# MEMO

**Subject: C/C between SOAP APIs and REST APIs**

## Introduction

The document serves as an overview for you, a developer, to compare and contrast two of the most widely-used web service application programming interfaces: Simple Object Access Protocol (SOAP) and Representational State Transfer (REST). See Table 1 for a comparison of the two in terms of definition, operation, performance, popularity, complexity, and security.

## Comparison

**Table 1:** Comparison of SOAP and REST interfaces.

| **Criteria** | **Simple Object Access Protocol**[1, 2] | **Representational State Transfer**[1, 2] |
|---|---|---|
| **Definition** | A protocol designed to exchange information across web services in computer networks using HTTP and XML. | An architectural style for distributed hypermedia systems, designed to exchange information across web services in computer networks. |
| **Operation** | SOAP is platform and language independent, and has stateful operations. | Accesses data via HTTP request and has stateful operations, typically HTTP.<br>Basic HTTP requests: GET, PUT, POST, DELETE, UPDATE. |
| Payload | The message is contained in an envelope, with the payload in the body. SOAP relies on XML, which makes the payload heavy. | Request payload can can contain data in JSON, XML, or Text; typically JSON, which lightens the payload. |
| Transport | Generic transport; can use almost any transport to send a request. | Specific transport; uses HTTP/HTTPS. |
| Data Formats | Allows only XML data formats. XML has verbiage and generally requires time and tools to parse. | Allows a variety of data formats, including JSON, YAML, and Text/HTML. Can also use XML. |
| Interface | Has different interfaces through which it operates. | Has a uniform interface to access resources. |
| Built-In Error Handling | Yes; built-in error handling to help fix any problems with requests. | No; does not have built-in error handling help fix any problems with requests. |
| **Performance** | Has performance limitations due to the verbiage of XML. | Fast; no extensive processing required. Efficient; uses smaller message formats instead of XML. |
| **Popularity** | A traditional approach; popular in enterprise environments where complex transactional systems are needed. See *Security.* | A newer approach; used by most major Web Services providers such as Amazon, Twitter, Yahoo, Flickr. |
| **Criteria** | **Simple Object Access Protocol**[1, 2] | **Representational State Transfer**[1, 2] |

| | | |
|---|---|---|
| **Complexity** | Designed to handle distributed computing environments using message patterns between intermediaries. Supports expansion (WSDL). Requires significant tooling and middleware. | Is not designed to handle distributed computing environments. Is simple in design. Does not support expansion. Requires minimal tooling and middleware. |
| **Security** | Standard HTTP protocol that allows operation across firewalls and proxies without modifications to the protocol. Has ACID-compliant transactions. | No established standards defined for security, except for HTTP. |
| Transport-Level Security | Supports SSL in addition to WS-Security, which offers protection from creation to consumption of message in addition to transport-level security. | Supports SSL, as all security measures applied to HTTP are inherited; therefore, offers only transport level security. |
| **Sources:** [1] Stringfellow, Angela. "SOAP vs. REST: Differences in Performance, APIs, and More." *DZone*, https://dzone.com/articles/differences-in-performance-apis-amp-more. [2] Richards, Robert. *Pro PHP XML and Web Services.* Apress, 2006. | | |

## Conclusion

It must be noted that Simple Object Access Protocol is a *protocol*, while Representational State Transfer is an *architectural style* and *design*; a comparison of two is relative. It is most important that you thoroughly test your APIs for functionality, performance, and security.

- A SOAP interface is best if your Web service is a stateful operation independent of platform and language;REST if your services require stateless operations via HTTP.
- A SOAP payload is heavier than the lightweight payload of REST+JSON.
- SOAP has the functionality to use generic transport to send a request, while REST uses only HTTP/HTTPS.
- If you need a variety of data formats, REST is the better option because unlike SOAP, it supports more than just XML.
- SOAP has different interfaces through which it operates, while REST has a uniform interface to access resources.
- While SOAP is not idempotent, it has built-in error handling to help fix errors. REST, on the other hand, is idempotent but does not have a built-in system for error handling.
- SOAP interfaces have performance limitations; REST is faster and more efficient due to its simplicity.
- REST is more commonly used than SOAP, if you are looking for a more popular approach.
- REST is simpler to use and less complex than SOAP. However, if your Web service supports complex operations that require heavy maintenance, SOAP protocols can be less complex than REST in terms of design; a SOAP design requires less coding for application layers (security, transactions, etc.).
- In terms of security, SOAP has a standard HTTP protocol that, while the DELETE and PUT methods in REST often get disabled by firewalls. If you are required to have ACID-compliant transactions or greater transactional reliability, SOAP is definitely the better choice as REST only provides transport-level security.