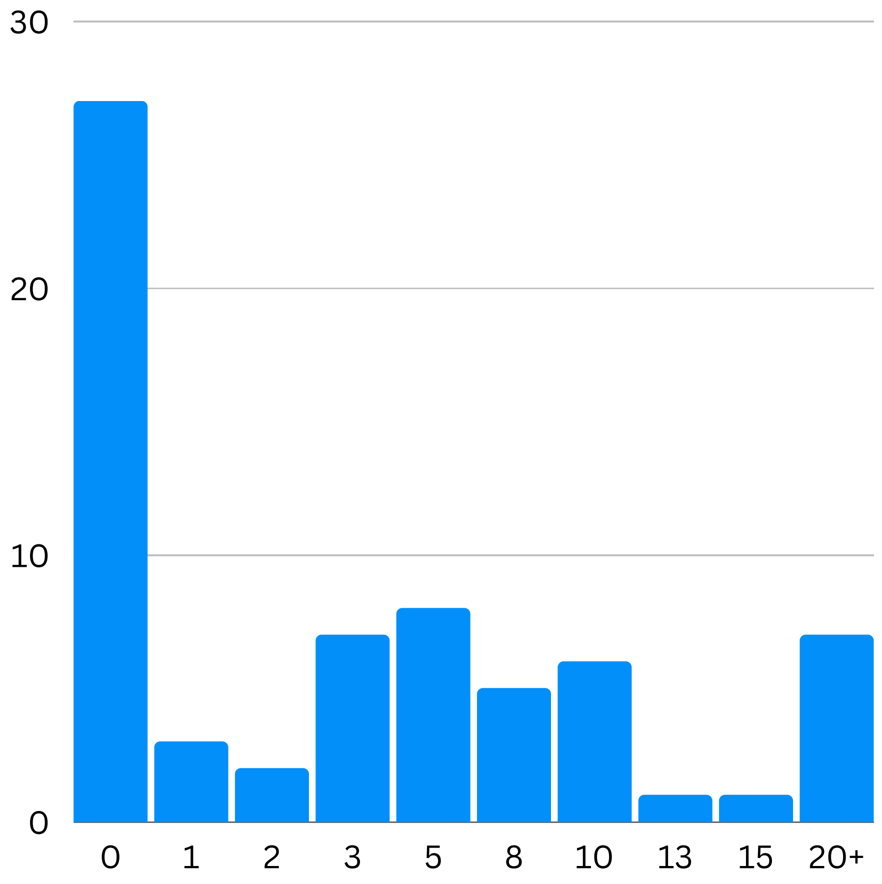


RETRIES

Retries involve re-sending a webhook if an attempt fails. They are crucial for a reliable webhook service because temporary network issues, server downtimes, or other transient errors can impede immediate data delivery. An excellent example of a retry policy is in Dwolla's webhook docs[2].

67% of services offered automatic retries. The most common amount of retries offered is 5 with most offering between 3-10 retries. Around 10% of the services stated they retried failed messages, but did not provide any information about the retry schedule itself.

