API QUESTIONS & ANSWERS

1. What is an API?

An API (Application Programming Interface) is a software intermediary that enables two applications to communicate with each other. It comprises a number of subroutine definitions, logs, and tools for creating application software.

In an API testing interview, you could be asked to give some API examples, here are the well-known ones: Google Maps API, Amazon Advertising API, Twitter API, YouTube API, etc.

2. What are main differences between API and Web Service?

- All Web services are APIs but not all APIs are Web services.
- Web services might not contain all the specifications and cannot perform all the tasks that APIs would perform.
- A Web service uses only three styles of use: SOAP, REST and XML-RPC for communication whereas API may be exposed to in multiple ways.
- A Web service always needs a network to operate while APIs don't need a network for operation.

3. What are the Limits of API Usage?

Many APIs have a certain limit set up by the provider. Thus, try to estimate your usage and understand how that will impact the overall cost of the offering. Whether this will be a problem depends in large part on how data is leveraged. Getting caught by a quota and effectively cut-off because of budget limitations will render the service (and any system or process depending on it) virtually useless.

4. What are some architectural styles for creating a Web API?

This is one of the fundamental Web API interview questions. Bellows are four common Web API architectural styles:

- HTTP for client-server communication
- XML/JSON as formatting language
- Simple URI as the address for the services
- Stateless for communication

5. Who can use a Web API?

Web API can be consumed by any client which support HTTP verbs such as GET, PUT, DELETE, POST. Since Web API services do not require configuration, they can be easily used by any client. In fact, even portable devices such as mobile devices can easily use Web API, which is undoubtedly the biggest advantage of this technology.

6. What is API Testing?

API testing is a type of software testing that involves testing APIs directly and also as a part of integration testing to check whether the API meets expectations in terms of functionality, reliability, performance, and security of an application. In API Testing our main focus will be on Business logic layer of the software architecture. API testing can be performed on any software system which contains multiple APIs.

- API testing is conducted by QA Team
- API testing is a form of Black box testing
- API testing is conducted after the build is ready for testing

- Source code is not involved in API testing
- In API testing, the scope of testing is wide, so all the issues that are functional are considered for testing

7. What are the advantages of API Testing?

In an API interview, they are likely to ask about the advantages of API testing. So be prepared with the significant ones such as:

- ***Test for Core Functionality:*** API testing provides access to the application without a user interface. The core and code-level of functionalities of the application will be tested and evaluated early before the GUI tests. This will help detect the minor issues which can become bigger during the GUI testing.
- ***Time Effective:*** API testing usually is less time consuming than functional GUI testing. The web elements in GUI testing must be polled, which makes the testing process slower. Particularly, API test automation requires less code so it can provide better and faster test coverage compared to GUI test automation. These will result in the cost saving for the testing project.
- ***Language-Independent:*** In API testing, data is exchanged using XML or JSON. These transfer modes are completely language-independent, allowing users to select any code language when adopting automation testing services for the project.
- ***Easy Integration with GUI:*** API tests enable highly integrable tests, which is particularly useful if you want to perform functional GUI tests after API testing. For instance, simple integration would allow new user accounts to be created within the application before a GUI test started.

8. Some common protocols used in API testing?

Many protocols are now available to be used in API testing, such as JMS, REST, HTTP, UDDI and SOAP.

9. What is the test environment of API?

Setting up the API's test environment is not an easy task, so you should have a ready answer if your API testing interview is coming. The test environment of API is a bit complete and requires the configuration of the database and server, depending on the software requirements. No GUI (Graphical User Interface) is available in this test form.

When the installation process is complete, API is verified for the proper operation. Throughout the process, the API called from the original environment is set up with different parameters to study the test results.

10. What are principles of an API test design?

The five most important principles of an API test design are:

- Setup: Create objects, start services, initialize data, etc.
- Execution: Steps to apply API or the scenario, including logging
- Verification: Oracles to evaluate the result of the execution
- Reporting: Pass, failed or blocked
- Clean up: Pre-test state

11. What are the common API testing types?

While there are certainly specialty tests, and no list can be asked to be comprehensive in this realm, most tests fit broadly into these following nine categories that you should remember before attending in an API testing interview.

1. Validation Testing

2. Functional Testing
3. UI testing
4. Load testing
5. Runtime/ Error Detection
6. Security testing
7. Penetration testing
8. Fuzz testing
9. Interoperability and WS Compliance testing

12. What is the procedure to perform API testing?

1. Choose the suite to add the API test case
2. Choose the test development mode
3. Demand the development of test cases for the required API methods
4. Configure the control parameters of the application and then test conditions
5. Configure method validation
6. Execute the API test
7. Check test reports and filter API test cases
8. Arrange all API test cases

13. What must be checked when performing API testing?

During the API testing process, a request is raised to the API with the known data. This way you can analyze the validation response. While testing an API, you should consider:

- Accuracy of data
- Schema validation
- HTTP status codes
- Data type, validations, order and completeness
- Authorization checks
- Implementation of response timeout
- Error codes in case API returns, and
- Non-functional testing like performance and security testing

14. What is the best approach method to perform API testing?

The following factors should be considered when performing API testing:

- Defining the correct input parameters
- Verifying the calls of the mixture of two or more added value parameters
- Defining the basic functionality and scope of the API program
- Writing appropriate API test cases and making use of testing techniques such as equivalence class, boundary value, etc. to check the operability
- Testing case execution
- Comparing the test result with the expected result
- Verifying the API behavior under conditions such as connection to files and so on.

