**API**

API stands for Application Programming Interface. This interface allows people to further build upon another application’s functionality and data. One might understand them as building blocks you can use to make almost anything as they can be found in everything from Spotify to Yahoo Finance.

The API frameworks allow developers to perform tasks that aren’t all that different from everyday events. For instance, think of giving an order to a server, that server putting your order in, and then bringing back the order when it’s ready. This step-by-step process returns the desired outcome: a tasty meal (in this case). A web-based example might be someone signing up to a new e-Commerce site by using their Facebook account.

**web service**

A web service, in contrast to an API, functions more like a resource that’s available using the internet. The network-based resource can be applied to specific tasks, but they require a network to function. This means that all web services are APIs, but only some APIs are web services.

A web service works by supporting interoperable machine-to-machine communication using a network. As such, web services tend to be connected with SOA or Service Oriented Architecture. This allows for different features to be separated then made available as various services within a network.

**KEY DIFFERENCE**

* Web service is a collection of Open Source protocols and standards used for exchanging data between systems or applications whereas API is a software interface that allows two applications to interact with each other without any user involvement.
* Web service is used for REST, SOAP and XML-RPC for communication while API is used for any style of communication.
* Web service supports only HTTP protocol whereas API supports HTTP/HTTPS protocol.
* Web service supports XML while API supports XML and JSON.
* All Web services are APIs but all APIs are not web services.

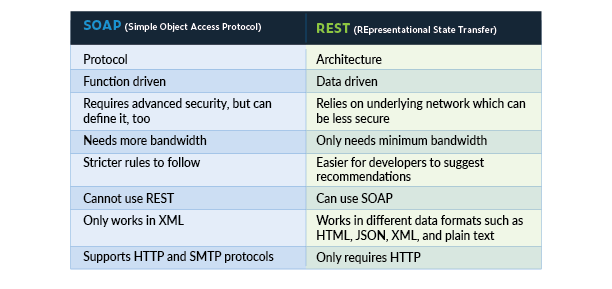
Furthermore, web services are not protocol-agnostic like APIs. APIs can use any design style or protocol, but web services are restricted mostly to SOAP or Simple Object Access Protocol.

Public APIs are often also open source and more transparent about their documentation. Web services sacrifice that transparency for more specific data, partners, and security. However, API security remains a challenge.

**Types of Web Services**

Web services should be implemented in various ways. The two types of widely used web services are SOAP and RESTful web services.

**SOAP** – SOAP is a protocol which was designed before REST came into the picture. The main idea behind creating SOAP was to ensure that programs built on different platforms and programming languages could securely exchange data.

****  
**REST** – This was designed specifically for working with components such as media components, files, or even objects on a particular hardware device. Any web service which is defined on the principles of REST can be called a RESTful web service.REST uses the normal HTTP verbs of GET, POST, PUT and DELETE for working with the required components.

**REST** stands for Representational State Transfer. It is developed by Roy Thomas Fielding who also developed HTTP. The main goal of RESTful web services is to make web services more effective. RESTful web services try to define services using the different concepts that are already present in HTTP. REST is an architectural approach, not a protocol.

It does not define the standard message exchange format. We can build REST services with both XML and JSON. JSON is more popular format with REST. The key abstraction is a resource in REST.

Something else to keep in mind: Headers and parameters are also important in the HTTP methods of a RESTful API HTTP request, as they contain important identifier information as to the request's metadata, authorization, uniform resource identifier (URI), caching, cookies, and more. There are request headers and response headers, each with their own HTTP connection information and status codes.

Though, because REST also intends to make the web (internet) more streamline and standard, he advocates using REST principles more strictly. And that’s from where people try to start comparing REST with web (HTTP). Roy fielding, in his dissertation, nowhere mentioned any implementation directive – including any protocol preference and HTTP. Till the time, you are honoring the 6 guiding principles of REST, you can call your interface RESTful.

In simplest words, in the REST architectural style, data and functionality are considered resources and are accessed using Uniform Resource Identifiers (URIs). The resources are acted upon by using a set of simple, well-defined operations. The clients and servers exchange representations of resources by using a standardized interface and protocol – typically HTTP.

