Web Service

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

How Does a Web Service Work?

A web service enables communication among various applications by using open standards such as HTML, XML, WSDL, and SOAP. A web service takes the help of −

1. XML to tag the data
2. SOAP to transfer a message
3. WSDL to describe the availability of service.

Example

Consider a simple account-management and order processing system. The accounting personnel use a client application built with Javascript to create new accounts and enter new customer orders.
The processing logic for this system is written in Java and resides on a Solaris machine, which also interacts with a database to store information. The steps to perform this operation are as follows −

1. The client program bundles the account registration information into a SOAP message.
2. This SOAP message is sent to the web service as the body of an HTTP POST request.
3. The web service unpacks the SOAP request and converts it into a command that the application can understand.
4. The application processes the information as required and responds with a new unique account number for that customer.
5. Next, the web service packages the response into another SOAP message, which it sends back to the client program in response to its HTTP request.
6. The client program unpacks the SOAP message to obtain the results of the account registration process.

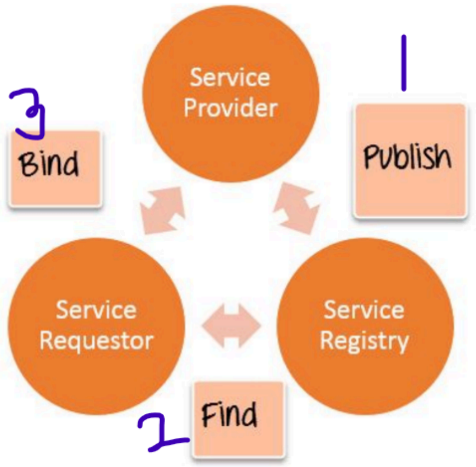
Why Web Services?

https://www.tutorialspoint.com/webservices/why_web_services.htm

Web Services - Characteristics

https://www.tutorialspoint.com/webservices/web_services_characteristics.htm
https://www.guru99.com/web-service-architecture.html

Web Service Architecture (Link)

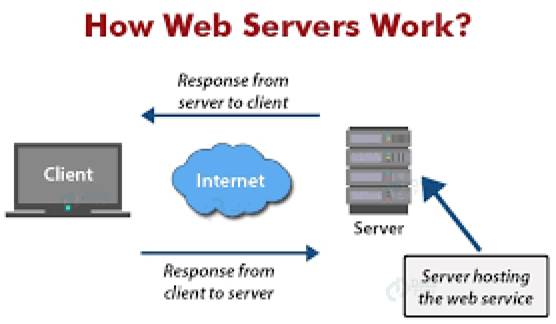


Service Provider
This is the provider of the web service. The service provider implements the service and makes it available on the Internet. Creates web service and makes it available to clients.

Service Requestor (client)
This is any consumer of the web service. The requestor utilizes an existing web service by opening a network connection and sending an XML request.

Service Registry/Broker
This is a logically centralized directory of services. The registry provides a central place where developers can publish new services or find existing ones. It therefore serves as a centralized clearing house for companies and their services.

Service provider first e create kora web service ta publish korbe Service Registry er kase. Service Registry sheta UDDI te add kore dibe. Erpor client er je service dorkar, client sheta Service Registry theke UDDI diye find korbe je tar required service ta kono web service provide kortese kina. Jodi kono web service provide kore, tokhon client oi service provider ke message diye tar sathe bind hoye jabe.

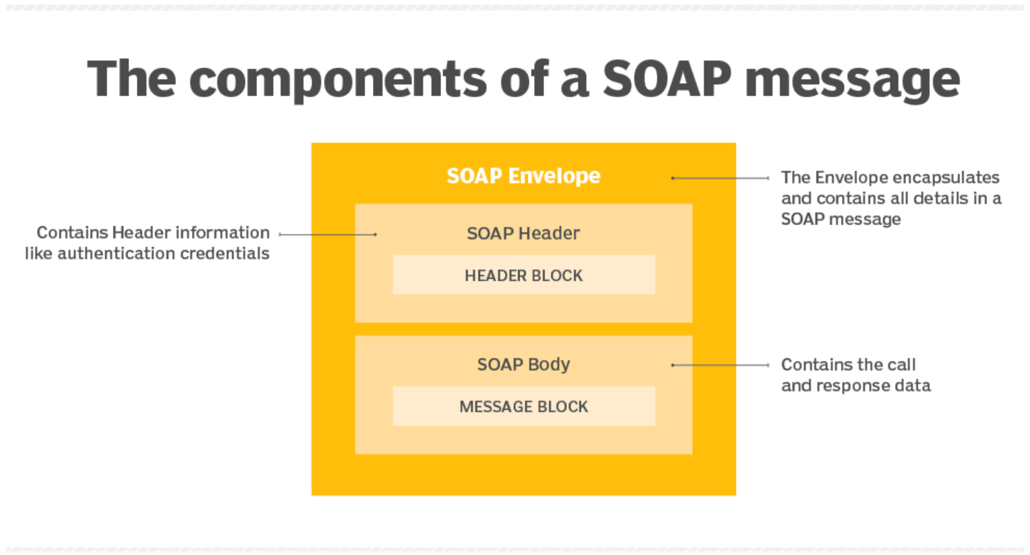

How Web Servers Work?
Response from server to client
Client
Internet
Server
Response from client to server
Server hosting the web service