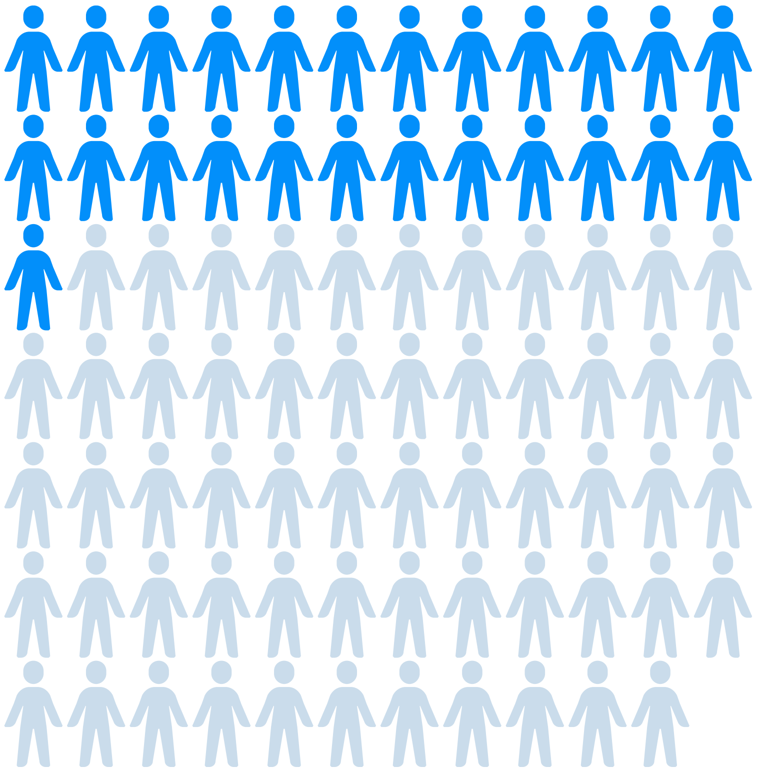
67



RETRIES: EXPONENTIAL BACKOFF

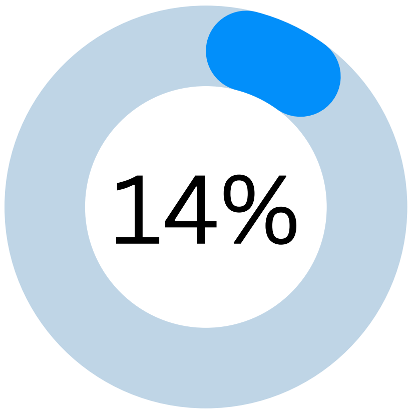
Webhook retries use exponential backoff to efficiently handle failures without overwhelming the receiving server.

By progressively increasing the wait time between retries, it reduces the risk of exacerbating potential server issues and provides a more adaptive approach to handling transient failures.



25/83 providers specified that their retry schedule follows an exponential backoff.

MANUAL RETRIES



Being able to retry messages manually speeds up troubleshooting. It is faster to trigger a retry immediately instead of waiting for the next automatic retry.

12/83 providers specified that retries could be triggered manually. This was the least adopted best practice.

LOGGING

For testing, troubleshooting, and debugging purposes, a log of webhook delivery attempts is extremely useful.

This was the second least adopted functionality at 23% adoption.

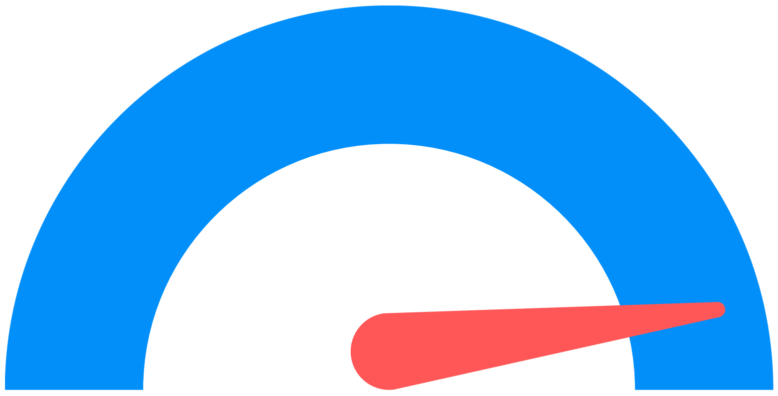
23

EVENT TYPES

Organizing the events offered to webhook consumers into event types lets users choose which events they want to receive webhooks for and cuts down on the number of unnecessary and irrelevant messages being sent. See Cloudinary's event catalog[3] for an example.

93% of providers offered event types. This is the most widely adopted best practice which probably stems from the fact that events are the core value of webhooks.

