WEB SERVICES

WEB SERVICES IS A STANDARDIZED WAY OR MEDIUM TO PROPAGATE COMMUNICATION BETWEEN THE CLIENT AND SERVER APPLICATIONS ON THE WORLD WIDE WEB.

Web services provide a common platform that allows multiple applications built on various programming languages to have the ability to communicate with each other. Web Services refers to the technologies that allow for making connections. Services are what you connect together using Web Services. A service is the endpoint of a connection. Also, a service has some type of underlying computer system that supports the connection offered. The combination of services—internal and external to an organization—make up a service-oriented architecture.

To summarize, a complete web service is, therefore, any service that −

* Is available over the Internet or private (intranet) networks
* Uses a standardized XML messaging system
* Is not tied to any one operating system or programming language
* Is self-describing via a common XML grammar
* Is discoverable via a simple find mechanism

**SERVICE-ORIENTED ARCHITECTURE (SOA) DEFINITION**

A service-oriented architecture is essentially a collection of services. These services communicate with each other. The communication can involve either simple data passing, or it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed.

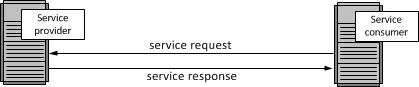
Service-oriented architectures are not a new thing. The first service-oriented architecture for many people in the past was with the use DCOM or Object Request Brokers (ORBs) based on the CORBA specification.

**SERVICES**

If a service-oriented architecture is to be effective, we need a clear understanding of the term service. A service is a function that is well-defined, self-contained, and does not depend on the context or state of other services.

**CONNECTIONS**

The technology of [Web Services](https://www.service-architecture.com/articles/web-services/web_services_definition.html) is the most likely connection technology of service-oriented architectures. The following figure illustrates a basic service-oriented architecture. It shows a service consumer at the right sending a service request message to a service provider at the left. The service provider returns a response message to the service consumer. The request and subsequent response connections are defined in some way that is understandable to both the service consumer and service provider. How those connections are defined is explained in [Web Services Explained.](https://www.service-architecture.com/articles/web-services/web_services_explained.html) A service provider can also be a service consumer.



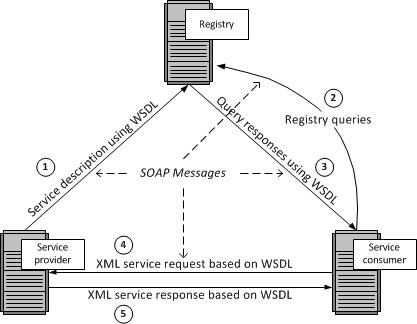
**HISTORY OF THE WEB SERVICES SPECIFICATION**

Web Services Description Language (WSDL); Universal Description and Discovery (UDDI); and SOAP formed the original Web Services specification. This section provides a history.

**Web Services Description Language (WSDL)**

The Web Services Description Language (WSDL) forms the basis for the original Web Services specification. The following figure illustrates the use of WSDL. At the left is a service provider. At the right is a service consumer. The steps involved in providing and consuming a service are:

1. A service provider describes its service using WSDL. This definition is published to a repository of services. The repository could use Universal Description, Discovery, and Integration (UDDI). Other forms of directories could also be used.
2. A service consumer issues one or more queries to the repository to locate a service and determine how to communicate with that service.
3. Part of the WSDL provided by the service provider is passed to the service consumer. This tells the service consumer what the requests and responses are for the service provider.
4. The service consumer uses the WSDL to send a request to the service provider.
5. The service provider provides the expected response to the service consumer.



**HOW DOES A WEB SERVICE WORK?**

A web service enables communication among various applications by using open standards such as HTML, XML, WSDL, and SOAP. A web service takes the help of −

* XML to tag the data
* SOAP to transfer a message
* WSDL to describe the availability of service.

You can build a Java-based web service on Solaris that is accessible from your Visual Basic program that runs on Windows.

You can also use C# to build new web services on Windows that can be invoked from your web application that is based on JavaServer Pages (JSP) and runs on Linux.

**Example**

Consider a simple account-management and order processing system. The accounting personnel use a client application built with Visual Basic or JSP to create new accounts and enter new customer orders.

The processing logic for this system is written in Java and resides on a Solaris machine, which also interacts with a database to store information.

