DevNet Associate v1.0

Understanding and Using APIs / Introducing APIs

Introducing APIs

4.1.1

What is an API?

An API allows one piece of software talk to another. An API is analogous to a power outlet. Without a power outlet, what would you have to do to power your laptop?

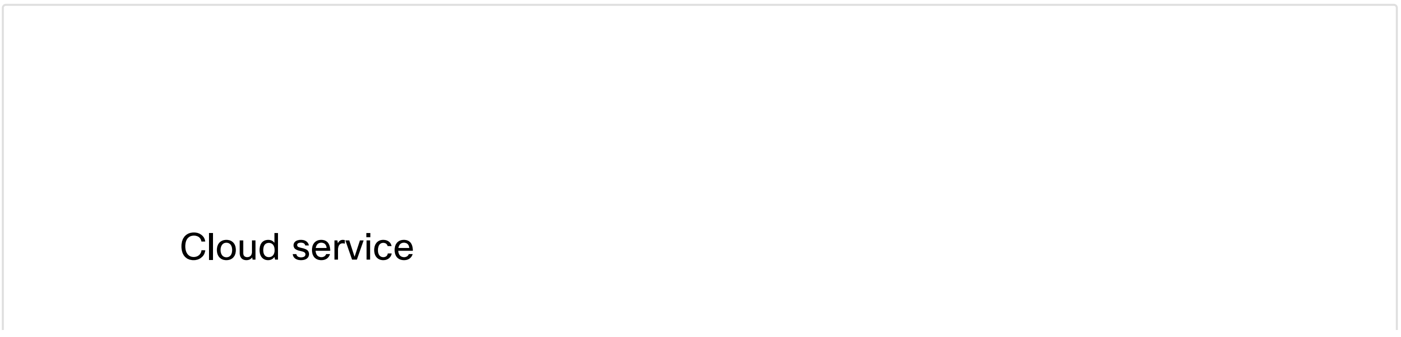
- Open the wall
- Unsheath wires
- Splice wires together
- Understand all the wires in the wall

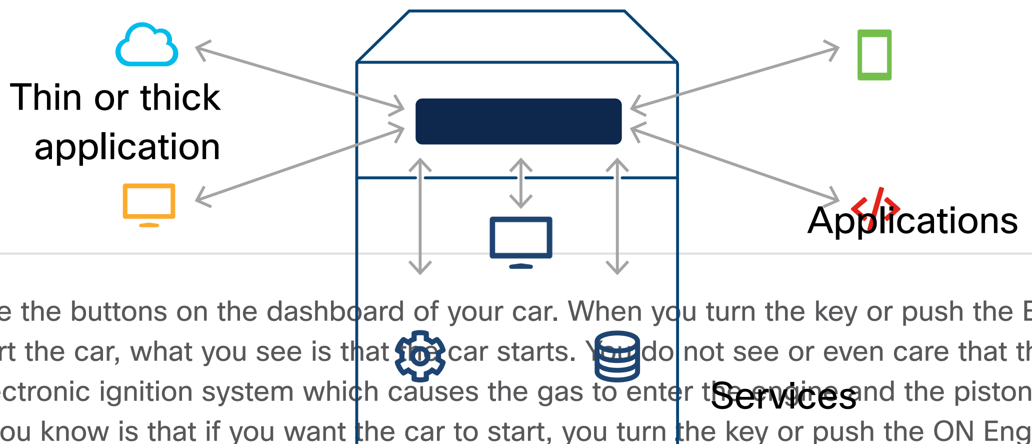
An API defines how a programmer can write one piece of software to talk to an existing application's features or even build entirely new applications.

An API can use common web-based interactions or communication protocols, and it can also use its own proprietary standards. A good example of the power of using an API is a restaurant recommendation app that returns a list of relevant restaurants in the area. Instead of creating a mapping function from scratch, the app integrates a third-party API to provide map functionality. The creator of the API specifies how and under what circumstances programmers can access the interface.

As part of this process, the API also determines what type of data, services, and functionality the application exposes to third parties; if it is not exposed by the API, it is not exposed, period. (That is, assuming security is properly configured!) By providing APIs, applications can control what they expose in a secure way.

Example of different types of API integrations.





Think of it like the buttons on the dashboard of your car. When you turn the key or push the Engine ON button to start the car, what you see is that the car starts. You do not see or even care that the engine starts the electronic ignition system which causes the gas to enter the engine and the pistons to start moving. All you know is that if you want the car to start, you turn the key or push the ON Engine button. If you push a different button, such as the radio ON button, the car will not start because that was not the definition that the car (application) defined for starting the engine. The car (application) itself can do many more things than start the engine, but those things are not exposed to the driver (third-party user).

4.1.2

Why use APIs?

APIs are usually built to be consumed programmatically by other applications, which is why they are called Application **Programming** Interfaces. But they do not have to be used programmatically; they can also be used by humans who want to interact with the application manually.

Here are just a few of the many use cases for APIs:

- **Automation tasks -** Build a script that performs your manual tasks automatically and programmatically. *Example*: You are a manager and on the last day of every pay period, you need to manually log into the portal and download the timecard for each individual who reports to you. Then you need to manually add up the number of hours each person worked so that you can see how much to pay them. If you create an automated script that calls the portal's API to get the data from each timecard and calculate the total from that data, your script can print out a list of total hours worked per person in a user friendly format. Think about how much time that would save!
- **Data integration -** An application can consume or react to data provided by another application. *Example*: Many e-commerce websites use payment services which are accessed via a REST API. The payment interface lets a vendor site transmit relatively low-value data, such as a description of object purchased and the price. The user then independently authenticates to the payment service to confirm the payment. This allows the vendor to receive payment without needing direct access to customer credit card data. The vendor gets back a confirmation code for use by accounting.
- **Functionality -** An application can integrate another application's functionality into its product. *Example*: Many online services, such as Yelp and Uber, exchange data with Google Maps to create and optimize travel routes. They also embed functionality from Google Maps within their own apps and websites to present realtime route maps.

4.1.3

Why are APIs so popular?

APIs have existed for decades, but exposure and consumption of APIs has grown exponentially in the last 10 years or so. In the past, applications were locked down and the only integration between them was through predetermined partnerships. As the software industry has grown, so has the demand for integration. As a result, more and more applications have started to expose bits and pieces of themselves for third-party applications or individuals to use.

Most modern APIs are designed into the product rather than being an afterthought. These APIs are usually thoroughly tested, just like any other component of the product. These APIs are reliable and are sometimes even used by the product itself. For example, sometimes an application's user interface is built on the same APIs that are provided for third parties.

The popularity of easier and more simplified coding languages such as Python have made it possible for non-software engineers to build applications and consume these APIs. They get the results they want without having to hire expensive development talent.

4.0 Introduction

4.2 API Design Styles