

[IBM Live Webinar — Register now](#)

What is the Future of APM and Observability?



What is an API (application programming interface)?



09 April 2024

Authors



Michael Goodwin

Editorial lead, Automation & ITOps

What is an API?

Sandpiper EX2012
Google v. Sandpiper
IPR2025-01010

An API, or application programming interface, is a set of rules or protocols that enables software applications to communicate with each other to exchange data, features and functionality.

APIs simplify and accelerate application and [software development](#) by enabling developers to integrate data, services and capabilities from other applications, instead of developing them from scratch. APIs give application owners a simple, secure way to make application data and functions available to both internal and external API consumers.

APIs are predefined interfaces that share only the necessary data and functions for specific queries. Servers do not have to fully expose data—APIs enable the sharing of small packets of data, relevant to the specific request. They keep other internal system details hidden, which helps preserve the integrity of system security.

Successful API adoption, integration and [management](#) is driven by comprehensive, well-maintained API documentation. API documentation is like a technical instruction manual that provides details about an API (such as supported protocols, languages and authentication methods) and information for developers on how to work with an API and its services. Well-designed documentation promotes a better API experience for users and generally makes for more successful APIs.

Think Newsletter

The latest tech news, backed by expert insights

Join over **100,000 subscribers** who receive access to learning hubs, expert insights and industry news on AI, security, automation, data and infrastructure—all curated in the Think Newsletter. See the [IBM Privacy Statement](#).

Subscribe

How do APIs work?

It's useful to think about API communication in terms of a request and response between a client and server. The application submitting the request is the client, and the server provides the response. The API is the bridge establishing the connection between them.

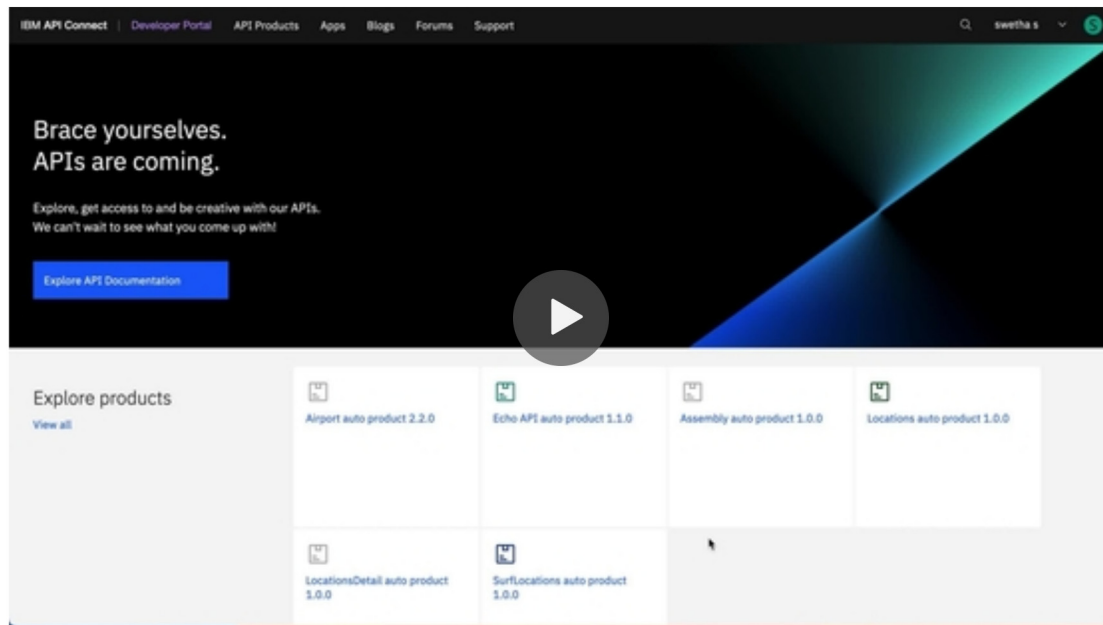
A simple way to understand how APIs work is to look at a common example, third-party payment processing. When a user purchases a product on an [e-commerce](#) site, the site might prompt the user to “Pay with PayPal” or another type of third-party system. This function relies on APIs to make the connection.

- When the buyer clicks the payment button, an API call is sent to retrieve information. This is the request. This request is processed from an application to the web server through the API's Uniform Resource Identifier (URI) and includes a request verb, headers, and sometimes, a request body.
- After receiving a valid request from the product webpage, the API calls to the external program or web server, in the case, the third-party payment system.
- The server sends a response to the API with the requested information.

- The API transfers the data to the initial requesting application, in this case, the product website.

While the data transfer differs depending on the web service used, the requests and responses all happen through an API. There is no visibility on the user interface, meaning APIs exchange data within the computer or application, and appear to the user as a seamless connection.