Types of Web Service

SOAP
REST

SOAP (Simple Object Access Protocol)

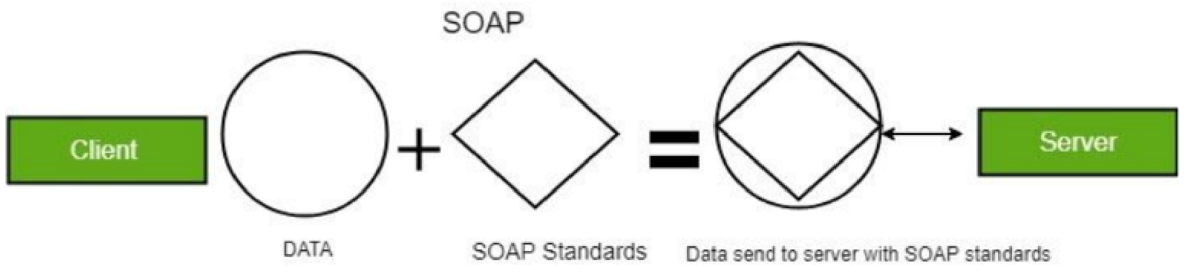
https://www.techtarget.com/searchapparchitecture/definition/SOAP-Simple-Object-Access-Protocol
It is an XML-based messaging protocol for exchanging information among computers. SOAP can be carried over a variety of standard protocols, including the web-related Hypertext Transfer Protocol (HTTP).


The components of a SOAP message
SOAP Envelope
The Envelope encapsulates and contains all details in a SOAP message
Contains Header information like authentication credentials
SOAP Header
HEADER BLOCK
SOAP Body
MESSAGE BLOCK
Contains the call and response data

https://www.tutorialspoint.com/soap/soap_message_structure.htm
****Fault** − An optional Fault element that provides information about errors that occur while processing the message.
** msg block contains the xml data. XML is a msg format. Example:
https://www.w3schools.com/js/js_json_xml.asp

How does SOAP work?

SOAP requests are easy to generate and process responses. First, a request for a service is generated by a client using an XML document. Next, a SOAP client sends the XML document to a SOAP server. When the server receives the SOAP message, it sends the message as a service invocation to the requested server-side application. A response containing the requested parameters, return values and data for the client is returned first to the SOAP request handler and then to the requesting client. Both SOAP requests and responses are transported using Hypertext Transfer Protocol Secure (HTTPS) or a similar protocol like HTTP.



SOAP Message Example

- Transport independent messaging protocol
- Doesn’t restrict content
- But document has to follow a specific pattern
- Ei SOAP message er ekta root element thaka lagbe, named Envelope
- Content-type: **text/xml**

WSDL (Web services description language)

A web service cannot be used if it cannot be found. The client invoking the web service should know where the web service actually resides.

Secondly, **the client application needs to know what the web service actually does**, so that it can invoke the right web service. This is done with the help of the WSDL, known as the Web services description language. The WSDL file is again an XML-based file which basically **tells the client application what the web service does**. By using the WSDL document, the client application would be able to understand where the web service is located and how it can be utilized.

Universal Description Discovery and Integration (UDDI)

Now we discussed in the previous topic about WSDL and how it contains information on what the Web service actually does. But how can a client application locate a WSDL file to understand the various operations offered by a web service? So UDDI is the answer to this and provides a repository on which WSDL files can be hosted. So the client application will have complete access to the UDDI, which acts as a database containing all the WSDL files.

RESTful (Representational State Transfer) Web Services

RESTful Web Services are basically REST Architecture based Web Services. In REST Architecture **everything is a resource**. It's more of an approach that's centered around resources and things you can do to resources. The HTTP verbs GET, POST, PUT and DELETE are typical actions that you can apply against any resource.