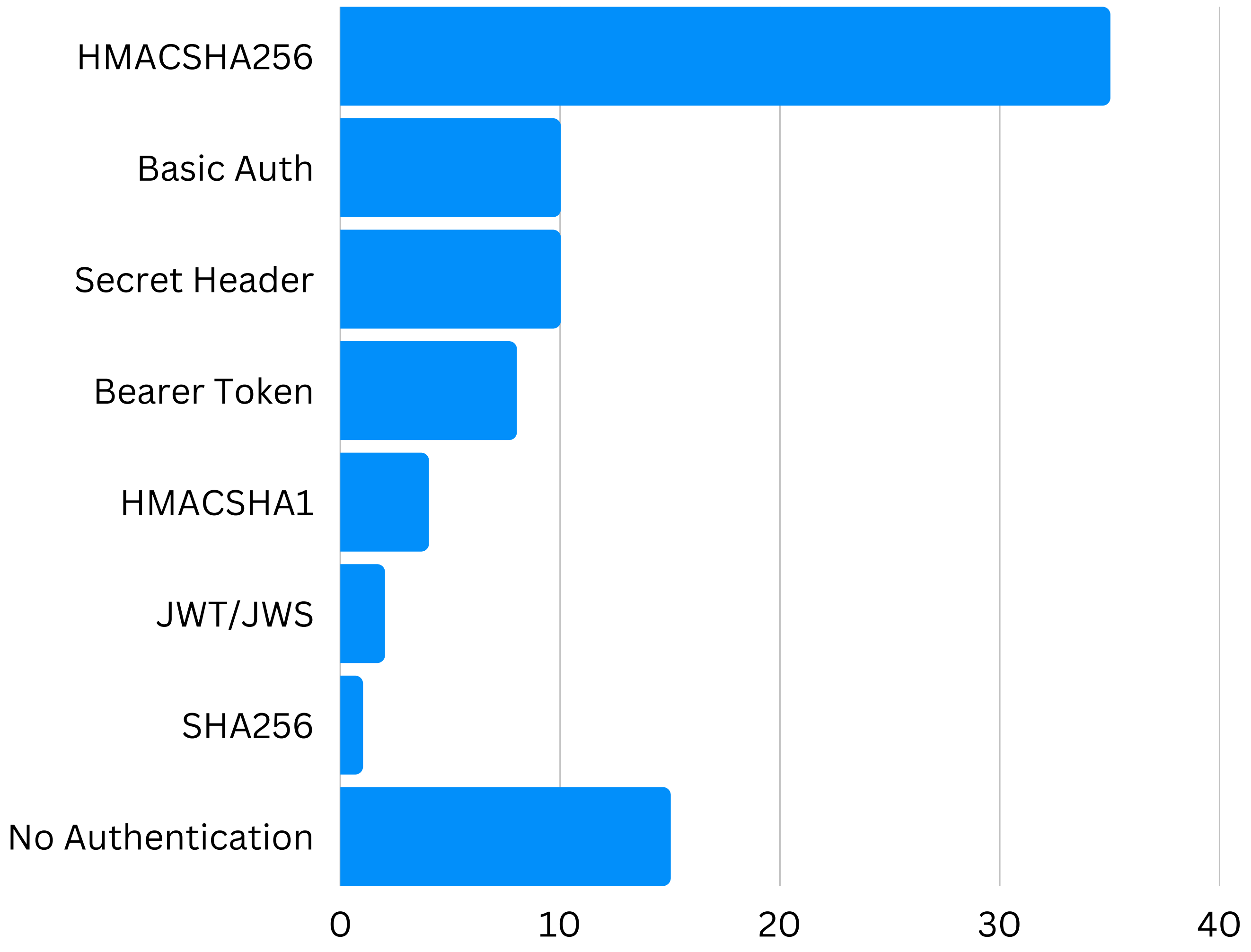
93



MESSAGE AUTHENTICATION

Giving users a way to authenticate the origin and content of a webhook message is essential for ensuring that data has not been tampered with during transit and confirms it originates from a trusted source.

The best practice is to use HMACSHA256 signatures that has the payload, timestamp, and message ID. See Zoom's docs[4] for a good example.



45

45 of 83 webhook providers included a timestamp. The timestamp is critical for preventing replay attacks

