WS-\* Standards

Abstract

WS-\* is an overarching term to describe the standards governing interactions with web services. These standards allow for interactions between users from any operating system with web services and for developers, provides a method to advertise what they have to offer to clients. This paper will explore the initial development of WS-\* standards, their current usage trends, and future projections.

1. Introduction

WS\* standards are a family of protocols promoting interoperability and communication between dissimilar operating systems. WS\* standards begin with SOAP (Simple Object Access Protocol) which was created by a group of Microsoft engineers and first released in 1999. SOAP serves as the foundation for the XML based WS\* standards that facilitate cross platform communications. It is essentially a template for where to place message headers, actual content, and the desired action when communicating between two applications. We can think of SOAP as the steel support beams for a large office building. SOAP provides structure and show the builders (developers) where walls, floors, and windows should be created. With SOAP to provide the exoskeleton, Web Services Description Language (WSDL) could then be introduced which dictate how each individual application accepts messages (walls, floors, and windows in the example above). The combination of these services dramatically improved the ability of developers to create cross platform applications that could run on any operating system. Remember that unlike the Web of today, in 1999 Microsoft had a monopolistic grip on computing. Publishing standard messaging formats for anyone to be able to communicate with web services opened up application and service development to anyone willing to use that format. An Apple web developer could now interact with Microsoft web services and advertise their own services by using the WSDL standard. This paper will cover initial WS-\* standards and how they operate, current standards, and wrap up with how WS-\* is relevant to E-Business.

2 First-Generation Standards

WS-\* has three initial (first generation standards). Two of them, SOAP and WSDL were introduced earlier, but will be explored in more depth in this section. The third, Universal Description, Discovery, and Integration is far less popular than the other two but will also be touched on.

**SOAP**

SOAP messages are the crux of web services communications. Using XML as its formatting language and HTTP or SMTP for transport, SOAP can traverse networks or the Internet allowing clients to request resources and serves to provide responses in a processable format. Most importantly though is that SOAP is compatible with any operating system. Prior to SOAP, application interoperability was far more onerous and subsequently not as popular. With SOAP’s introduction, developers now had a large incentive to standardize their applications to its format. Why? So that they could communicate with other services.

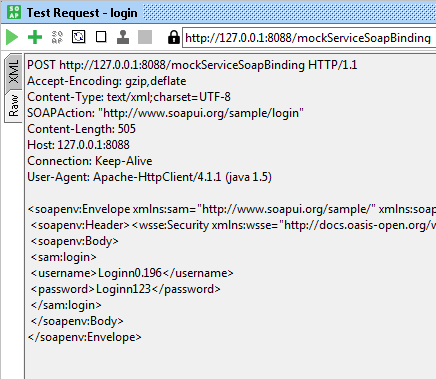


Figure 1 – Example of SOAP message

**WSDL**

Enter WSDL. Now that services had a way to connect to each other, they needed a way to translate the messages that were being sent into an application understandable format. WSDL is XML based and provides a readout of exactly how a service can be called and what it is capable of providing to the client. WSDL messages aren’t the actual messages being exchanged, but instead provides server specific instructions on what services are available and how connecting clients can access them.

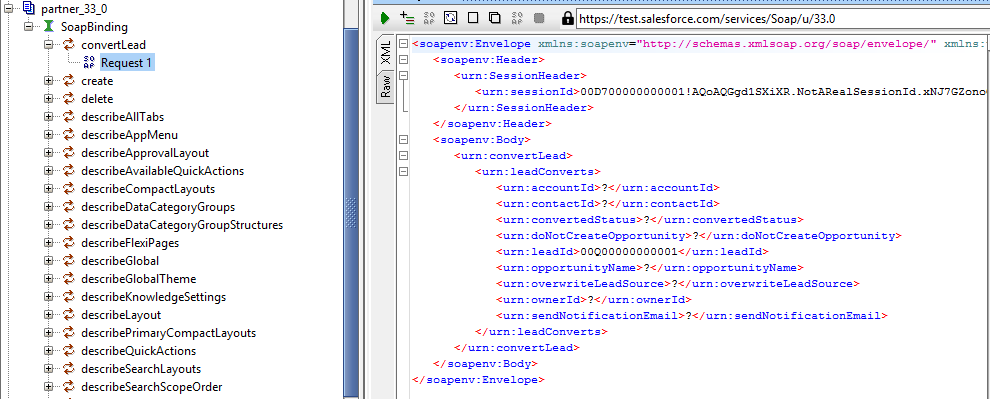


Figure 2 -Example of WSDL Message

**Practical Process Flow**

To illustrate how SOAP, WSDL, XML, and transport level protocols work together let’s consider the example of retrieving data from a database server. When the client computer connects it will read the WSDL file to determine what services are available to the client. In this case it will see that there is a database available. The WSDL file, which is written in XML format, will also contain the datatypes required to successfully interact with the database service. Now, the client will craft a SOAP message to call the database service. This will be sent to the server via a transport level protocol, most likely HTTP/HTTPS. Now the client and server can continue to communicate via HTTP to allow POSTs, GETs, or other operations.

**UDDI**

UDDI is the first generation WS-\* service that really didn’t gain the same popularity as its peers. It made sense for the time it was created, but the growth and scale of the Internet caused it to become largely ignored after a few years. UDDI was intended to be the “yellow pages” for Internet services. They were to be a public repository for web services. Businesses could register the services they offered, and clients would be able to browse for the specific functionality they needed. In terms of computing this makes sense, create and exhaustive registry for similar services so that anyone can search for and access any of them. From a business perspective though, this model doesn’t work. Businesses are constantly competing with each other and providing homogenous services is not how they win customer transactions and market share. IBM, Microsoft, and other maintainers of pubic UDDI registries began shutting them down in 2006 and have continued to decrease support to the point that you can’t find them anymore outside of those for educational purposes.