Resources are decoupled from their representation so that their content can be accessed in a variety of formats, such as HTML, XML, plain text, PDF, JPEG, JSON, and others. Metadata about the resource is available and used, for example, to control caching, detect transmission errors, negotiate the appropriate representation format, and perform authentication or access control. And most importantly, every interaction with a resource is stateless.

**SOAP** is a standard communication protocol system that permits processes using different operating systems like Linux and Windows to communicate via HTTP and its XML. SOAP based APIs are designed to create, recover, update and delete records like accounts, passwords, leads, and custom objects.

These offers over twenty different kinds of calls that make it easy for the API developers to maintain their accounts, perform accurate searches and much more. These can then be used with all those languages that support web services.

SOAP APIs take the advantages of making web based protocols such as HTTP and its XML that are already operating the all operating systems that are why its developers can easily manipulate web services and get responses without caring about language and platforms at all.

SOAP relies heavily on XML, and together with schemas, defines a very strongly typed messaging framework.

Every operation the service provides is explicitly defined, along with the XML structure of the request and response for that operation.

Each input parameter is similarly defined and bound to a type: for example an integer, a string, or some other complex object.

All of this is codified in the WSDL – Web Service Description (or Definition, in later versions) Language. The WSDL is often explained as a contract between the provider and the consumer of the service. In programming terms the WSDL can be thought of as a method signature for the web service.

The first SOAP specification was published in 2000. An earlier version, released in 1998, was known as XML-RPC and had a more focused feature set. Like SOAP, XML-RPC allows for remote procedure calls via XML. XML-RPC specifically only uses HTTP to transport the data. While this is the common protocol for SOAP, as well, SOAP can technically use any protocol.

It took three years for the SOAP specification to reach recommendation stage. Quickly it became the most common approach to web services. Prior to SOAP, there was not a standards-based approach to creating programmable interfaces for exchanging data between systems. SOAP helped shepherd innovation both within the enterprise and encouraged the first wave of public APIs from companies like eBay, Salesforce, and Amazon.

**SOAP vs REST**

REST and Simple Object Access Protocol (SOAP) offer different methods to invoke a web service. REST is an architectural style, while SOAP defines a standard communication protocol specification for XML-based message exchange. REST applications can use SOAP.

RESTful web services are stateless. A REST-based implementation is simple compared to SOAP, but users must understand the context and content being passed along, as there's no standard set of rules to describe the REST web services interface. REST services are useful for restricted profile devices, such as mobile, and are easy to integrate with existing websites.

* REST API has no has no official standard at all because it is an architectural style. SOAP API, on the other hand, has an official standard because it is a protocol.
* REST APIs uses multiple standards like HTTP, JSON, URL, and XML while SOAP APIs is largely based on HTTP and XML.
* As REST API deploys multiple standards, so it takes fewer resources and bandwidth as compared to SOAP that uses XML for the creation of Payload and results in the large sized file.
* The ways both APIs exposes the business logics are also different. REST API takes advantage of URL exposure like @path("/WeatherService") while SOAP API use of services interfaces like @WebService.

SOAP provides the following advantages when compared to REST:

* Language, platform, and transport independent (REST requires use of HTTP)
* Works well in distributed enterprise environments (REST assumes direct point-to-point communication)
* Standardized
* Provides significant pre-build extensibility in the form of the WS\* standards
* Built-in error handling
* Automation when used with certain language products

REST is easier to use for the most part and is more flexible. It has the following advantages when compared to SOAP:

* Uses easy to understand standards like swagger and Open API Specification 3.0
* Smaller learning curve
* Efficient (SOAP uses XML for all messages, REST mostly uses smaller message formats like JSON)
* Fast (no extensive processing required)
* Closer to other Web technologies in design philosophy

GraphQL is a query language and server-side runtime APIS that prioritizes giving clients exactly the data they request and no more.

API developers use GraphQL to create a **schema** to describe all the possible data that clients can query through that service.

A GraphQL schema is made up of object types, which define which kind of object you can request and what fields it has.

As **queries** come in, GraphQL validates the queries against the schema. GraphQL then executes the validated queries.