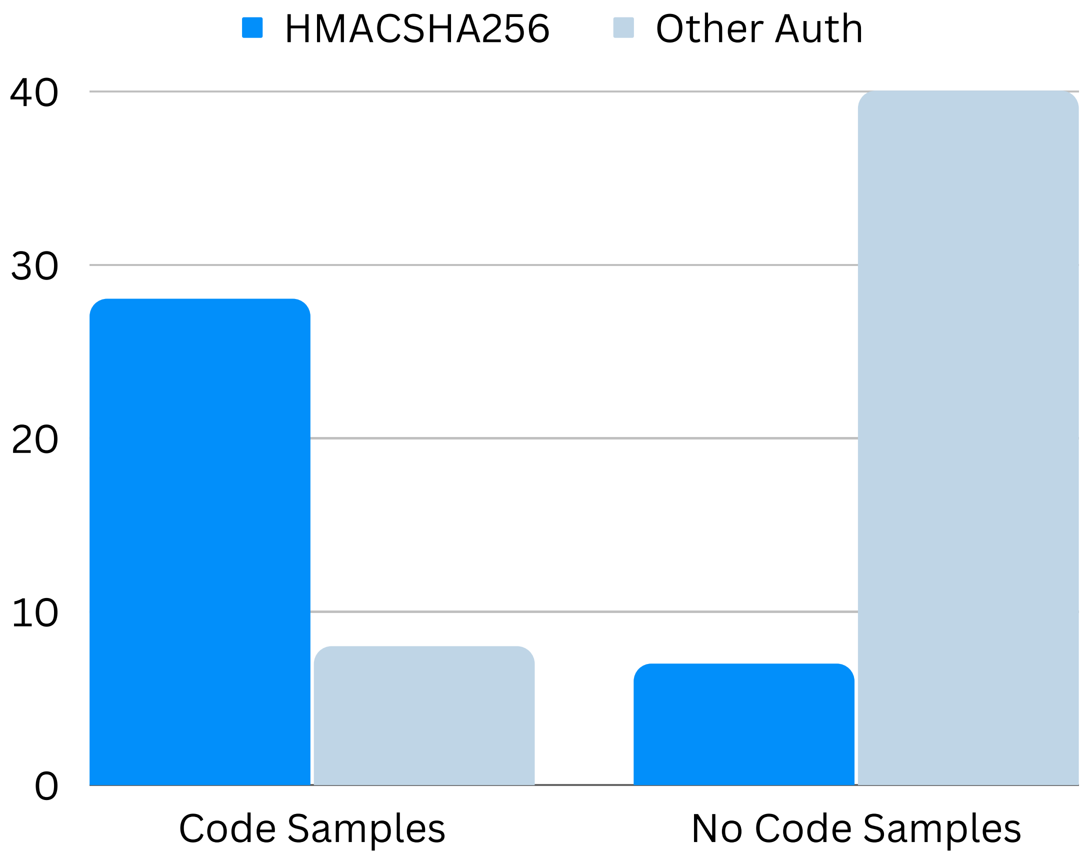
DOCUMENTATION

Documenting a webhook service well can save users time and spare them a headache.