API Connect



IBM API Connect Developer Portal

See how the API Connect Developer Portal empowers you to share, manage, and customize APIs for your developers—accelerating integration and collaboration with a secure, scalable platform.

[Explore the developer portal](#) →

Types of APIs

APIs can be categorized by use case, including data APIs, operating system APIs, remote APIs and web APIs.

Web APIs

Web APIs enable the transfer of data and functionality over the internet using HTTP protocol.

Today, most APIs are web APIs. Web APIs are a type of remote API (meaning that the API uses protocols to manipulate external resources) that expose an application's data and functionality over the internet.

The four main types of web APIs are:

Open APIs



Open APIs are open-source application programming interfaces you can access with the HTTP protocol. Also known as public APIs, they have defined API endpoints and request and response formats.

Partner APIs

Partner APIs connect strategic business partners. Typically, developers access these APIs in self-service mode through a public [API developer portal](#). Still, they need to complete an onboarding process and get login credentials to access partner APIs.

Internal APIs

Internal, or private, APIs remain hidden from external users. These private APIs aren't available for users outside of the company. Instead, organizations use them to improve productivity and communication across different internal development teams.

Composite APIs

Composite APIs combine multiple data or service APIs. They allow programmers to access several endpoints in a single call. Composite APIs are useful in microservices architecture where running a single task might require information from several sources.

Other types of APIs

Less common types of APIs include:

- **Data (or database) APIs**, used to connect applications and database management systems
- **Operating system (or local) APIs**, used to define how apps use operating system services and resources
- **Remote APIs**, used to define how applications on different devices interact

API examples

Because APIs allow organizations to open access to their resources while maintaining security and control, they have become a valuable aspect of modern business and personal applications.

Here are some API examples that users often encounter:

Universal logins

A popular API example is the function that enables people to log in to websites by using their Facebook, X, or Google profile login details. This convenient feature allows any website to use an API from one of the more popular services for quick authentication. This capability helps save users the time and hassle of setting up a new profile for every web application or new membership.

Internet of Things (IoT)



These “[smart devices](#)” offer added functionality, such as internet-enabled touchscreens and data collection, through APIs. For example, a smart fridge can connect to recipe applications or take and send notes to mobile phones through text message. Internal cameras connect to various applications so that users can see the contents of the refrigerator from anywhere.

Travel booking comparisons

Travel booking sites aggregate thousands of flights, showcasing the cheapest options for every date and destination. APIs enable this service by providing application users access to the latest information about availability from hotels and airlines.

This access is available either through a web browser or the travel booking company’s own application. With an autonomous exchange of data and requests, APIs dramatically reduce the time and effort involved in checking for available flights or accommodation.

Navigation apps

Navigation apps use core APIs that display static or interactive maps. These apps also use other APIs and features to provide users with directions, speed limits, points of interest, traffic warnings and more. Users communicate with an API when plotting travel routes or tracking items on the move, such as a delivery vehicle.

Social media

Social media companies use APIs to allow other entities to share and embed content featured on social media apps to their own sites. For example, the Instagram API enables businesses to embed their Instagram grid on their website and for the grid to update automatically as users add new posts.

SaaS applications

APIs are an integral part of the growth in [software as a service \(SaaS\)](#) products. Platforms like CRMs (customer relationship management tools) often include several built-in APIs that let companies [integrate with applications](#) they already use, such as messaging, social media and email apps.

This integration drastically reduces time spent switching between applications for sales and marketing tasks. It also helps reduce or prevent data silos that might exist between departments that use different applications.

API protocols, architectural styles and languages

Traditionally, API referred to an interface connected to an application created with any of the low-level programming languages, such as JavaScript. However, modern APIs vary in their architectures and use of data formats. They are typically built for HTTP, resulting in developer-friendly interfaces that are easily accessible and widely understood by applications written in [Java](#), Ruby, Python and many other languages.

As the use of web APIs has increased, it has led to the development and use of certain protocols, styles, standards and languages. These structures provide users with a set of defined rules, or API specifications, that create accepted data types, commands and syntax. In effect, these API protocols facilitate standardized information exchange.



SOAP (simple object access protocol)

SOAP is a lightweight XML-based messaging protocol specification that enables endpoints to send and receive data through a range of communication protocols including SMTP (simple mail transfer protocol) and HTTP (hypertext transfer protocol.) SOAP is independent, which allows SOAP APIs to share information between apps or software components running in different environments or written in different languages.

Remote procedure call (RPC)

Remote procedure call (RPC) is a protocol that provides the high-level communications paradigm used in the operating system. RPC presumes the existence of a low-level transport protocol, such as transmission control protocol/internet protocol (TCP/IP) or user datagram protocol (UDP), for carrying the message data between communicating programs.

