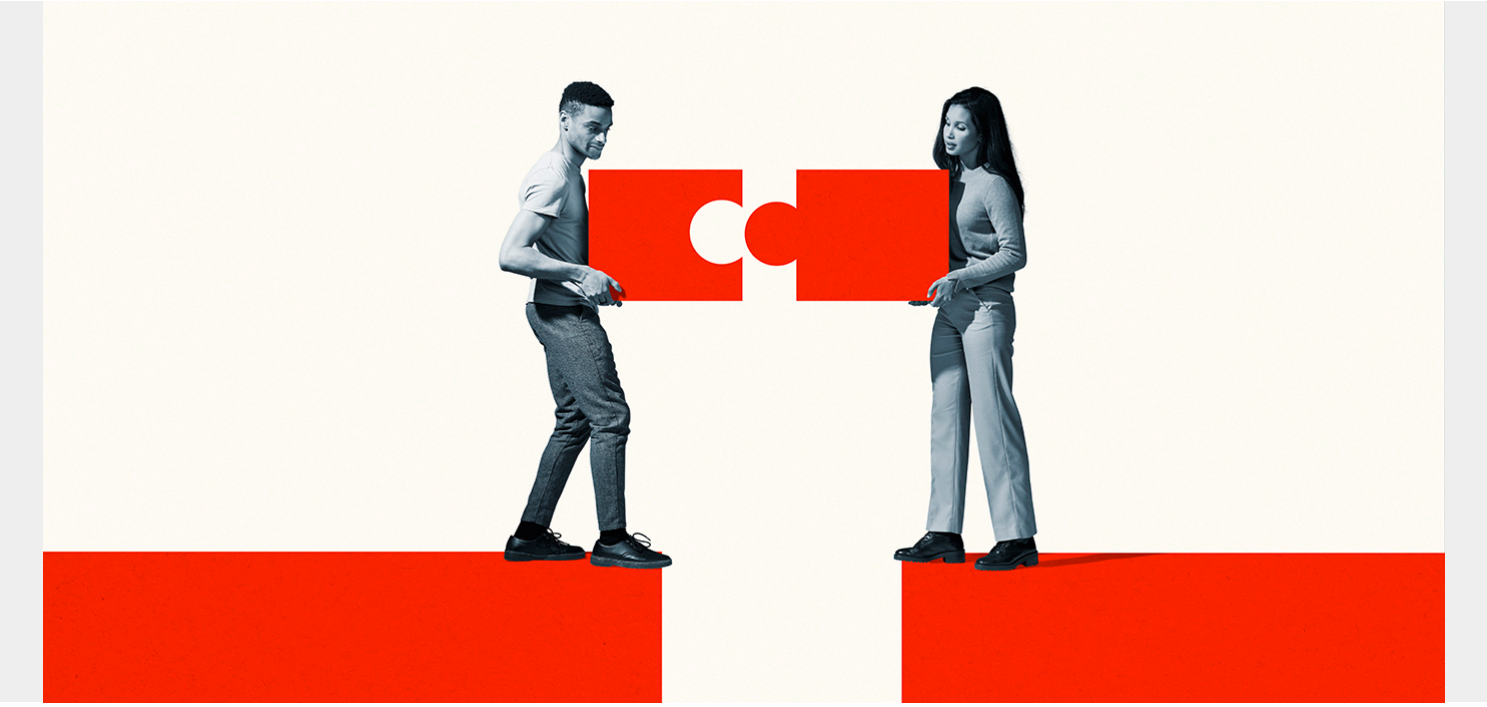
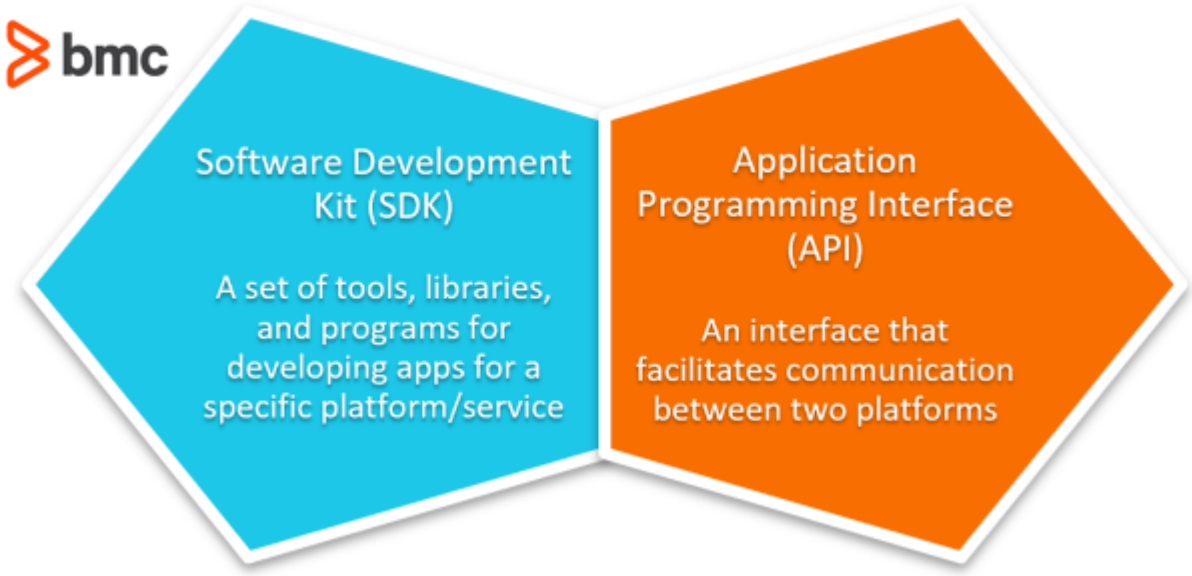
SDK VS API: WHAT'S THE DIFFERENCE?