CODE SAMPLES

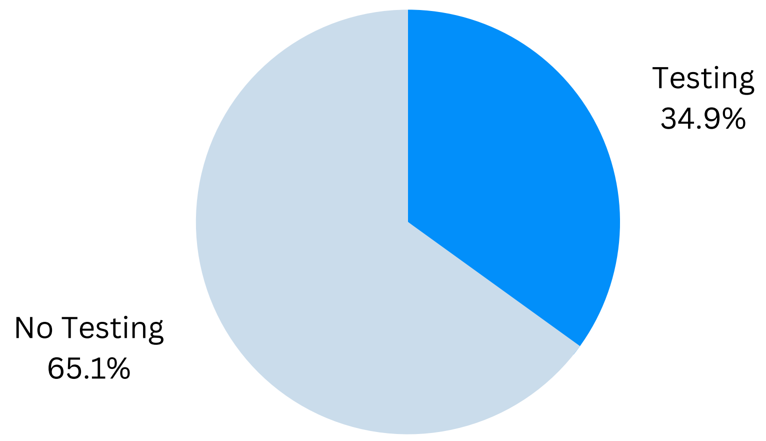
43

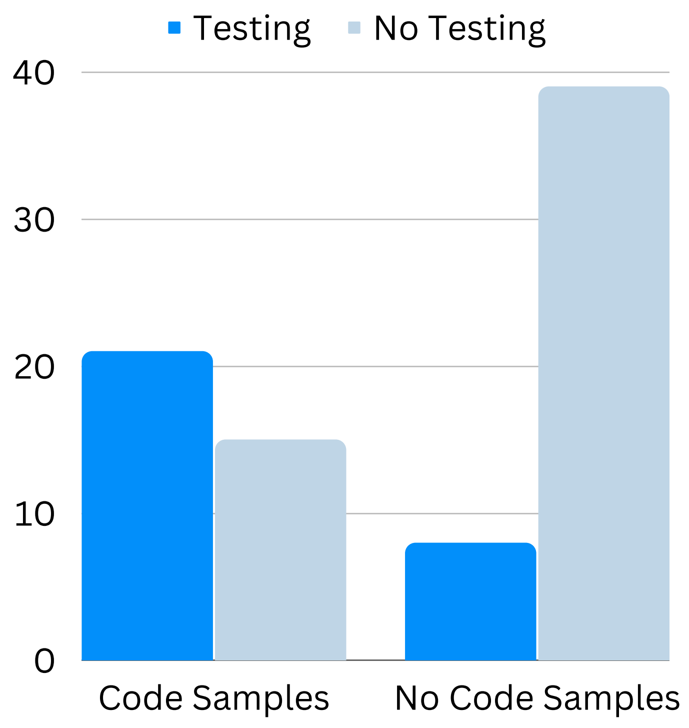
Having sample code makes developers lives easier. While only 43% offered code samples, the inclusion of code samples correlated heavily with using HMACSHA256 signatures. Github's docs[5] have a plethora of code samples.



TESTING

The best webhook documentation gives guidance on how to test their webhook implementations. Common guidance includes how to test webhooks locally (mostly using ngrok), listing various tools for spinning up endpoints to receive test messages, and providing the ability to send test events.





There is a high correlation between having code samples and providing guidance and tooling to test webhooks.

72% of those with code samples in their docs also provided testing guidance.

In contrast, those without code samples only provided guidance on testing 28% of the time.

DELIVERY

This section is based on internal delivery data gathered from Svix's Webhook as a Service product.

