A guide to the different types of APIs

The Postman Team

July 6, 2022

APIs are one of the most important components of modern-day development, forming the bridges between applications to transfer information and deliver services. However, even as developers and C-suite executives increasingly consider APIs early in the development lifecycle, there remains some confusion about which types of APIs are best suited to particular applications.

Each API type has its own architecture, which evolved to meet the demands of that time. And as technology has developed over the years, and our relationship to technology has shifted, we've seen APIs emerge to fill new gaps and serve different uses.

In this blog post, we'll run through the different types of API protocols to give you a sense of their pros and cons, common use cases, and other considerations when you're choosing the best one for your needs.

Related: **What are the components of an API?**

REST

Related: **REST API client in Postman**

REST APIs are designed to make server-side data readily available by representing it in simple formats such as JSON and XML. The acronym stands for REpresentational State Transfer, and it was released in 2000 after being introduced in an academic thesis by Roy Fielding. This particular type of API adheres to six specific architectural constraints:

A uniform interface

Completely stateless

Native caching

Client-server architecture

A layered system

The ability to provide executable code to the client

Pros

Operations are executed with different HTTP methods including GET, POST, PUT, DELETE, OPTIONS, and PATCH. By leveraging these functions, REST APIs become extremely capable across the internet.

The key benefits of this API type are that the client and the server are completely decoupled from one another. This allows for abstraction layers that help to

maintain flexibility even as a system grows and evolves. In addition, REST is cache friendly and supports multiple formats, which is important when you're building public-facing APIs. The flexible data formatting that you get as a result makes them extremely useful for varied applications.

REST APIs are most commonly used as management APIs to interact with objects in a system. They also are useful when you're building simple resource-driven apps that don't need much in terms of query flexibility.

Cons

Where REST APIs fall short is that their rich metadata creates big payloads that can sometimes cause more trouble than they're worth. You can get over- and under-fetching problems that require further API requests, bogging down the process.

From an industry perspective, another key externality resulting from the flexibility is that there is no binding contract on what structure is used for messages. As a result, there is a lot of back and forth when it comes to implementation – which can cause unnecessary frustration and bottlenecks.

Related: REST vs SOAP

SOAP

Related: SOAP API client in Postman

SOAP APIs are formatted as XML files and they are extremely common web communication protocols. The acronym stands for **Simple Object Access Protocol**, and it was developed in the late 1990s. Despite its age, SOAP still remains one of the more popular API types used by developers.

Pros

The major advantage of SOAP is the fact that it is completely agnostic when it comes to the programming language and processing platform. The standardized format ensures that no matter what is receiving the message on the other end, the request can be executed.

In addition, this type of API comes with native error handling already built-in, helping developers to be proactive and solve issues before they snowball.

Perhaps the strongest SOAP use case is with high-security data transmissions in situations where two parties have agreed to a specific legal contract. Here SOAP's standardization works wonders because it allows for the contract's terms to be formally codified and enforced throughout all of the API's processing. You'll see plenty of this in high-stakes industries like financial services, where the data being passed back and forth is highly sensitive.

Cons

SOAP's standardization makes requests incredibly accessible to applications, but as a side effect, the format can become very formal and verbose. Every message must include an envelope tag at the start and the end, a body that includes the actual request, a header for specific information and additional requirements, as well as any faults that occur throughout processing.

SOAP has become less popular in recent years because of the sheer volume of information it requires. The XML files are large and are often unnecessarily clunky – especially for simple systems. The number of people who specialize in SOAP servers is rapidly declining, and that makes it a difficult thing to maintain if you

don’t already have the right talent on board. It feels like the format is lagging behind newer, more flexible communication methods that offer more robust, sustainable results.

GraphQL

Related: **GraphQL client in Postman**

GraphQL was developed in response to REST APIs, with the idea that you could execute precise syntax that retrieves only what is needed, lightening the payload and simplifying the process significantly.

GraphQL is an internal Facebook protocol that was first released in 2015, quickly becoming the go-to API type for JSON. You start by creating a schema that describes all the possible queries and the specific types that they return. This can be challenging, but once completed, the API can accept specific requests and return a result that matches exactly what the user is looking for.

There remains a heated debate in the technology industry as to whether companies should build a GraphQL API alongside an existing REST API or whether it should be a complete replacement. If you’re ever looking to stir up some controversy among developers, this is a good way to do it.

Pros

The advantages of immersing yourself in GraphQL APIs are that the API queries are transparent and well documented, giving users all the information they need to use it effectively. In addition, the precise results, detailed error messages, and flexible permissions help round out a well-balanced but high-functioning API. This is especially true when it comes to data structuring, where GraphQL gives users significant flexibility.

Cons

Performance can suffer with GraphQL if you have too many nested fields in any one request. It also doesn’t reuse standard HTTP caching semantics, so it actually requires custom efforts to achieve proper caching. All of this makes GraphQL difficult to pick up without lots of training and experience. It certainly is an acquired skill, breaking away from the long track record and industry knowledge around REST APIs.

gRPC

Related: **gRPC APIs in Postman**

In gRPC, RPC stands for Remote Procedure Call and refers to something that can execute a function housed elsewhere but in a different context. The ‘*g*’ appended at the beginning narrows things down to the most advanced version developed by Google back in 2015. A user on one side will select a remote procedure to execute, serialize the necessary parameters and then append any additional information into the message. This will then be sent to the server, which interacts with the other application, decoding the message and executing the operation. A result then comes back to the initial user.

