API

* Appplication program Interface, is a set of rules or protocols that let software applications communicate with each other to exchange data, features and functionality.
* APIs simplify application development by allowing developers to integrate data, services and capabilities from other applications, instead of developing them from scratch.
* APIs also give application owners a simple, secure way to make their application data and functionality available to internal departments within their organizations.
* Application owners can also share or market that data and functionality to business partners or third parties.

How an API works

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 ecommerce site, they may be prompted 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 calls to retrieve information—also known as a 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 makes a call to the external program or web server, in this 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, here the product website.

While the data transfer will differ depending on the web service being 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 benefits

* APIs simplify design and development of new applications and services, and integration and management of existing ones. But they offer other significant benefits to developers and organizations at large.

1. Improved collaboration
2. Accelerated innovation
3. Data monetization
4. System security
5. End-user security and privacy

Types of APIs:

* Today most APIs are web APIs that expose an application's data and functionality over the internet. Here are the four main types of web API:

A **web API** is an API that can be accessed using the HTTP protocol. This is a broad category—really too broad to be very useful. Not all APIs are web APIs; some APIs are used only to communicate between two applications on the same computer, never making use of a web connection. But in practice, when developers talk about APIs, they are almost always talking about web-based APIs used to communicate between two computers connected remotely over the internet.

* **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** connect strategic business partners. Typically, developers access these APIs in self-service mode through a public [API developer portal](https://www.ibm.com/products/api-connect/developer-portal). Still, they need to complete an onboarding process and get login credentials to access partner APIs.
* **Internal APIs** remain hidden from external users. These private APIs aren't available for users outside of the company and are instead intended to improve productivity and communication across different internal development teams.
* **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 performing a single task may require information from several sources.

API protocols

As the use of web APIs has increased, it has lead to the development of certain protocols. These protocols 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):** Built with XML, SOAP enables endpoints to send and receive data through SMTP and HTTP. SOAP APIs make it easier to share information between apps or software components that are running in different environments or written in different languages.
* **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 much simpler, and relatively lightweight in that it uses minimum bandwidth.
* **JSON-RPC:** Like XML-RPC, JSON-RPC is a remote procedure call, but JSON (JavaScript Object Notation) is used instead of XML to transfer the data.
* **REST (Representational State Transfer):** REST is a set of web API architecture principles. [REST APIs](https://www.ibm.com/topics/rest-apis)—also known as a RESTful API)—are APIs that adhere to certain REST architectural constraints. It’s possible to build RESTful APIs with SOAP protocols, but the two standards are usually viewed as competing specifications.

Traditionally, API referred to an interface connected to an application created with any of the low-level programming languages, such as JavaScript. Modern APIs, however, adhere to REST principles and the JSON format. They are typically built for HTTP, resulting in developer-friendly interfaces that are easily accessible and widely understood by applications written in [Java](https://www.ibm.com/topics/java), Ruby, Python and many other languages.