RPC implements a logical client-to-server communications system designed specifically for the support of network applications. The RPC protocol enables users to work with remote procedures as if the procedures were local.¹

XML-RPC (XML- remote procedure call)

The XML-RPC protocol relies on a specific XML format to transfer data. XML-RPC is older than SOAP, but simpler, and relatively lightweight in that it uses minimum bandwidth.

JSON-RPC

Like XML-RPC, JSON-RPC is a remote procedure call, that uses JSON (JavaScript Object Notation) instead of XML. JSON is a lightweight format for data exchange that is simple to parse and uses name/value pairs and ordered lists of values. Because JSON uses universal data structures, it can be used with any programming language.

gRPC

gRPC is a high-performance, open-source RPC framework initially developed by Google. gRPC uses the network protocol HTTP/2 and Protocol Buffers data format and is commonly used to connect services in a microservices architecture.

WebSocket

WebSocket APIs enable bidirectional communication between client and server. This type of API does not require a new connection to be established for each communication, once the connection is established it allows for continuous exchange. This makes Web Socket APIs ideal for real-time communication.

REST (representational state transfer)

REST is a set of web API architecture principles. [REST APIs](#), also known as RESTful APIs, are APIs that adhere to certain REST architectural constraints. REST APIs use HTTP requests such as GET, PUT, HEAD and DELETE to interact with resources. REST makes data available as resources, with each resource represented by a unique URI. Clients request a resource by providing its URI.

REST APIs are stateless, they do not save client data between requests. It's possible to build RESTful APIs with SOAP protocols, but practitioners usually view the two standards as competing specifications.



GraphQL

[GraphQL](#) is an open-source query language and server-side runtime that specifies how clients should interact with APIs.² GraphQL allows users to make API requests with just a few lines, rather than having to access complex endpoints with many parameters. This capability can make it easier to generate and respond to API queries, particularly more complex or specific requests that target multiple resources.

REST vs. SOAP

SOAP and REST represent different approaches to API design, describing rules and standards for how an API should interact with other applications. SOAP is a protocol while REST is a set of constraints that constitute an architectural style. Both use HTTP to exchange information.

REST is often considered a simpler alternative to SOAP because it is lightweight, flexible, transparent and relatively easy to use; SOAP requires users to write more code to complete each task than REST requires.

SOAP is more deterministic and robust (due to type checking), and proponents make the case that it is easier to use because of the SOAP support built into many development tools.³ SOAP features built-in compliance, and developers often consider it a more secure protocol, better suited for situations with strict data integrity requirements.

RESTful systems support messaging in different formats, such as plain text, HTML, YAML, XML and JSON, while SOAP only allows XML. Each has their strengths, and the “right choice,” might depend on use case. However, the ability to support multiple formats for storing and exchanging data is one of the reasons REST is a prevailing choice for building public APIs.

REST vs. GraphQL

GraphQL is a query language and API runtime that Facebook developed internally in 2012 before it became [open source](#) in 2015. GraphQL and REST are both stateless, use a client/server model and use HTTP. GraphQL solves for some limitations of REST, for example, providing the ability to more accurately target wanted resources with a single request.

REST APIs follows a fixed structure, and always return a whole data set for a specified object. If the request is more complex, spanning multiple resources, for example, the client must submit separate requests for each resource. These limitations can lead to under or over-fetching issues.

Neither REST nor GraphQL APIs are inherently superior. They’re different tools that are suited to different tasks.

REST is generally easier to implement and can be a good choice when a straightforward, cacheable communication protocol with stringent access controls is preferred (for public-facing e-commerce sites like Shopify and GitHub, as one example).

GraphQL APIs enable more flexible, efficient data fetching, which can improve system performance and ease-of-use for developers. These features make GraphQL especially useful for building APIs in complex environments with rapidly changing front-end requirements.⁴

APIs, web services and microservices



APIs vs. web services

A web service is an internet software component that facilitates data transfers over a network. Because a web service exposes an application's data and functionality to other applications, in effect, every web service is an API. However, not every API is a web service.

APIs are any software component that serves as an intermediary between two disconnected applications. While web services also connect applications, they require a network to do so. Web services are typically private and only approved users can access them.

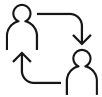
APIs, microservices and cloud-native development

[Microservices](#) is an architectural style that divides an application into smaller, independent components, often connected by using REST APIs. Building an application as a collection of separate services enables developers to work on one application component independent of the others, and makes applications easier to test, maintain and scale.

Microservices architecture has become more prevalent with the rise of [cloud computing](#) and, together with [containers](#) and [Kubernetes](#), is foundational to cloud-native application development.