Pros

This is a powerful form of API because of its simplicity. It’s straightforward, using GET to fetch information and POST for everything else. This means that functions are easy to add, and for lightweight payloads, you get great performance overall. The ability to define any type of function makes it infinitely configurable. Essentially you’re limited only by your imagination.

Typical use cases include command APIs that send simple requests to remote systems and customer-specific APIs, which help manage internal microservices at scale with great speed. By simplifying otherwise complex remote calls, gRPC has also become a staple of the Docker-based application world, proving its value when you have massive numbers of remote calls to execute.

Cons

Where gRPC falls short is in the fact that it's tightly coupled to the underlying system, which restricts its reusability in many cases. In addition, there is no abstraction layer sitting between the API and the actual system functions, which can raise security concerns.

WebSockets

Related: **Using WebSocket requests in Postman**

Finally, we must briefly touch on WebSockets in relation to gRPC as discussed above. WebSockets is a communications protocol that provides dynamic communication over a single TCP connection.

Pros

WebSockets acts as a thin transport layer that sits on top of your TCP/IP technology stack. With the API, you can send messages to a server and receive event-driven responses without needing to poll the server for a reply.

Cons

The inherent difference between a WebSocket API and a gRPC API is that WebSockets is based on HTTP/1.1 whereas gRPC was built using HTTP/2. It was a natural step to take the technology forward, though that's not to say that either one is better. There are pros and cons to both options, and your specific choice should be in line with your needs.

Finding the right fit

As you can see, there is no "best" API type. Each has its own quirks and will be suited to particular applications. We're spoiled to have so many great choices at our disposal—and it's our job to find the one that makes sense for the objectives we're trying to accomplish.

When making a decision, you need to consider a few things:

What programming language do you want to use?

Which environment are you going to be developing in?

What is the core functionality that matters for this API?

What skills do you have in-house? Or what skills can you procure?

What resources have you budgeted for this work?

If you can answer those in advance, you can select the API type best suited for your needs. Keep in mind that you can always try a small proof-of-concept with one or two types to see how they will function.

We hope this blog post has been useful in differentiating between the most

popular types of APIs. Each has its place, and when you make a thoughtful and intentional decision up front, you're well on your way to creating an API that will not only be effective but resource-conscious.

Technical review by Arlemi Turpault.

Tags: API 101 Tutorials

The Postman Team

Today, more than 30 million developers use the Postman API Platform. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster.

View all posts by The Postman Team →

What do you think about this topic? Tell us in a comment below.

Comment

Your name

Your email

Write a public comment

Post Comment

9 thoughts on "A guide to the different types of APIs"

Alfredo Colombano
July 11, 2022

Great to have these short knowledge pills for those who are beginning in this area.

Kaushalya
October 4, 2023

I learnt many things about API. It’s very important.

Linal Punchihewa
December 29, 2023

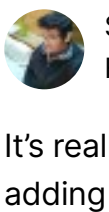
I learnt many things about API. It’s very important and i like to thank you

Rahul Varma
January 22, 2024

A great introduction to API types.

Prathmesh Jadhav
February 12, 2024

Very Informative for begineers who want to know more from less content.



Satwik Jain
February 22, 2024

It’s really a good article, still it can be more interesting by adding examples and use cases for all API’s

Mugisha
March 6, 2024

What a brief yet exclusive summary of the API technology , Grateful for the lesson

Goitseone Themba
May 13, 2024

Great read, I now have a clearer picture on different types of APIs. Can’t wait to learn more...

Adyakanta Behuria
July 10, 2024

Good

You might also like

How to improve API

Announcing Postman has

How Postman uses

discoverability with Postman workspaces

Gbadebo Bello

One of the best ways for API producers to improve their API distribution strategy is to invest heavily in the discoverability of...

Read more →

acquired Orbit

Abhinav Asthana

Since day one, Postman has been about developer productivity. It started by saving developers from repeatedly typing the same commands and grew...

Read more →

Postman: bridging the gap between technology and data science

Melinda Gutermuth

We love hearing from our community about how you're using Postman to improve your API development experience—it truly continues to inspire us....

Read more →

Postman v11 is here!

It's jam-packed with updates to help you collaborate on your APIs, augment yourself with AI, and more.

See what's inside v11 →

Product

What is Postman?

API Repository

Tools

Governance

Workspaces

Integrations

Enterprise

Plans and pricing

Download the app

Support Center

Company

About

Careers and culture

Press and media

Contact us

Partner program

Legal and Security

Terms of Service

Trust and Safety

Privacy policy

Cookie notice

Privacy choices

API Categories

App Security

Payments

Financial Services

DevOps

Developer Productivity

Data Analytics

Communication

Artifical Intelligence

Social

Twitter

LinkedIn

GitHub

YouTube

Twitch