WHITE PAPER

API Monitoring: The Basics

Learn more about the engines that businesses rely on to power their applications to outpace the competition.

Introduction

Digital businesses are making a radical change in the way they build and deliver software. Gone are the days of apps that rely solely on in-house tools. Rather, today's apps are increasingly dependent on external APIs and third-party app providers (which, in turn, are reliant on other APIs and apps).

While this type of modularity allows for product flexibility and rapid development, it can be difficult to address any issues that arise. If even one component of this chain breaks, it can have a domino effect on its dependents, whereas a similar failure in the closed systems of yesterday would have just led to an isolated incident. As such, if your product relies on external APIs, it's important for you to monitor for more than just availability. You'll also need to keep tabs on performance, data validation and processes, feature changes and security.

This paper will:

- Describe and outline how businesses are using APIs.
- Explain the value of API monitoring.
- Help you understand best practices for API monitoring.
- Equip you to confidently craft an API monitoring strategy.

Background

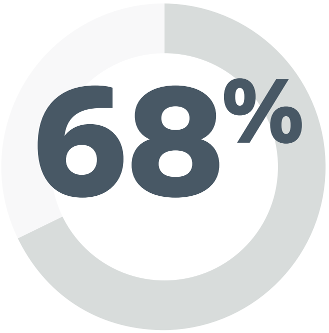
With the popularity of Agile methods, DevOps as its own (prominent) entity, and continuous integration/deployment in today's software engineering world, it's now possible and increasingly common to build large-scale systems that are truly modular. This approach is based on microservices — small, independent processes that can interact with other small processes and form complete applications when implemented and utilized together.

Complex software that was once shipped as a single, large entity is now being broken down into its constituent parts for design, implementation and maintenance.

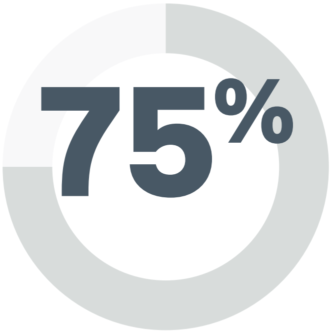
There are numerous benefits associated with this approach, including:

- Agility
- Efficiency
- Resiliency
- Revenue

According to **Forbes**, it's clear that microservice and API dependence is only increasing. Yet, for such a critical component of companies' digital user experience, far too often they're neglected when it comes to performance monitoring.



68% of companies are currently using, building or investigating microservices



75% of companies recognize a gap in microservice performance

[source] **NGINX: The Future of Application Development and Delivery is Now**

What are APIs?

An API is a set of programming instructions and standards for accessing a web-based software application or service. APIs give developers instructions about how to interact with services and can be used to connect data between different systems.

Today, entire businesses and applications are built on open APIs, relying on the ability to pass data back and forth between systems.

When it comes to sending and receiving API messages, it's important to know that APIs may be private, programmer APIs used within an internal organization or they may be public, consumer-facing APIs.

Private APIs make businesses more agile, flexible and powerful. Public APIs help businesses connect to offer new integrations and build new partnerships.

APIs can:

1. Provide documentation on how to programmatically interact with a web application.
2. Act as intermediaries between two web applications or an application and a developer.
3. Execute a series of predefined functions for interacting with an application.

Common functions include:

- Fetching data
- Posting data
- Executing tasks

SOAP and REST

At their core, APIs are about requesting and receiving data from a remote service. But different API formats may use different ways of structuring request and response bodies.

SOAP and REST have both been around for over a decade, yet there are still many misconceptions surrounding the debate between them. Some experts argue that comparing these two types of APIs is like comparing apples to oranges. They emphasize that SOAP is a protocol and REST is an architectural style.

For our purposes we treat them as types of APIs, with different benefits.