**The WS-\* Army**

Like many Internet standards and protocols, SOAP and WSDL were not all encompassing when released and have needed continual extensions over time to provide better quality or additional functionality for web services. Four popular extensions are detailed below. While this is not an exhaustive list it clearly illustrates the concept of WS-\* as well as one of the reasons SOAP has a successor.

***WS-Addressing*** – this extension to SOAP includes message-routing data so that messages can be sent independently without relying on a transport protocol. SOAP messages with WS-Addressing enabled contain the additional properties of message destination, source endpoint, reply endpoint, fault endpoint, action, unique message ID in order to deliver the message to the correct endpoint.

***WS-Reliable Messaging*** – provides a method to confirm SOAP message delivery and receipt. If messages are not delivered to the destination successfully, they will report back to the sending node. The report provided will depend on the stringency of delivery assurance options specified.

***WS-Security*** – this extension works in the application layer and provides the ability to protect messages in transit. Signatures, encryption, and other security measures can all be configured using WS-Security. WS-Security contains multiple subset WS-\* standards that provide options for individual security measures. One of them is WS-Policy below.

***WS-Policy*** – more in-depth specification of WS-Security. This allows for endpoints to advertise and enforce policies specific to data formatting, session communications, or basic security controls. Examples of policies could be the presence of a token, type of encryption, and message signing.

3 JSON and REST

As seen above, WS-\* extensions are numerous and multiple must be configured to obtain modern day security settings, routing, and QoS. The larger the XML messages, the longer it takes to send and read. In business, especially E-Business time is of the essence. Representation State Transfer (REST) is an alternative to SOAP and WSDL that is prized for its speed and flexibility. REST APIs allow HTTP calls to interact with service functionality directly and provide several benefits over SOAP style communications. REST calls are stateless, flexible, and cacheable. Caching and statelessness reduce processing time and make web services instantly responsive when calls are made. Flexibility for REST API’s comes in the data formats they can support. They include XML, Atom, and most popularly JSON (JavaScript Object Notation). JSON is arranged in key/value pairs making it easily read by humans and machines alike. Unlike XML, JSON does not need a parser and doesn’t use tags making it many times faster than XML and subsequently SOAP messages.



Figure 3 – Comparison of XML to JSON formatting

4 Conclusion

WS-\* standards are a crucial part to the current Internet landscape and will continue to be in the future. They paved the way for newer frameworks such as REST but are still in use today for certain situations and it may be many years before they become obsolete. Allowing interoperability and access to web service functions has contributed greatly to how those services have evolved and where they are going. WS-\* standards are especially important to E-Business as they allow for high quality interaction between clients (consumers) and servers (sellers). We saw above how business needs directly correlated to popularity and adoption of SOAP, WSDL, and UDDI. In the case of UDDI, not being conducive to business needs forced its early descent into obsolescence while the usefulness of SOAP and WSDL ensured their continued development and inspired continued extensions to increase usefulness. REST’s success lies in its performance advantages over SOAP as well the flexibility in data formats it accepts – once again promoting seamless user experiences that promote e-commerce. As evidenced throughout this paper, WS-\* standards and web service interoperability is very much here to stay. Expect to see a continued adoption of RESTful services with increased focus on securing sessions and transactions and further curtailment of SOAP over the next decade. The push towards cloud computing and SaaS, PaaS, and IaaS architectures will cement HTTP/HTTPS interactions with services as the primary method for obtaining and updating data.

**References**

Box, Don, et al. “Web Services Addressing (WS-Addressing).” *Web Services Addressing (WS-Addressing)*, 10 August 2004, [www.w3.org/Submission/ws-addressing/](http://www.w3.org/Submission/ws-addressing/).

Bustamente, Michele Leroux. “Making Sense of all these Crazy Web Service Standards” *InfoQ.com,* 16 May 2007.

<https://www.infoq.com/articles/ws-standards-wcf-bustamante>

Chase, Nicholas “SOAP.” *Understanding Web Services Specifications, Part 1*,12 May 2006, [www.ibm.com/developerworks/webservices/tutorials/ws-understand-web-services1/ws-understand-web-services1.html?ca=drs-](http://www.ibm.com/developerworks/webservices/tutorials/ws-understand-web-services1/ws-understand-web-services1.html?ca=drs-).

“Extensible Markup Language (XML).” *W3C*, www.w3.org/XML/.

May, Howard. “WSDL vs REST Pros and Cons.” *Stack Overflow*, stackoverflow.com/questions/840653/wsdl-vs-rest-pros-and-cons.

“The Pros and Cons of Web Services.” *Dovel Technologies*, doveltech.com/innovation/the-pros-and-cons-of-web-services/.

Tutorialspoint.com. “What Are Web Services.” *Www.tutorialspoint.com*, [www.tutorialspoint.com/webservices/what\_are\_web\_services.htm](http://www.tutorialspoint.com/webservices/what_are_web_services.htm).

“Web Services Standards.” *OpenEdge 11.7 Documentation*, documentation.progress.com/output/ua/OpenEdge\_latest/index.html#page/bpm-web/web-services-standards.html.

“Web Services Security (WS Security) - Definition from Techopedia.” *Techopedia.com*, [www.techopedia.com/definition/24385/web-services-security-ws-security](http://www.techopedia.com/definition/24385/web-services-security-ws-security).

Wyse, Josh. “Why JSON Is Better Than XML” *API Industry Trends,* 27 August 2014.

<https://blog.cloud-elements.com/json-better-xml>