Soap vs Rest

Contents

Introduction

SOAP and REST are different methods of exchanging information between applications. Both enable web services to transfer data over a network. Both can use HTTP as their transport protocol, and XML as the message format. However, there are important distinctions between SOAP and REST, and while it is impractical to compare them directly, each will work best in different settings.

SOAP

SOAP, the older of the two, is a fully standardized *messaging protocol* with the following characteristics:

- **Exclusive use of XML** as the messaging format. SOAP defines the exact structure for its XML messages. An often-reported downside of using a standardized XML is its verbosity, which results in slow parsing speed and user-perceived latency.
- Can use almost **any transport protocol**: HTTP, SMTP, JMS.
    - If used with SMPT, SOAP can easily support asynchronous communication for time-consuming operations. Since SMTP is asynchronous by nature, the web service making the call (the caller) can execute other operations while the reponding web service is generating the response. The response will be delivered to the caller immediately after it has been generated. SMTP also has build-in retry and delivery notification systems.
- Although stateless by default, SOAP **can support stateful operations**, where the state is transferred between operations:
    - SOAP is a good choice in situations where the exchange of data between web services is composed of multiple chained operations that need to act as one. By referring to the state of a previous operation, each service involved in exchanging data always knows how to perform without making additional calls. SOAP implements this with the help of Web Services Extension Specification (WS).
- Ability to establish a **formal contract** between the web service provider and its consumers with the WSDL document which defines . Having a formal contract also helps reduce misinterpretation in settings where multiple web services involved in the exchange of data are required to behave in a certain way.

SOAP example use case

Web services utilizing SOAP are most commonly found in enterprise environments. SOAP is the preferred choice for example, in bank transfers. Communication between two banks is highly formalized due to security reasons related to the confidentiality of banking data, and the rules governing interactions between the banks do not change very often. Additionally, each bank transfer is composed of multiple operations, so SOAP's ability to support stateful operations comes in handy. For banks, SOAP's reliability, rigidity and security outweigh gains in bandwidth and optimized use of resources that come with REST.

REST

REST, unlike SOAP, is not a messaging protocol but an *architectural style* which prefers convention over formal specification. REST was designed for Internet-scale usage with emphasis on:

- **statelessness** - every packet of information transferred between two web services can be understood in isolation, without any context information. This property increases performance by removing server load caused by retention of information.
- **cacheability** – responses are cacheable, which improves the performance and scalability of a web service by eliminating unnecessary calls to the backend.
- **flexibility** – REST can accept and serve data in many different formats, including JSON, XML and HTML. JSON format is lightweight, parses fast, and in settings where bandwidth is limited works better than a verbose XML.

REST example use case

REST is preferred over SOAP where the efficient use of bandwidth and browser support are of paramount importance. Public APIs and Single Page Applications (SPAs) are two examples where REST is the dominant approach.

Summary

When deciding whether to use SOAP or REST, it might be helpful to ask yourself these questions:

1. Is your API going to be called from web browsers or web apps? If yes, go for REST.
2. Is network connection a constraint? If yes, REST would be a better choice thanks to JSON as the messaging format, statelessness and cacheability.
3. Is asynchronous processing and statefulness a requirement? These demands will be better met by SOAP.
4. How fast would you like your API to be ready? REST is known for its simplicity, ease of development and little learning overhead. These are some of the reasons why many developers prefer REST over SOAP.

Sources

1. https://en.wikipedia.org/wiki/SOAP
2. https://smartbear.com/blog/soap-vs-rest-whats-the-difference/#:~:text=SOAP%20(Simple%20Object%20Access%20Protocol,around%20for%20a%20long%20time.&text=REST%20(Representational%20State%20Transfer)%20is,method%20of%20accessing%20web%20services
3. https://www.soapui.org/learn/api/soap-vs-rest-api/
4. https://www.cleo.com/blog/soap-vs-rest-which-web-service-protocol-is-better#:~:text=REST%20is%20a%20better%20choice,for%20security%2C%20addressing%2C%20etc.
5. https://www.infoq.com/articles/rest-soap-when-to-use-each/#:~:text=Totally%20stateless%20operations%3B%20if%20an,operations%2C%20then%20REST%20is%20it
6. https://nordicapis.com/common-cases-when-using-soap-makes-sense/
7. https://auth0.com/learn/rest-vs-soap/
8. https://www.quora.com/Why-are-bank-transactions-better-suited-to-use-SOAP-Simple-Object-Access-Protocol-rather-than-the-more-popular-REST-Representational-state-transfer

---

Statement: This is my original work. No one has participated in writing this document.