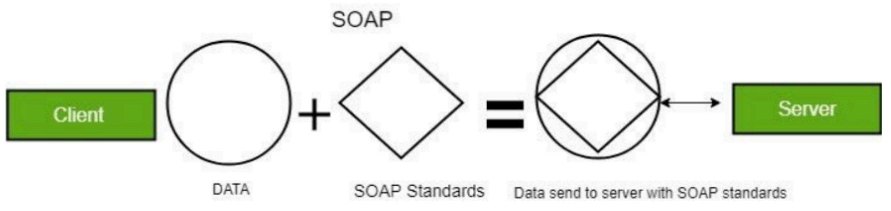
RESTful web services use the HTTP protocol to perform requests from a web service. They use the HTTP verbs: GET, POST, PUT and DELETE etc. Sometimes the requests will contain data in the body that could be HTML, JSON, binary data or other.

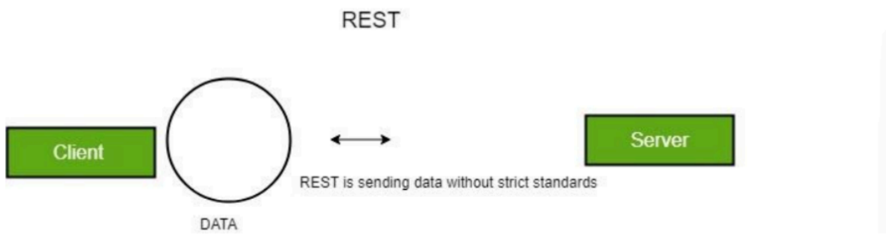
A purely RESTful web service only **requires the URL and the HTTP** verb to describe the requested action. The body data is usually a payload to be involved in the requested action. It should not dictate the requested action.

RESTful web services use URL to access the components on the hardware devices. **Alada kono XML document create kore pathano lagtese na.**

bandwidth: REST does not need much bandwidth when requests are sent to the server. REST messages mostly just consist of JSON messages.

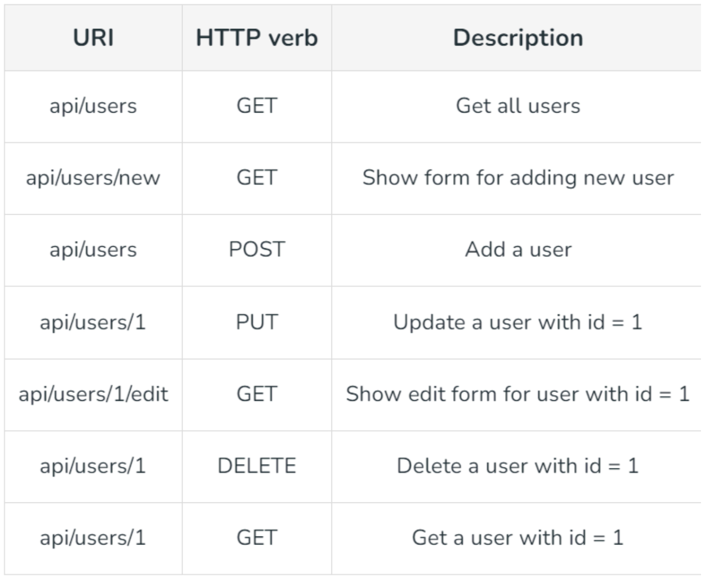
Data format: SOAP supported only XML. REST permits different data formats such as Plain text, HTML,XML, JSON, etc. But the most preferred format for transferring data is JSON.