Why use SOAP?	Why use REST?
Because of the popularity of RESTful APIs, SOAP is sometimes viewed as a legacy technology. There are still cases where it makes sense to implement SOAP because it offers:	**REST APIs are becoming increasingly popular on the web. Right now over 70% of public APIs rely on REST. That's because it's:**
Standardization: With defined rules and specs, developers are able to reference best practices for their APIs and be sure about what their code does.	**Easy to use:** Quick setup and documentation is easy to create and maintain.
Enterprise-forward design: SOAP is designed for the enterprise, supporting ACID-compliant transactions and two-phase commits across distributed transactional resources.	**Efficient:** Unlike with SOAP, REST data can be easily cached. Combined with lower message overhead (lightweight JSON vs. hefty XML), you get a faster, lower bandwidth API, especially for mobile applications.
Built-in extensibility: Extensions like WS-Reliable Messaging and WS-Security help protect the consistency of data and ensure end-to-end security. REST has no such built-in functionality.	**Scalable:** REST can be swiftly adapted to suit the needs of expanding operations.
Transport layer independence: Unlike REST, which is HTTP/HTTPS-only, SOAP can be sent over HTTP/HTTPS, SNMP, TCP, UDP, JMS or XMPP.	**Flexible:** Output data is not restricted to XML. Can be JSON, CSV, XML, etc.
Error handling: SOAP's XML error metadata allows fault checking and automation of error handling.	**JavaScript-friendly:** REST integrates well with JavaScript, which allows easy building of dynamic APIs using tools such as Node.js.

API Performance

Why performance matters

Third-party APIs that you rely on

When an API fails and disrupts the performance or user experience on your site, this failure reflects on your brand and reputation. End users and customers likely won't recognize that a third party could be at fault. And depending on how critical that API is to a transaction process, this failure could impact your bottom line right away.

For example, if a key component of the checkout process on your website is a location-based search and you rely on a third-party API to provide the search by location, when that API doesn't work correctly then your potential customers can't check out successfully.

As a developer or a site owner you may decide that the benefit of relying on the third-party service outweighs the risk of these types of failures. In order to accurately assess the risk and have visibility into the impact of these services over time, it's crucial that you monitor the part of your site's user flow that relies on an API. Similarly, if you have an open API that you've made available to partners or developers, then you have a responsibility to ensure that the API is available and working as expected.

APIs that you build or support

Let's say that you've developed a new internal API that passes order data from a mobile device to a system in your product warehouse. Maybe it's critical that the data passes to the warehouse within two minutes, or the entire production schedule will be off. When you developed the API and tested it in staging it always passed data successfully within one minute, but when you launch the API and start processing real requests, you notice that the real response time is creeping up closer and closer to that two-minute threshold.

Without active monitoring on the API in production your team might assume that the developed API is fast enough based on pre-production tests.

APIs that you host should be actively monitored in both pre-production and production environments.

Whether it's to keep tabs on your own APIs or see the impact of external services that you rely on, it's important to monitor APIs for availability, functionality, speed and performance.

If you know you have an API that's been unreliable in the past and you're not actively monitoring that API, start the conversation with your team and develop a plan to begin monitoring that API today.

Monitor for availability

At the most basic level, API monitoring checks to see if the resource is available (and therefore responding to calls) or not. However, with the increasing levels of interdependence of apps on other apps and services, you should strongly consider monitoring the availability of the resources your APIs rely on as well. Receiving notifications about any possible breaks in the chain of dependence allows you to act appropriately to ensure that your site or app stays online, even if others don't.

Monitor for performance

So, an API returns calls correctly. That's all there is to it, right? Wrong. Even if an API returns calls correctly, you'll want to ensure that it's performant. How quick are the responses? Are the response times degrading (even if they're still pretty quick)? Does the API's performance vary in different environments (such as in development versus production, rural versus urban and so on)?

Any of these issues, whether it's caused by the APIs you call or ones down the line, means that your product is slow as well. You may not be able to control the API's performance, but there are things you can control, such as whether or not your app relies on this API or if you need to implement temporary changes that address the way the data is coming back to you.

