Choosing Between REST and SOAP

This document summarizes the differences between representational state transfer (REST) and SOAP (Simple Object Access Protocol). While both REST and SOAP define API approaches for transferring data between web services, SOAP is a specific protocol, whereas REST is an architectural style defined by a set of principles and constraints. At a glance, REST is more lightweight and flexible, whereas SOAP is more resource intensive, rigid, and secure.

Table 1: Comparative Attributes of REST and SOAP

Point of Comparison	REST	SOAP
Classification	data-driven architectural style	function-driven protocol
Organization Scheme	compliance with REST architectural constraints	envelope message structure
Performance / Efficiency	lightweight, requires fewer resources and less bandwidth	heavyweight, requires more resources and bandwidth
Supported Messaging Formats	JSON, HTML, XML, and more	XML
Security	SSL, HTTPS	WS-security with SSL support, built-in ACID compliance
Supported Transfer Protocols	HTTP	HTTP, SMTP, FTP, and more
Statefulness	stateless	stateless (default) or stateful
Caching	HTTP caching	no HTTP caching
Ease of Adoption	easy	more difficult

When to use REST

Consider using REST in use cases like simple, resource-driven applications or public APIs where:

- You have restricted bandwidth and resources.
- You have no need to maintain state.
- You want easy caching and coding.

When to use SOAP

Consider using SOAP in cases where:

- You need to process stateful operations or have complex, interdependent API calls.
- Security is a priority. SOAP is well suited for Enterprise-level web services with high security requirements and complex transactions, such as financial services.
- You want built-in error handling.

Additional Background and Context

REST or RESTful APIs are a popular choice for developers building public APIs. REST architectural constraints include a uniform interface, client-server separation, statelessness, cacheable resources, compatibility with layered systems, and support for code on demand (this last constraint is optional). REST allows you to deliver data in multiple convenient formats including JavaScript Object Notation (JSON). This support for lightweight, human-

readable formats makes REST APIs attractive due to their speed and efficiency. The example below shows a simple JSON message, note the brevity of this message compared to the SOAP example further down the page:

```
{
  meta: {
  },
  data: [{
    id: 0,
    title: 'The Worm Ouroboros',
    author: 'E. R. Eddison'
  }]
}
```

SOAP is older than REST. Originally developed by Microsoft, SOAP is a highly extensible protocol for sending XML messages comprised of the following elements:

- **<Envelope>** – The root element in every SOAP message. An <Envelope> element contains an optional header element and a mandatory body element.
  - **<Header>** – (optional) <Header> elements contain application-specific information to provide context for the SOAP message.
  - **<Body>** – The <Body> element is a mandatory part of every SOAP message that contains the information to be transferred to the final message recipient. A <Body> element can also contain a <Fault> child element for error reporting.

The following example from Wikipedia shows a SOAP message encapsulated in HTTP:

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 299
SOAPAction: "http://www.w3.org/2003/05/soap-envelope"

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-
envelope" xmlns:m="http://www.example.org">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <m:GetStockPrice>
      <m:StockName>T</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

The relative complexity and overhead associated with the mandatory XML format of SOAP messages is one of the main reasons many developers prefer RESTful APIs over SOAP based APIs.

Summary

Ultimately, the choice between REST or SOAP depends on your use case and, to a certain extent, on the work-bandwidth available within your team. Both SOAP and REST can accomplish the same goals with enough work, but the amount of work involved, and the optimality of the result, will depend on how well the use case aligns with the attributes described above.