The steps to perform this operation are as follows −

* The client program bundles the account registration information into a SOAP message.
* This SOAP message is sent to the web service as the body of an HTTP POST request.
* The web service unpacks the SOAP request and converts it into a command that the application can understand.
* The application processes the information as required and responds with a new unique account number for that customer.
* Next, the web service packages the response into another SOAP message, which it sends back to the client program in response to its HTTP request.
* The client program unpacks the SOAP message to obtain the results of the account registration process.

**DEFINING SOAP AND REST**

SOAP communication protocols. SOAP was long the standard approach to web service interfaces, although it’s been dominated by REST in recent years, with REST now representing more than 70% of public APIs according to [Stormpath](https://stormpath.com/blog/rest-vs-soap).  Understand the primary differences between SOAP vs. REST and how each can benefit your organization’s goals. (Simple Object Access Protocol) and REST (Representational State Transfer) are both web service

**SOAP VS REST: PRIMARY DIFFERENCES**

REST operates through a solitary, consistent interface to access named resources. It’s most commonly used when you’re exposing a public API over the Internet. SOAP, on the other hand, exposes components of application logic as services rather than data. Additionally, it operates through different interfaces. To put it simply, REST accesses data while SOAP performs operations through a more standardized set of messaging patterns. Still, in most cases, either REST or SOAP could be used to achieve the same outcome (and both are infinitely scalable), with some differences in how you’d configure it.

SOAP was originally created by Microsoft, and it’s been around a lot longer than REST. This gives it the advantage of being an established, legacy protocol. But REST has been around for a good time now as well. Plus, it entered the scene as a way to access web services in a much simpler way than possible with SOAP by using HTTP.

A screenshot of a social media post

Description generated with very high confidence

**BENEFITS OF REST OVER SOAP**

In addition to using HTTP for simplicity, REST offers a number of other benefits over SOAP:

* REST allows a greater variety of data formats, whereas SOAP only allows XML.
* Coupled with JSON (which typically works better with data and offers faster parsing), REST is generally considered easier to work with.
* Thanks to JSON, REST offers better support for browser clients.
* REST provides superior performance, particularly through caching for information that’s not altered and not dynamic.
* It is the protocol used most often for major services such as Yahoo, Ebay, Amazon, and even Google.
* REST is generally faster and uses less bandwidth. It’s also easier to integrate with existing websites with no need to refactor site infrastructure. This enables developers to work faster rather than spend time rewriting a site from scratch. Instead, they can simply add additional functionality.

Still, SOAP remains the preferred protocol for certain use cases. The general consensus among experts these days is that REST is the typically preferred protocol unless there’s a compelling reason to use SOAP (and there are some cases in which SOAP is preferred).

**BENEFITS OF SOAP OVER REST**

Because you can achieve most outcomes using either protocol, it’s sometimes a matter of personal preference. However, there are some use cases that SOAP tends to be better-suited for. For instance, if you need more robust security, SOAP’s support for WS-Security can come in handy.

* It offers some additional assurances for data privacy and integrity.
* It also provides support for identity verification through intermediaries rather than just point-to-point, as provided by SSL (which is supported by both SOAP and REST).
* Another advantage of SOAP is that it offers built-in retry logic to compensate for failed communications. REST, on the other hand, doesn’t have a built-in messaging system. If a communication fails, the client has to deal with it by retrying. There’s also no standard set of rules for REST. This means that both parties (the service and the consumer) need to understand both content and context.

**OTHER BENEFITS OF SOAP INCLUDE:**

SOAP’s standard HTTP protocol makes it easier for it to operate across firewalls and proxies without modifications to the SOAP protocol itself. But, because it uses the complex XML format, it tends to be slower compared to middleware such as ICE and COBRA.

Additionally, while it’s rarely needed, some use cases require greater transactional reliability than what can be achieved with HTTP (which limits REST in this capacity). If you need ACID-compliant transactions, SOAP is the way to go.

In some cases, designing SOAP services can actually be less complex compared to REST. For web services that support complex operations, requiring content and context to be maintained, designing a SOAP service requires less coding in the application layer for transactions, security, trust, and other elements.

SOAP is highly extensible through other protocols and technologies. In addition to WS-Security, SOAP supports WS-Addressing, WS-Coordination, WS-ReliableMessaging, and a host of other web services standards