Monitor for data validation and multi-step processes

Your API may be available and responding to requests, but what if it sends back incorrect data or is not formatted in a way you anticipate? This is why you should test regularly to ensure that your systems are getting what they need to carry out tasks.

Additionally, you should check to see if any multi-step processes you carry out once you've received the response also work as expected. Can you cache data from calls to save on repeat calls to the API? Does your authentication work as expected?

Certainly, testing in production is important, but it's better to get ahead of any bugs before they're deployed. This, combined with greater adoption of continuous integration/deployment methodologies, amplifies the need for robust API testing throughout the development life cycle.

Monitor for integrations with third party and partner APIs

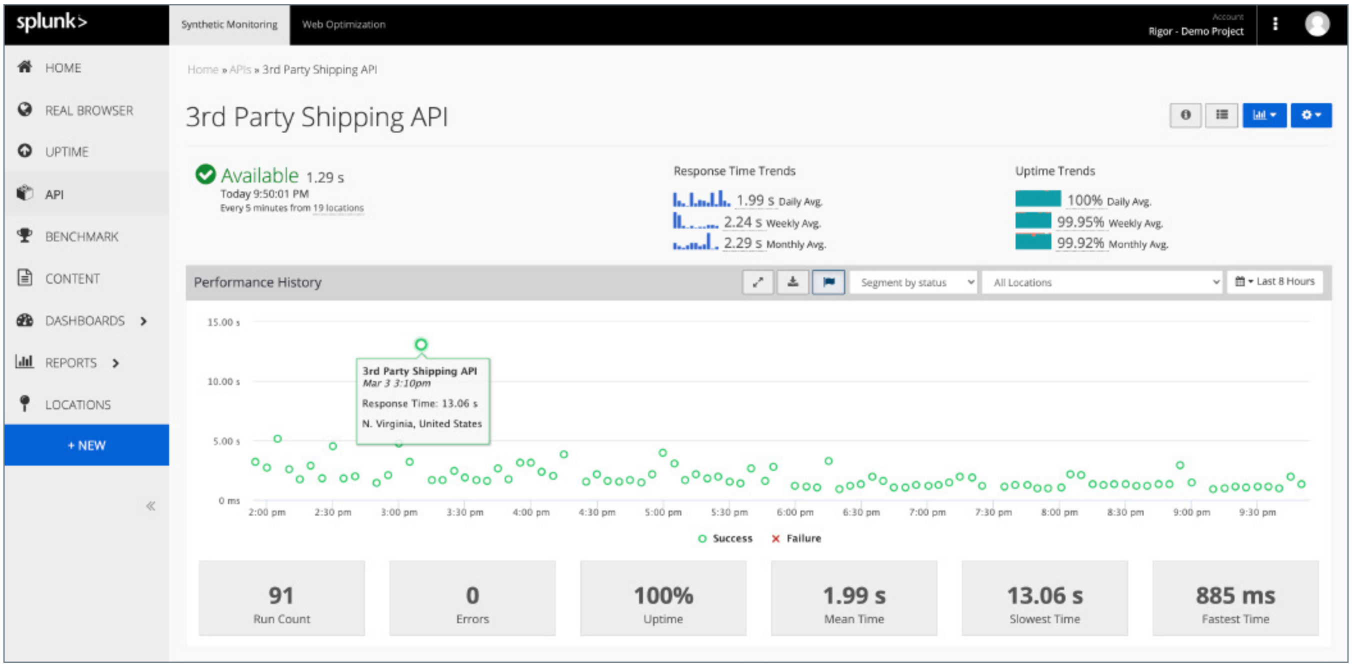
As we mentioned before, apps and services oftentimes rely on both managed and third-party APIs, and frequently users can't tell the difference between the two. Ensure that your monitoring solution gives you visibility into the performance of third party and partner APIs in addition to those you manage. By doing so, you'll not only be able to hold partners accountable, but also know who to contact should any issues arise.