API benefits

APIs simplify the design and development of new applications and services, and the integration and [management](#) of existing ones. They also offer significant benefits to [developers](#) and organizations at large.



Improved collaboration

The average enterprise uses almost [1,200 cloud applications](#) [↗](#), many of which are disconnected. APIs enable [integration](#) so that these platforms and apps can seamlessly communicate with one another. Through this integration, companies can automate workflows and improve workplace collaboration. Without APIs, many enterprises would lack connectivity, causing information silos that compromise productivity and performance.



Accelerated innovation

APIs offer flexibility, allowing companies to make connections with new business partners and offer new services to their existing market. This flexibility also enables companies to access new markets that can boost returns and drive digital transformation.

For example, the company Stripe began as an API with just seven lines of code. The company has since worked with many of the biggest enterprises in the world. Stripe has diversified to offer loans and corporate cards, and received a [recent valuation](#) [↗](#) of USD 65 billion .



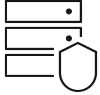
Data monetization

Many companies choose to offer APIs for free, at least initially, so that



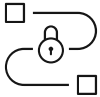
they can build an audience of developers around their brand and forge relationships with potential partners. If the API grants access to valuable digital assets, a business monetizes it by selling access. This practice is referred to as the API economy.

When [AccuWeather](#)  started its self-service developer portal to sell a wide range of API packages, it took just 10 months to attract 24,000 developers, selling 11,000 API keys. This move helped to build a thriving community in the process.



System security

APIs separate the requesting application from the infrastructure of the responding service and offer layers of security between the two as they communicate. For example, API calls typically require authentication credentials. HTTP headers, cookies or query strings can provide additional [security](#) during data exchange. An [API gateway](#) can control access to further minimize security threats.



User security and privacy

APIs provide added protection within a network. They can also provide another layer of protection for personal users. When a website requests a user's location (a location API provides this information), the user can decide whether to allow or deny this request.

Many web browsers and desktop and mobile operating systems have built-in permission structures. When an app must access files through an API, operating systems such as iOS, macOS, Windows and Linux use permissions for that access.

Report

APIs: The Key to AI and Application Connectivity

Learn how modern API management delivers control, security, and AI-driven insights to help organizations scale innovation while safeguarding data.

[Read the report](#)



Resources



Report

Simplify API Management at Scale

Discover the essential capabilities of effective API management and how IBM API Connect unifies all your APIs into one secure, scalable platform.

[Read the report](#)



eBook

Build Resilient Connectivity Across Clouds

Explore how IBM Hybrid Cloud Mesh simplifies app-centric connectivity to accelerate integration and reduce complexity in hybrid and multi-cloud environments.

[Read the eBook](#)



Guide

Want to modernize your application architecture? Start here.

This practical eBook explains how DevOps can help developers and architects deliver applications faster, more reliably, and with less stress.

[Read the guide](#)



Report

Cost of a Data Breach Report 2025



Data breach costs have hit a new high. Get up-to-date insights into cybersecurity threats and their financial impacts on organizations.

[Read the report](#)



Report

Accelerate Your Data Tier with Parallel File Systems

IDC insights on how parallel file systems boost throughput and reliability in the data tier for high-demand, multi-tier applications.

[Read the report](#)



Report

IBM X-Force 2025 Threat Intelligence Index

Gain insights from IBM X-Force experts on emerging attack trends and learn how to strengthen cyber resilience across the security lifecycle.

[Read the report](#)



eBook

Leverage Analytics Across Every Tier

Learn how analytics in the data tier can inform application logic and front-end experiences for smarter decision-making.

[Read the eBook](#)



Report

Create Responsible AI Systems with End-



to-End Governance

Gain a structured approach to managing risk, trust, and compliance in AI systems. Learn how governance frameworks apply across layered cloud architectures.

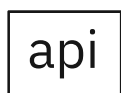
[Read the report](#)



1 / 2



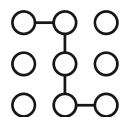
Related solutions



IBM API Connect®

Seamlessly develop, manage, secure and socialize all your application programming interface (API) types, wherever they live.

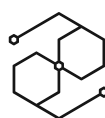
[Explore API Connect](#) →



API development with IBM API Connect

Build, standardize and secure new and existing APIs with ease using IBM API Connect.

[Create new APIs](#) →



API management

Drive new engagement models and channels by accelerating your API-led digital transformation.

[Explore API management](#) →

Take the next step



IBM API Connect® supports all modern application programming interface (API) types while bolstering security and governance. Generative AI (Gen AI) capabilities automate manual tasks, saving time and helping ensure quality.

Explore API Connect



Explore the product tour