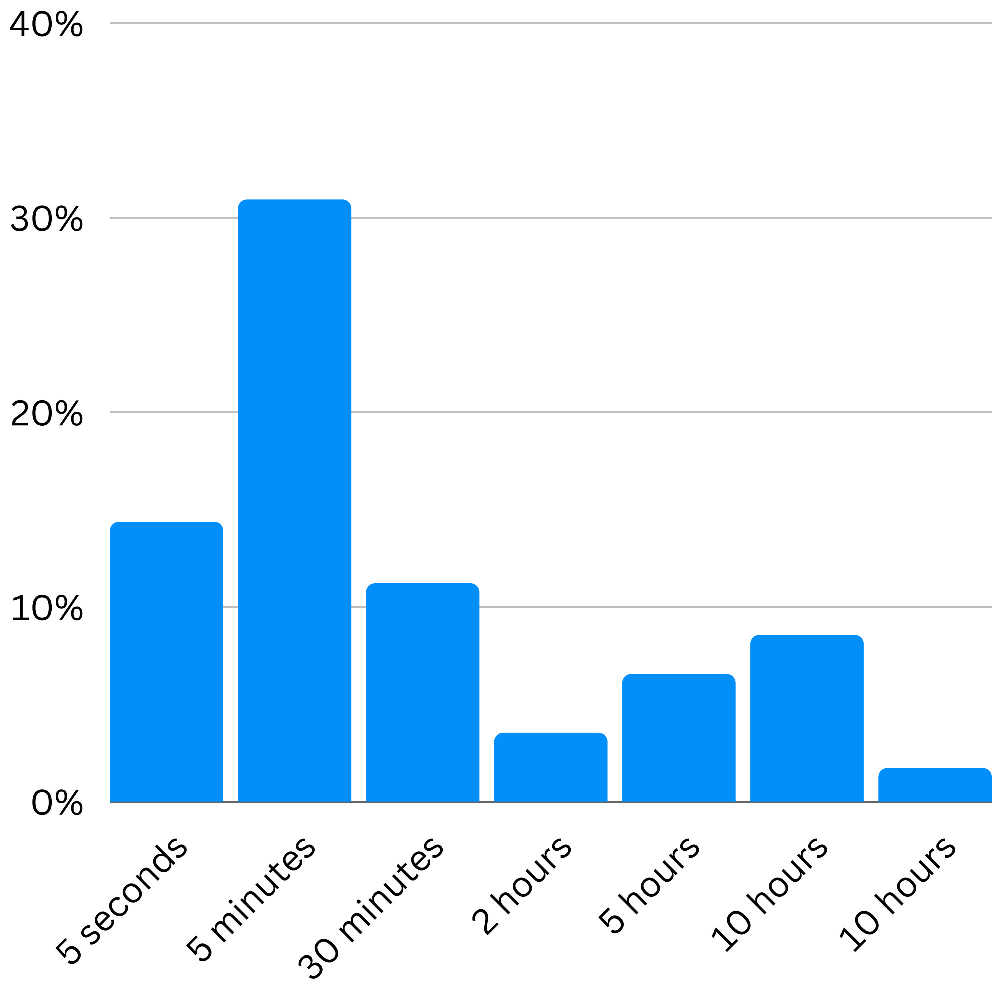
95.8%

95.8% of messages succeed on the first try. At first glance, this seems like a high success rate, but an endpoint failing to receive 4.2% of mission critical event notifications should be considered unacceptable. It's also probable this success rate is relatively high as Svix offers high deliverability functionality as part of our product offering (e.g. rate-limiting endpoints). Others' success rates will be lower based on the relatively low adoption rates of best practices.

This highlights the need to offer automatic retries. Not only does it increase the success rate of message delivery, but webhook consumers can be notified of endpoints that fail repeatedly and exhaust the retry schedule.

RETRIES

Automatic retries have been highlighted in this report but how useful are they in practice? Here are the success rates of retry attempts based on the Svix retry schedule:



Interestingly the 2nd retry succeeds at a higher rate than the first. This is probably due to the timing of the retries. On average, it take 2.5 retries to successfully deliver an initially failed message.

2.5

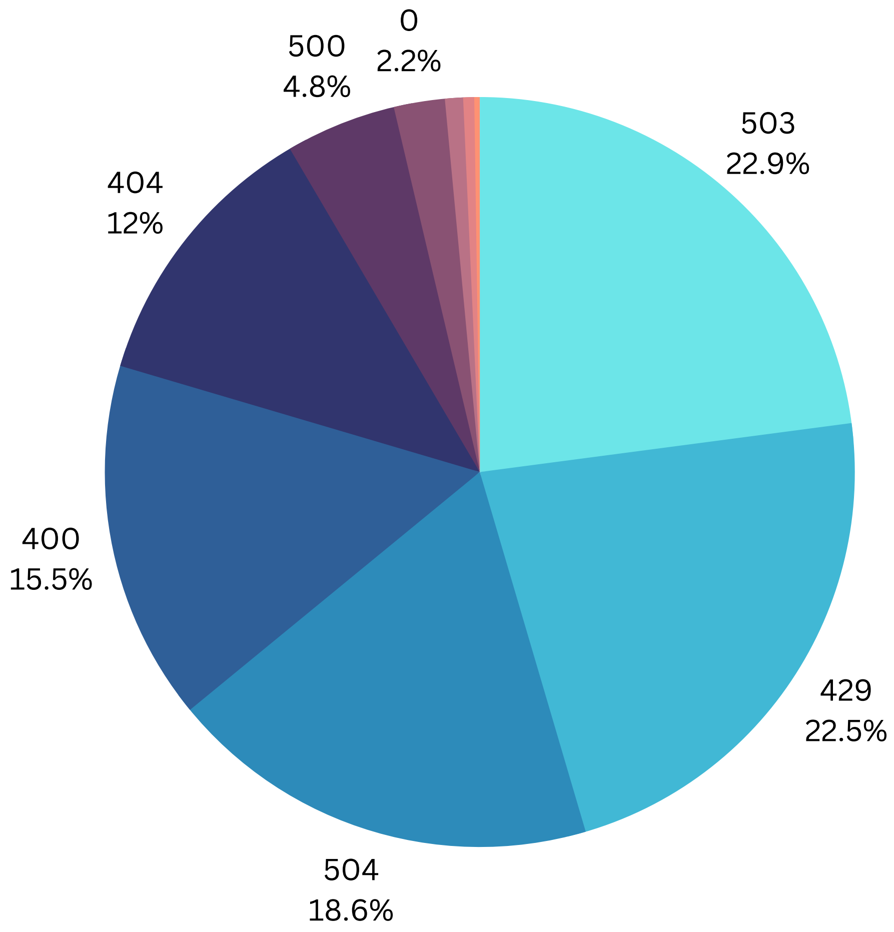
MESSAGE FAILURES

Unfortunately, sometimes endpoints are misconfigured and fail persistently. Retries are still helpful in this case as users can be sure there is an issue with their endpoint if they've exhausted the entire retry schedule.