Software Development Kits (SDKs) and Application Programming Interfaces (APIs) are two indispensable tools in modern software development. Both aim to enhance and extend the capabilities of the software. However, an SDK and API have different use cases and targets to address a specific concern in software development pipelines.

In this article, we will demystify SDKs and APIs to understand what they are, how they can be used, and how they relate to each other.



What is a Software Development Kit?

An SDK is a set of tools, libraries, and programs that can be used to develop applications for a specific platform or service.

Unlike general programming languages that allow users to develop software for any supported platform, SDKs enable users to develop platform-specific software utilizing all the features and functionality of those platforms. Most SDKs are language agnostic, meaning there will be different SDKs for different programming languages for the same platform.

Typically, an SDK includes the following resources:

- **Code libraries** are functions that allow users to interact with the underlying platform and utilize its capabilities
- **Compiler** translates the programming language code and converts source code to object code.
- **Debuggers** help identify platform or service-specific issues.
- **Documentation** includes usage instructions, how-to guides, best practices, and code samples to aid in your development.

What an SDK brings is simplicity. Developers can simply download and install an SDK and start developing for the specified platforms via their integrated development environments. APIs can also be a part of an SDK to bring interfacing functionality.

Common SDKs: Examples & usage

- Google's Android SDK and Apples' iOS SDKs for mobile application development for Android and iOS platforms.
- Microsoft's .NET SDK and macOS SDK for desktop application development for Microsoft Windows and Apple macOS.
- Amazon Web Services SDK, Microsoft Azure SDK, and Google Cloud SDK for the development of their respective cloud platforms.
- OpenAI SDK, Qualcomm Neural Processing SDK for AI, and Vertex AI for developing AI applications.
- Stripe SDK and Paypal SDK to integrate payment services into your applications.

Benefits of software development kits

- **Direct access to the functionality and features** of the SDK platform and the ability to use them within your application.
- **Straightforward integration and faster development.** The SDK provides all the required integration and development libraries.
- **No reinventing the wheel.** SDK eliminates the need for developers to create functions from scratch to interact with a platform as they can simply call upon the SDK for that, drastically reducing the development time.
- **Efficient resourcing and cost.** This shorter and faster development cycle leads to efficient resource management and reduced development costs.
- **Developer's choice**. Developers can use their preferred language for the development as most platforms provide SDKs for different programming languages.

- **Great support.** SDKs come with documentation and code samples, allowing developers to easily learn how to use them. Additionally, you can easily find answers to development matters with a simple search query as most common SDKs have a wide community.

What is an Application Programming Interface?

Application Programming Interface or API is an interface that facilitates communication between two platforms.

The primary goal of an API is to standardize the way third parties interact with a piece of software—without using custom connectors or integrations. An API allows external parties to utilize or integrate the services provided by the software in their applications.

The Application Programming Interface can dictate what kind of functionality is exposed and what information can be exchanged between the underlying software or service and the API user (the third party who consumes the API). Typically, an API will consist of two components.

- **API.** The interface that facilitates communication and data sharing.
- **Documentation.** Information on how to utilize the API, endpoints, information on authentication and authorization, data exposed, etc.

There are different kinds of APIs depending on the use case and functionally. Some common types of API architecture are:

- **Web API.** These are interfaces used to interact with standard web components like web browsers, devices, or custom services.
- **REST API.** Utilize the REST architectural style to facilitate the API and can be used with XML, JSON, and plain text. These APIs are also known as RESTful APIs. They have become the common choice for most web applications and rely on HTTP/S for communications.
- **SOAP API.** This highly extensible standard uses XML to provide messaging services and facilitate communication. While SOAP can be more complex than REST, it provides heightened data security, privacy, and transport-independent API implementation.
- **RPC.** These types of APIs are used to invoke actions or processes and get the desired outcome. Typically, these APIs will accept an API call with some parameters, carry out an action, and return the result. RPC can use JSON or XML, and they are also known as JSON-RPC or XML-RPC within development communities.

How APIs work: basic functionality

The goal of any API is to facilitate communication between two different platforms or services.

Assume you have an online booking platform for hotels. You need to know the availability of rooms with different hotel providers using this platform. As bookings happen in real-time across multiple platforms, you must have an updated inventory of available rooms at all times to facilitate a smooth booking process for your end-users.

