Introduction to SOAP and REST

Overview

When building web services, it is crucial to select the correct architecture and protocol for data exchange. This article compares SOAP and REST, two web service protocols, to help you understand their advantages and disadvantages.

Simple Object Access Protocol (SOAP)

Simple Object Access Protocol (SOAP) is an XML-based protocol that defines rules for structuring messages and exchanging data between web services. It facilitates communication between different systems, platforms, and programming languages.

It uses WSDL (Web Services Description Language) to define:

- bindings with network protocols
- resources, operations, and procedure calls
- message structure, encoding, and encapsulation

Advantages of SOAP

- Enhanced data integrity: SOAP enforces a strict contract, improving data integrity and service reliability.
- Protocol flexibility: SOAP supports transport protocols such as HTTP, SMTP, and TCP.
- In-built error handling: SOAP provides a streamlined approach to report errors.

Disadvantages of SOAP

- XML rigidity: SOAP exclusively uses XML, which is time-consuming and difficult to work with compared to JSON or HTML.
- Performance cost: SOAP messages are long, leading to higher processing overhead for both parsing and generation.
- Tight coupling: The tight coupling between server and client implementations limits the flexibility of the web.

Representational State Transfer(REST)

REST is an architectural style for designing stateless web services, widely adopted for its simplicity and flexibility. RESTful services are easier to build and maintain, with an emphasis on simplicity and scalability.

Advantages of REST

- Faster implementations: REST's constraints allow for rapid iteration and implementation.
- Cached responses: Responses are partially cached, leading to improved performance and reduced server load.
- Stateless: Service is stateless as each request from the client to the server contains all necessary information.

Disadvantages of REST

- Security concerns: By default, REST uses HTTP, which is an insecure protocol. For secure communication, implement REST over HTTPS to protect data in transit.
- Limited operations: REST supports only standard HTTP methods, which can be restrictive for complex use cases.
- Scalability challenges: Without a proper architecture, point-to-point communication is difficult to scale.

SOAP vs REST

Both SOAP and REST have their strengths and weaknesses. SOAP is suited for highly secure, reliable services, while REST is ideal for flexible, scalable web services.

The following table lists the differences between the two protocols:

SOAP API	REST API
Security	
SOAP API message verifies security information in the header to confirm it is not altered. WS-Security governs the confidentiality and authentication processes for passwords, XML encryption, and security tokens.	REST APIs lack the built-in security features that SOAP APIs offer. Instead, common authentication methods for REST APIs include HTTP Basic Authentication, JSON Web Tokens (JWT), OAuth, and API keys.
Services	
A user can create, retrieve, update, or delete records such as passwords, accounts, leads, and custom objects.	A user can use REST APIs to modify existing items or add new ones on a server using various HTTP methods.
Protocol	

SOAP API	REST API
SOAP APIs work with any transport protocol, making them versatile.	REST APIs typically work with the HTTPS transport protocol.
Uses	
SOAP APIs are used in both web and non-web applications	REST APIs are primarily used in web applications.

SOAP Use Case

Use SOAP API when you require:

- high-security services, such as for financial transactions or healthcare
- guaranteed delivery and reliable messaging
- services that require stringent data integrity or transaction standards

REST Use Case

Use REST API when you require:

- a lightweight and flexible solution, ideal for mobile applications or social media platforms
- serving JSON data for web applications and mobile clients
- prioritizing performance and scalability for high traffic environments
- a public API that supports third-party integration