1.4

At Svix, 1.4% of all messages end up exhausting their retry schedule without a successful delivery. This amounts to approximately 1/3 of all failures.

Here is a breakdown of the error codes in responses of failed webhooks. There is an even breakdown between the top 4 status codes. Status code 0 represents a timeout or DNS lookup failure which are rare.



RESPONSE TIMES

If a webhook attempt takes too long to finish, it may fail due to a request timeout. Svix requests will timeout after 15 seconds, with most other webhook providers timing out after 3-5 seconds. Below is the distribution of response times for successful deliveries:

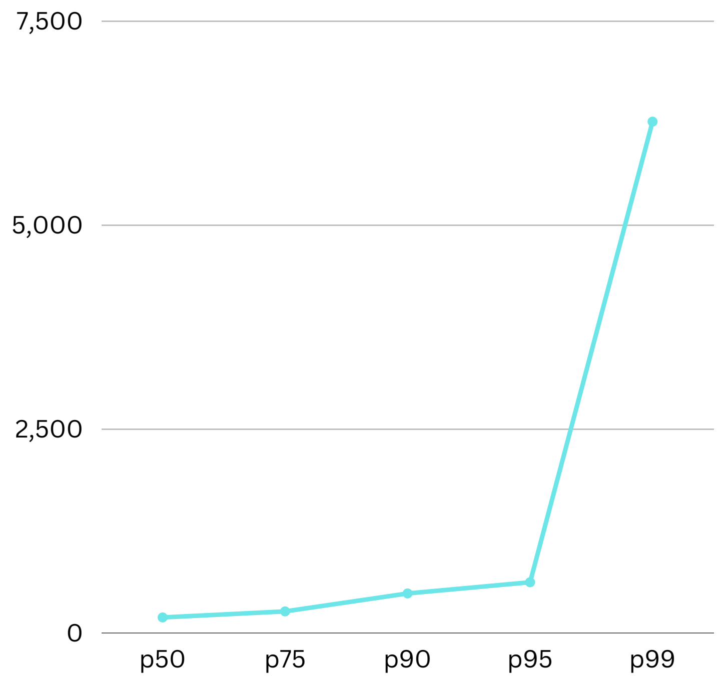
p50: 191 milliseconds

p75: 265 milliseconds

p90: 485 milliseconds

p95: 622 milliseconds

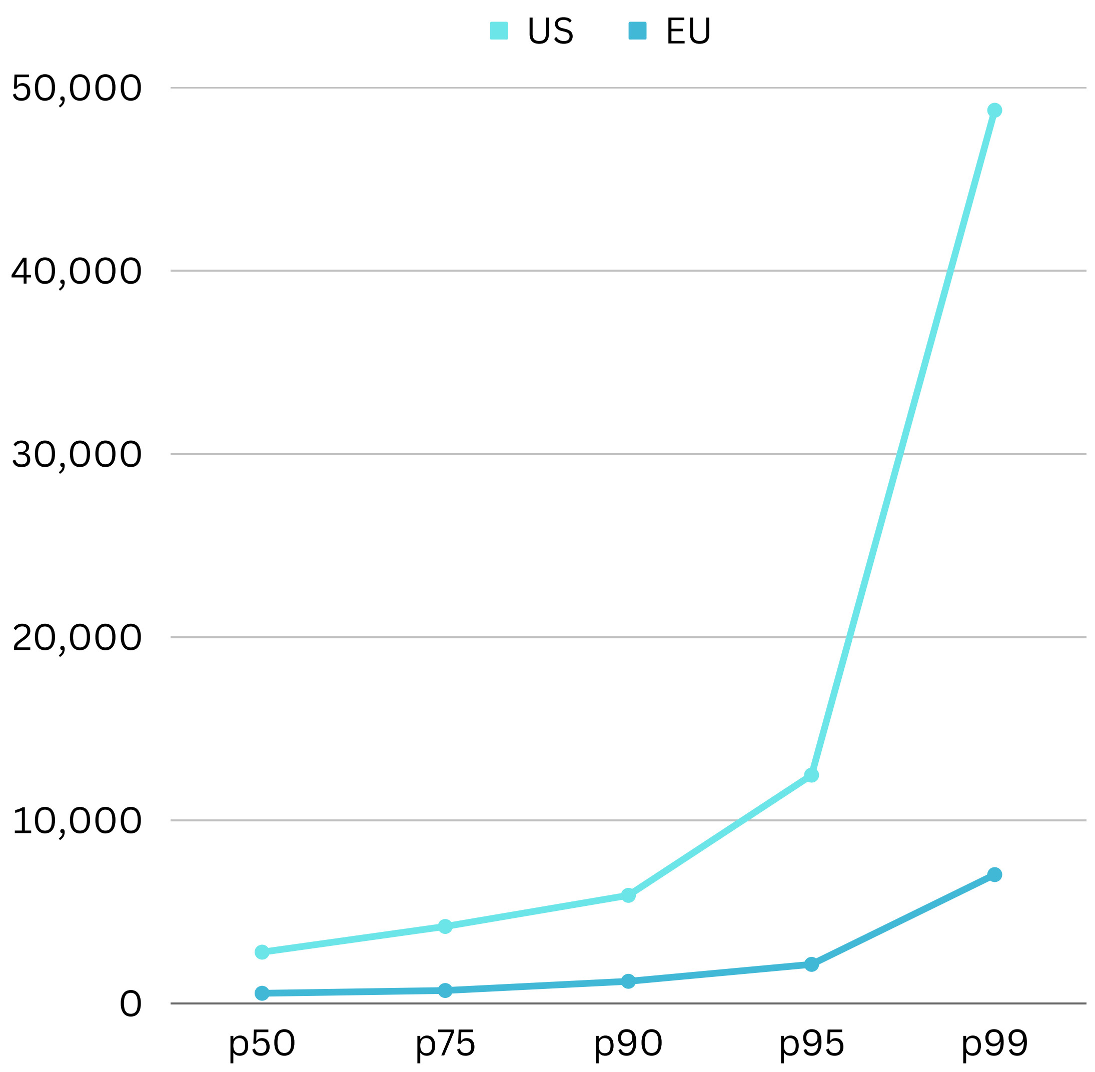
p99: 6,267 milliseconds



From the data, a 3s timeout policy would result in ~2-3% more failures, increasing failure rates by 50-60% and making timeouts the most common reason for failed deliveries.

PAYLOAD SIZES

Payload sizes depend heavily on the use case. Even when sending billions of webhooks for many different customers, the payload sizes varied greatly between the US and EU:



p99 payload size in the US was almost 50kb while in the EU it was only 7kb. For reference, GitHub's limit is 25MB while PagerDuty's is 55kb. There is a lot of variance here so think carefully about the use case before setting a payload limit.

METHODOLOGY

In order to generate the Best Practices and Documentation sections of this report, a list of 100 of the top API companies[6] was used as a starting point.

Several API providers on the list had no publicly available documentation of their API so they were replaced with well known companies that offer a public API.

