SOAP VS REST

What is the difference between SOAP and REST?

SOAP and REST are two internet data exchange mechanisms. For example, imagine that your internal accounts system shares data with your customer's accounting system to automate invoicing tasks. The two applications share data by using an API that defines communication rules. SOAP and REST are two different approaches to API design. The SOAP approach is highly structured and uses XML data format. REST is more flexible and allows applications to exchange data in multiple formats.

What are the similarities between SOAP and REST?

To build applications, you can use many different programming languages, architectures, and platforms. It’s challenging to share data between such varied technologies because they have different data formats. Both SOAP and REST emerged in an attempt to solve this problem.

You can use SOAP and REST to build APIs or communication points between diverse applications. The terms *web service*and *API* are used interchangeably. However, APIs are the broader category. Web services are a special type of API.

Here are other similarities between SOAP and REST:

* They both describe rules and standards on how applications make, process, and respond to data requests from other applications
* They both use HTTP, the standardized internet protocol, to exchange information
* They both support SSL/TLS for secure, encrypted communication

You can use either SOAP or REST to build secure, scalable, and fault-tolerant distributed systems.

When to use SOAP vs REST?

Before choosing between SOAP and REST, consider your scenarios and your API users' requirements. The following criteria are worth considering.

Overall application design

Modern applications like mobile apps and hybrid applications work better with REST APIs. REST gives you the scalability and flexibility to design applications using modern architecture patterns like microservices and containers. However, if you need to integrate or extend legacy systems that already have SOAP APIs, you may be better off continuing with SOAP.

Security

Public APIs have lower security requirements and demand greater flexibility so anyone can interact with them. So, REST is a better choice when you build public APIs. Conversely, some private APIs for internal enterprise requirements (like data reporting for compliance) may benefit from the tighter security measures in WS-Security of SOAP.

ACID compliance

Do your API users require stringent consistency and data integrity across a chain of transactions? For instance, finance transactions require an entire batch of data updates to fail if even one update fails.

SOAP has built-in compliance for atomicity, consistency, isolation, and durability (ACID). And SOAP may be better suited for high data integrity requirements. In this case, REST APIs may require additional software modules to enforce the state at the server or database level.

How do SOAP APIs and REST APIs work?

SOAP is an older technology that requires a strict communication contract between systems. New web service standards have been added over time to accommodate technology changes, but they create additional overheads. REST was developed after SOAP and inherently solves many of its shortcomings. REST web services are also called *RESTful web services*.

SOAP APIs

SOAP is a protocol that defines rigid communication rules. It has several associated standards that control every aspect of the data exchange. For example, here are some standards SOAP uses:

* Web Services Security (WS-Security) specifies security measures like using unique identifiers called *tokens*
* Web Services Addressing (WS-Addressing) requires including routing information as metadata
* WS-ReliableMessaging standardizes error handling in SOAP messaging
* Web Services Description Language (WSDL) describes the scope and function of SOAP web services

When you send a request to a SOAP API, you must wrap your HTTP request in a *SOAP envelope*. This is a data structure that modifies the underlying HTTP content with SOAP request requirements. Due to the envelope, you can also send requests to SOAP web services with other transport protocols, like TCP or Internet Control Message Protocol (ICMP). However, SOAP APIs and SOAP web services always return XML documents in their responses.

REST APIs

REST is a software architectural style that imposes six conditions on how an API should work. These are the six principles REST APIs follow:

1. *Client-server architecture*. The sender and receiver are independent of each other regarding technology, platforming, programming language, and so on.
2. *Layered*. The server can have several intermediaries that work together to complete client requests, but they are invisible to the client.
3. *Uniform interface*. The API returns data in a standard format that is complete and fully useable.
4. *Stateless*. The API completes every new request independently of previous requests.
5. *Cacheable*. All API responses are cacheable.
6. *Code on demand*. The API response can include a code snippet if required.

You send REST requests using HTTP verbs like *GET* and *POST*. Rest API responses are typically in JSON but can also be of a different data format.

Key differences: SOAP vs REST

SOAP is a protocol, while REST is an architectural style. This creates significant differences in how SOAP APIs and REST APIs behave.

Design

The SOAP API exposes functions or operations, while REST APIs are data-driven. For example, consider an application with employee data that other applications can manipulate. The application's SOAP API could expose a function called *CreateEmployee*. To create an employee, you would specify the function name in your SOAP message when sending a request.

However, the application's REST API could expose a URL called */employees,* and a POST request to that URL would create a new employee record.

Flexibility

SOAP APIs are rigid and only allow XML messaging between applications. The application server also has to maintain the state of each client. This means it has to remember all previous requests when processing a new request.

REST is more flexible and allows applications to transfer data as plain text, HTML, XML, and JSON. REST is also stateless, so the REST API treats every new request independently of previous requests.

Performance

SOAP messages are larger and more complex, which makes them slower to transmit and process. This can increase page load times.

REST is faster and more efficient than SOAP due to the smaller message sizes of REST. REST responses are also cacheable, so the server can store frequently accessed data in a cache for even shorter page load times.

Scalability

The SOAP protocol requires applications to store the state between requests, which increases bandwidth and memory requirements. As a result, it makes applications expensive and challenging to scale.

Unlike SOAP, REST permits stateless and layered architecture, which makes it more scalable. For example, the application server can pass the request to other servers or allow an intermediary (like a content delivery network) to handle it.

Security

SOAP requires an additional layer of WS-Security to work with HTTPS. WS-Security uses additional header content to ensure only the designated process in the specified server reads the SOAP message content. This adds communication overheads and negatively impacts performance.

REST supports HTTPS without additional overheads.

Reliability

SOAP has error handling logic built into it, and it provides more reliability. On the other hand, REST requires you to try again in case of communication failures, and it’s less reliable.