# REST vs. SOAP: Stay Calm or Keep Clean?

By **Duncan Brown**, dzone.com
March 21st, 2016

Back in "the day" (which I believe was a Thursday), remote procedure calls (RPCs) were the stuff of dreams--and nightmares. A natural extension of RPCs, some may argue, was the advent of the SOAP communication protocol to allow for the interoperation and integration of remote services. Now, anyone who has worked with SOAP for any amount of time will likely tell you it's a bit of a mixed blessing. While it is very useful in enforcing rigid and consistent message passing over common channels, it also has its fair share of complexities and potential pitfalls ("To WSDL or not to WSDL?", and, "Have fun parsing all that!").

Then we saw the rise of RESTful services--stateless calls typically done over HTTP using a JSON or XML body. Things seem to become a bit simpler, although much less strict in its natural enforcement of schema and communication.

Both SOAP and REST have been utilized to varying degrees of success to help implement what has become has known as Services-Oriented Architecture (SOA), a cornerstone of many an enterprise.

To this day, there remains a mix of both SOAP and REST services being used in various capacities. Enterprises still make extensive use of SOAP services, especially those legacy services that have been in place for 10 or more years. Each has their strengths, but is there (or will there ever be) a clear winner?

In this author's humble opinion, REST has been able to meet 90% of the use cases he has come across. That said, let's have a quick look at the strengths of each to help you decide for yourself:

## SOAP

- SOAP often uses a WSDL (Web Services Description Language) endpoint to help service consumers discover the capabilities and schema of exposed services. This can in turn aid in the auto-generation of model classes to aid in the rapid generation of consumers.

- Systems programming is a frequent use case for SOAP. Asynchronous communications are thereby made easier as the SOAP paradigm often includes consumers that can also act as receivers, and so messaging at later times is perfectly acceptable (try that with traditional REST services over HTTP).

- Schemas help ensure consistency of communication and messages, reducing the potential for "bad" or misinterpreted data. Further to that, the SOAP protocol (and its supporting technologies) are well and thoroughly defined.

## REST

- The overhead for setting up and getting started with REST services (both consuming and producing) is arguably a fair bit less. If you can set up a server-side script that runs on a web server, you are 80% of the way there.

- Statelessness aids not only in testing a REST service but in reducing complexity and memory requirements/overhead for running said service.

- Caching is greatly facilitated, especially for data that changes infrequently.

- By being less adherent to a specific schema, REST is very flexible.

## Conclusion

The purpose of this article has not been to determine the superiority of one method over the other. As with most everything else in the development world, it is a matter of choosing the right tool for the right job.

I see in the foreseeable future a continuation of the use of both SOAP and REST.

However, with the explosion of mobile-based applications and web applications that need to support several different types of clients, the number of REST services has grown dramatically and shows no signs of slowing down. So the number of SOAP implementations that exist may slow down or remain steady, but only by direct comparison to the suitability of REST to the aforementioned web applications.

(InfoQ cites a study from 2011 in which, even then the growth of REST services dwarfs that of SOAP.)

For another, more in-depth comparison, please check out this link.