The set of best practices to evaluate was based on the critical elements implemented by industry leaders who offer reliable webhook services.

The Deliveries section of the report is based on internal data from Svix.

SUMMARY AND KEY FINDINGS

Several key insights into the state of webhooks surfaced after evaluating 100 API providers and analyzing internal delivery data from Svix's Webhook as a Service product:

1. Webhook adoption is high at 83%.
2. Most best practices are not being implemented.
3. Automatic retries are necessary.
4. A majority of webhook documentation do not provide code samples or offer instructions on testing.
5. Automatic retries make webhook services more reliable.

In conclusion, webhooks are being adopted by most API providers, but they mostly fail to implement best practices.

Even the ones that do implement most best practices do so in different ways. The space is so fragmented the only providers with similar implementations were those that did not implement best practices at all.

Hopefully, this report will spark an increase in adoption of webhook best practices to help improve the developer experience around webhooks.

APPENDIX

Links to various resources:

1. List of 100 APIs: https://svix.com/blog/state-of-webhooks-2023
2. Dwolla’s retry schedule: https://developers.dwolla.com/api-reference/webhook-subscriptions#delivery-rate
3. Cloudinary’s event catalog: https://cloudinary.com/documentation/admin_api#event_types_table
4. Zoom’s signature scheme: https://stripe.com/docs/webhooks#verify-manually
5. Github’s webhook testing: https://docs.github.com/en/webhooks/using-webhooks/validating-webhook-deliveries
6. Original 100 APIs: https://www.blobr.io/report-state-of-api-developer-portals/lessons-from-top-100-api-companies