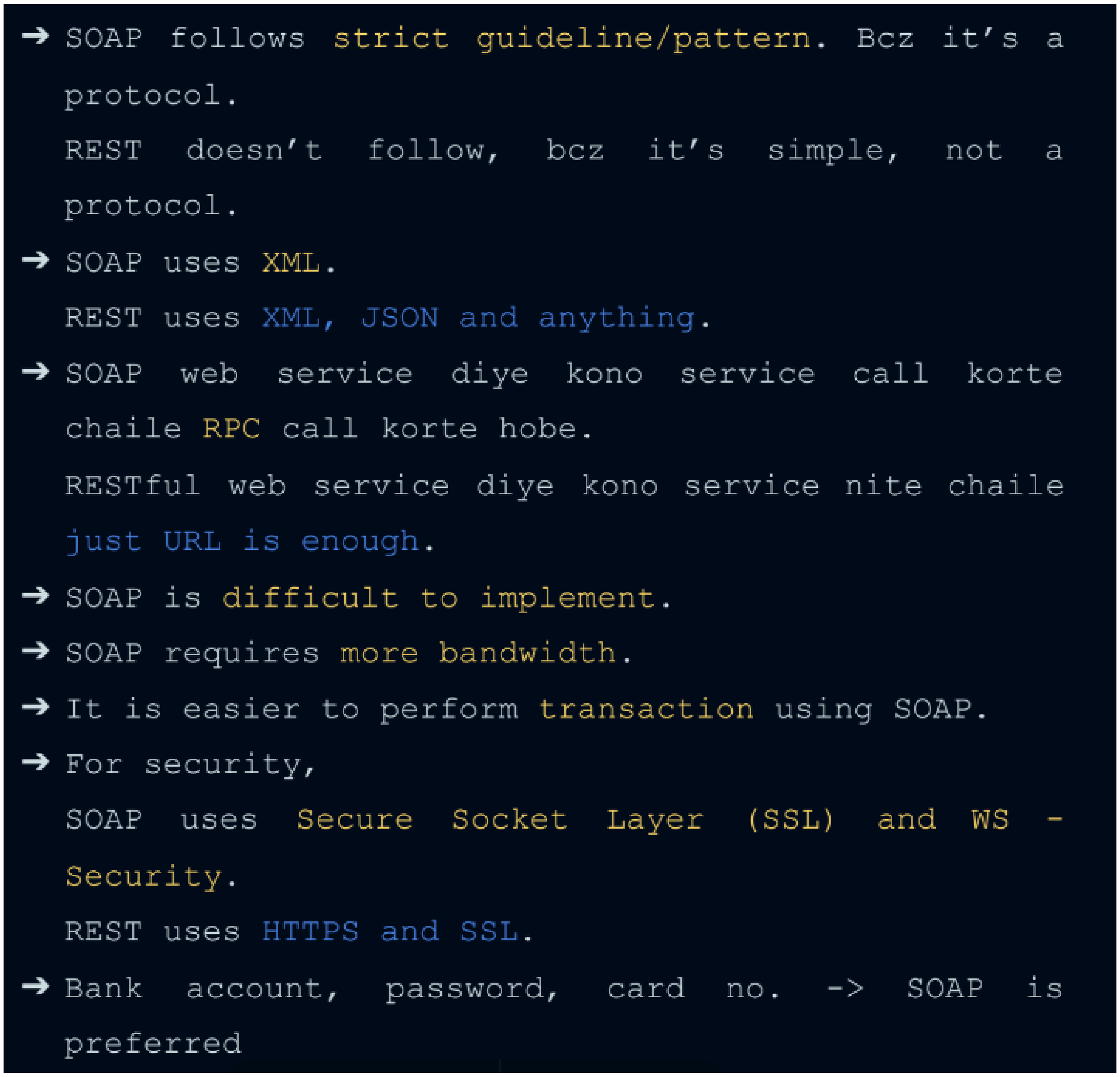
Rules of REST API (Link)

1. Noun endpoint / Resource. It means that a URI of a REST API should always end with a noun. Example: /api/users is a good example, but /api?type=users is a bad example of creating a REST API.
2. Request using HTTP Action Verb (GET, POST etc)
3. Plural of resources (/users, /users/1, not user/1)
4. HTTP Response Code (jeno success/failure client bujhte pare)
5. RESTful web service ta khubi organized way te howa lagbe (e.g. api/users/1 DELETE dekhe bujhtesi user with id 1 ke delete kora lagbe)



URI	HTTP verb	Description
api/users	GET	Get all users
api/users/new	GET	Show form for adding new user
api/users	POST	Add a user
api/users/1	PUT	Update a user with id = 1
api/users/1/edit	GET	Show edit form for user with id = 1
api/users/1	DELETE	Delete a user with id = 1
api/users/1	GET	Get a user with id = 1

SOAP VS REST



➔ SOAP follows strict guideline/pattern. Bcz it's a protocol.
REST doesn't follow, bcz it's simple, not a protocol.

➔ SOAP uses XML.
REST uses XML, JSON and anything.

➔ SOAP web service diye kono service call korte chaile RPC call korte hobe.
RESTful web service diye kono service nite chaile just URL is enough.

➔ SOAP is difficult to implement.

➔ SOAP requires more bandwidth.

➔ It is easier to perform transaction using SOAP.

➔ For security,
SOAP uses Secure Socket Layer (SSL) and WS - Security.
REST uses HTTPS and SSL.

➔ Bank account, password, card no. -> SOAP is preferred

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.

Example: Oracle Enterprise - SOAP for communication among internal modules.
Healthcare - patient-hospital → Rest ; hospital-insurance company → SOAP

Web service vs API:

(Link)

1. Need of Network
   API - Not always
   Web Services - Always
2. Lightweight Architecture
   API - Yes
   Web Services - No
3. Style of Communication
   API - Any
   Web Services - SOAP, REST, XML - RPC
4. Protocol Restriction
   API - WebAPI -> HTTP
   Web Services - HTTP
5. Hosting
   API - Application / Internet
   Web Services - Internet
6. Open Source
   API - Yes (Mostly)
   Web Services - Mostly No

Asynchronous Function:

https://www.mendix.com/blog/asynchronous-vs-synchronous-programming/

https://www.reddit.com/r/expressjs/comments/15yx7yx/when_to_use_async/