Choosing between REST and SOAP: A guide for engineers

Table of Contents

When designing a web service, you must carefully evaluate the choice between SOAP and REST. This guide explains their strengths, limitations, and use cases to help you make an informed decision.

SOAP overview

SOAP (Simple Object Access Protocol) is a protocol for exchanging XML-based messages over a network. It uses Web Services Description Language (WSDL) to define:

- Resources, operations, and procedure calls
- Message structure, encoding, and encapsulation
- Bindings with network transport protocols

SOAP APIs are often auto-generated using tools that create server-side and client-side code templates, but the resulting implementations can be challenging to develop and maintain.

When to choose SOAP

SOAP is suitable for:

- **Enterprise-level security:** Supports WS-Security for encryption, authentication, and message integrity.
- **Complex transactions:** Ensures reliability in distributed environments through features like ACID-compliant transactions.

- **Strict contracts:** Enforces formal service definitions using WSDL, ensuring consistency and reliability.
- **Stateful interactions:** Useful for cases requiring session management or multi-step workflows.
- **Legacy systems:** Facilitates integration with older enterprise systems.

SOAP limitations

- **Tight coupling:** Changes to the server often require updates to the client, reducing flexibility.
- **XML-only payloads:** Mandatory use of XML increases message size and complexity.
- **High overhead:** SOAP messages are verbose, requiring more bandwidth and processing power.
- **Slower development:** Complex implementation results in longer development cycles.

REST overview

REST (Representational State Transfer) is an architectural style that leverages standard HTTP methods for stateless web service interactions. REST APIs:

- Use URLs to represent resources.
- Support multiple data formats, including JSON, XML, and HTML.
- Implement HTTP methods such as `GET`, `POST`, `PUT`, `DELETE`, `PATCH`, and more.
- Can mark responses as cacheable, reducing client-server interactions.

Tools and frameworks can generate REST APIs and documentation from application code, enabling rapid development.

When to choose REST

REST is ideal for:

- **Public APIs:** Designed for widespread adoption and third-party integrations.
- **Rapid development:** Simple and flexible, with fewer constraints than SOAP.
- **Data format flexibility:** Supports lightweight formats like JSON, ideal for modern web and mobile applications.
- **Scalability and caching:** Statelessness and response caching improve performance and scalability.
- **Broad adoption:** REST is widely supported by tools, frameworks, and a large developer community.

REST limitations

- **Security challenges:** Requires additional measures (e.g., HTTPS, token-based authentication) to ensure security.
- **Limited operations:** Restricts interactions to standard HTTP methods, which may not cover all use cases.
- **Scalability complexities:** Point-to-point communication can complicate scaling with multiple clients.
- **Versioning difficulties:** Major schema changes can break clients if not managed properly.

Decision guide

- Use **SOAP** for secure, reliable, and stateful operations, especially in enterprise environments or when working with legacy systems.
- Use **REST** for simple, scalable, and fast APIs that support modern web or mobile applications and prioritize flexibility.

> **TIP** Choose REST for most modern web services unless your requirements specifically align with SOAP's strengths, such as strict contracts or enterprise-grade security.

Key considerations

- Assess your team's expertise with SOAP or REST.
- Plan for future maintenance, including versioning strategies for REST APIs.
- Use tools for API generation and documentation to speed up development and ensure consistency.