Monitor for feature changes

When you have functionality that depends on the performance of an external service, you'll want to ensure that your app remains compatible with the service. Regardless of whether the changes are a result of new releases or bug fixes, your base code may not work with the service from generation to generation.

Monitor for security

Be wary of any packages that you've used or included in your product, as well as any integrations you're using. Not only can this be a way to introduce bugs into your product, you can open yourself up to security vulnerabilities if the external source becomes compromised or doesn't include the appropriate precautions. For example, if you've included a third-party dependency to implement photo uploading functionality and the package doesn't include type checking to ensure that users can only upload photos, someone might attempt to upload an executable that's then run natively on your system.



3rd Party Shipping API

Available 1.29 s
Today 9:50:01 PM
Every 5 minutes from 19 locations

Response Time Trends
1.99 s Daily Avg.
2.24 s Weekly Avg.
2.29 s Monthly Avg.

Uptime Trends
100% Daily Avg.
99.95% Weekly Avg.
99.92% Monthly Avg.

Performance History

3rd Party Shipping API
Mar 3 3:10pm
Response Time: 13.06 s
N. Virginia, United States

91	0	100%	1.99 s	13.06 s	885 ms
Run Count	Errors	Uptime	Mean Time	Slowest Time	Fastest Time

How Splunk Synthetic Monitoring can help monitor your APIs

We've developed a framework for performance tests that can help businesses understand whether APIs are performing as expected.

With API checks we can:

- **Track** the availability of critical APIs.
- **Capture and trend** an API's response time.
- **Ensure API functionality** by validating the API's response values and structure.
- **Alert on any conditions** indicating a broken or poorly performing API, allowing teams to get ahead of issues before they affect users or breach SLAs.

Splunk's API checks are built upon four simple concepts:

1. Making HTTP requests
2. Extracting data from responses
3. Saving data for use in additional requests or analysis
4. Making assertions about data and its format

The components are simple but powerful because these pieces can be assembled for test cases capable of monitoring and verifying some of the most complicated API flows.

These four basic components can be configured in a robust way, allowing businesses to actively monitor almost any user flow; but even so, some API transactions may provide limited feedback. For this reason we added the ability to insert and execute custom JavaScript as part of a monitoring flow.

This ability to augment check steps with JavaScript means that you can adapt your API checks for any type of flow or transaction. This step type allows you to access variables, run computations and return data to be saved as new variables. This powerful feature makes API checks more flexible and robust than ever.

Built to scale with your business

Whether an API is powering a web property, a mobile app or even the core infrastructure of your business, an API check with Splunk Synthetic Monitoring ensures that API is available, performing and functional in ways that users of any technical level can understand. Additionally, the easy-to-use drag-and-drop interface makes it simple to create, test and manage APIs.

Here are just a few of the business needs that the API check with Splunk Synthetic Monitoring can solve:

- Testing complex, multi-step API flows
- Monitoring availability and response time from geographies around the world
- Tracking and enforcing performance SLAs of third-party APIs
- Verifying correctness of API responses
- Testing the entire CRUD life cycle of a data object via an API
- Handling complex token-based API authentication systems
- Monitoring application status pages

Takeaways

Companies increasingly rely on APIs to perform business-critical functions. And yet, many either aren't monitoring their APIs, or they're failing to implement best practices.

Because of the **high stakes associated with any type of downtime**, it's essential for companies to monitor dependencies to ensure that everything is working as expected. Businesses might not have any control over the third-party tools they rely on, but with comprehensive monitoring, they can know right away if they need to take action to keep their product up and running.

From basic to enterprise grade, Splunk has your back

Our customers use API checks to test everything from simple web services to complex multi-step streaming video services. No matter your API or your industry, Splunk has you covered.

To learn more about Splunk Synthetic Monitoring and to try a free demo, **contact us** today.

splunk>

Learn more: www.splunk.com/asksales

www.splunk.com