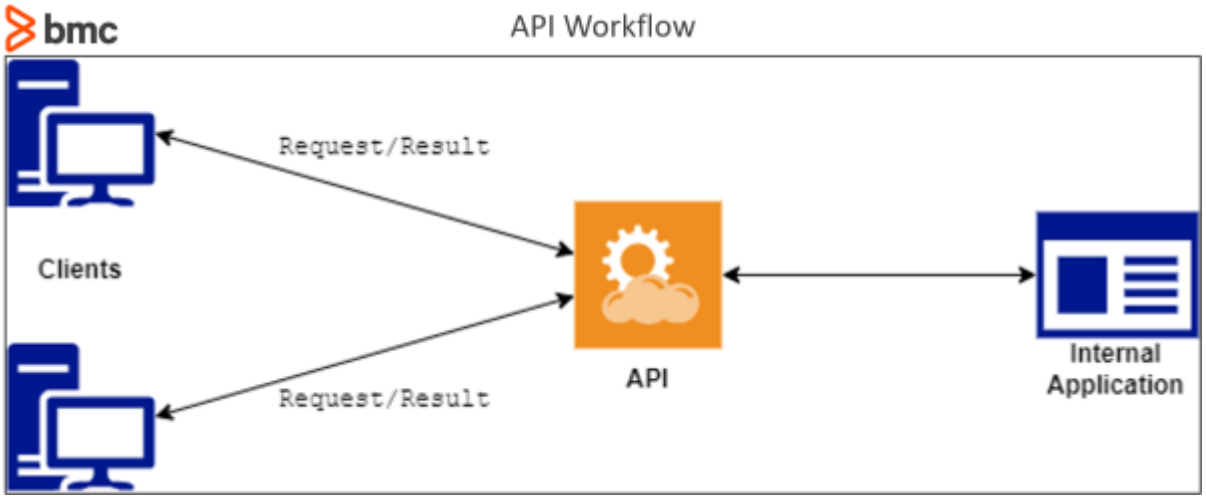
This is where APIs come into play. Each hotel provider will have an API that exposes the availability of their properties. You can query these API endpoints through your booking platform when an end-user requests a booking to a particular property. The API will then return the availability and any additional information requested by the booking platform. Then your platform will again utilize the

API to confirm the booking and inform the hotel provider about the booking so that he can update his internal systems.

The workflow of an API

Let's break it down:

1. A client application requests a specific API endpoint to carry out a specific functionality to obtain information.
2. The API captures this request. Then it checks for authentication and authorization and relays the request with any parameters to the internal platform.
3. Depending on the request type, the platform will process the request and communicate the result back to the client application which sent the request. For example, if it is a data request, the platform will send back the requested data, or if the request is to carry out a specific function, it will return the result of that specific function.



Common API use cases

- Mapping and weather APIs add customized maps and weather predictions to software applications or websites. Examples include Google Maps API, OpenStreetMap API, and OpenWeatherMap API.
- Payment APIs to facilitate payments across multiple providers. Examples: Stripe, Square, PayPal, KeyPay, and Bank APIs.
- Scientific Services like Open Science Framework API to access open-source research projects and data.
- Internal APIs facilitate communication between different components of an application. This is especially the case with distributed architectures like microservices, where smaller individual components within an application use APIs to facilitate communication between them.

(Understand how APIs & microservices work together.)

Benefits of Application Programming Interfaces

- **Fast integration.** Provide fast integration between different software and services.
- **Secure without customization.** Securely exposing platform capabilities and data without requiring custom integrations.

- **Enables distributed architectures.** The ability to facilitate distributed software architectures where internal communications are handled via APIs.
- **Increased productivity** in the development cycles with high reusability offered by APIs.
- **Reduced costs** as no custom development efforts are required to interact with API endpoints.
- **Easy analytics.** Easy integration of reporting and data analytics functionality through data queried via APIs.
- **Platform agnostic.** Any type of service, device, or platform can utilize APIs as they are platform-agnostic.

Choosing between SDK or APIs?

There is no need to select only one between SDKs and APIs for your development. In fact, both of them are essential for all modern applications to facilitate core services through your application. As mentioned:

- An SDK provides a complete development kit for software development for building applications for a specified platform, service, or language.
- An API is used to facilitate communication between two platforms.

Developers can create their applications using an SDK and use APIs to integrate with third-party platforms or services to bring additional functionality to the application. SDKs themselves will include APIs that facilitate interactions with the targeted platform.

Additionally, Software Development Kits can be used to create APIs that enable external parties to interface with your application.

SDKs & APIs are integral to software development

SDKs and APIs have become integral parts of modern software development. With the ever-increasing complexity of development requirements, SDKs and APIs aim to simplify the development process by offering the necessary tools to develop applications while utilizing the capabilities of the targeted platforms and services.

In this cloud-first world, both these tools have become invaluable for developing applications across different platforms and integrating with various services.

Related reading

- BMC DevOps Blog
- Getting Authentication Access Tokens for Microsoft APIs
- API/Developer Portals: How To Create Great API Portals
- The Role of Microservices in DevOps
- Functions as a Service (FaaS): A Beginner's Guide
- Today's Top Trends in Software Development