15. What tools could be used for API testing?

There are myriad of different **API testing tools** available. A few of common tools are Katalon Studio, Postman, SoapUi Pro, Tricentis Tosca, Apigee, etc. While doing Unit and API testing, both targets source code. If an API method uses code based in .NET then another supporting tool must have .NET.

16. What are differences between API Testing and Unit Testing?

API Testing	Unit Testing
• Conducted by QA team	• Conducted by the development
• Mostly black box testing	• White box testing
• Aimed to assess the full functionality of the system for it will be employed by the end-user (external developers who will use your API)	• Used to verify whether each unit in isolation performs as expected or not
• Often run after the build is ready and authors do not have access to the source code	• Each of the code modules must be ensured to pass the unit test before being built by developers.

17. What are differences between API Testing and UI Testing?

- API enables communication between two separate software systems. A software system implementing an API contains functions or subroutines that can be executed by another software system.
- On the other hand, UI (User Interface) testing refers to testing graphical interface such as how users interact with the applications, testing application elements like fonts, images, layouts etc. UI testing basically focuses on look and feel of an application.

18. What are major challenges faced in API testing?

If you can overcome the challenges in API Testing, you can be confident in the API testing interview too. They are:

- Parameter Selection
- Parameter Combination
- Call sequencing
- Output verification and validation
- Providing input values, which is very difficult as GUI is not available in this case.

19. What are the testing methods that come under API testing?

One of the most common Web API testing interview questions is about the testing methods. They are:

- Unit testing and Functional testing
- Load testing to test the performance under load
- Discovery testing to list, create and delete the number of calls documented in API
- Usability and Reliability testing to get consistent results
- Security and Penetration testing to validate all types of authentication
- Automation testing to create and run scripts that require regular API calls
- End to end Integration and Web UI testing
- API documentation testing to determine its efficiency and effectiveness

20. Why is API testing considered as the most suitable form for Automation testing?

API testing is now preferred over GUI testing and is considered as most suitable because:

- It verifies all the functional paths of the system under test very effectively.
- It provides the most stable interface.
- It is easier to maintain and provides fast feedback.

21. What are common API errors that often founded?

Not only API fundamental questions, the interviewer also determine your knowledge and experience by asking about the API errors in a Web API testing interview. So the most common ones are:

- Missing module errors
- Documentation errors
- Parameter validation errors
- And some standard error expectations as if the result is not so predicted then the occurrence of errors can be seen and for the same warnings are specified in the form of a message. There can be one or more warnings within an individual module.

22. What kinds of bugs that API testing would often find?

- Missing or duplicate functionality
- Fails to handle error conditions gracefully
- Stress
- Reliability
- Security
- Unused flags
- Not implemented errors
- Inconsistent error handling
- Performance
- Multi-threading issues
- Improper errors

23. What is API documentation?

The API documentation is a complete, accurate technical writing giving instructions on how to effectively use and integrate with an API. It is a compact reference manual that has all the information needed to work with the API, and helps you answer all the API testing questions with details on functions, classes, return types, arguments, and also examples and tutorials.

24. What are API documentation templates that are commonly used?

There are several available API documentation templates help to make the entire process simple and straightforward, which could be answered in your API testing interview, such as:

- Swagger
- Miredot
- Slate
- FlatDoc
- API blueprint
- RestDoc
- Web service API specification

25. When writing API document, what must be considered?

- Source of the content
- Document plan or sketch
- Delivery layout
- Information needed for every function in the document
- Automatic document creation programs

26. How often are the APIs changed and, more importantly, deprecated?

APIs, especially modern RESTful APIs, are a nice creation that can certainly simplify and accelerate integration efforts, which makes it more likely you will benefit from them. But APIs can and do change for various reasons, sometimes abruptly, and hence REST APIs do not differ from traditional integration methods in this respect. If an API call is obsolete and disappears, your procedure will interrupt and it is important to understand how often the APIs you depend on change or are deprecated.

27. What is REST?

REST (Representational State Transfer) is an architectural style for developing web services which exploit the ubiquity of HTTP protocol and uses HTTP method to define actions. It revolves around resource where every component being a resource that can be accessed through a shared interface using standard HTTP methods. In REST architecture, a REST Server provides access to resources and REST client accesses and makes these resources available. Here, each resource is identified by URIs or global IDs, and REST uses multiple ways to represent a resource, such as text, JSON, and XML. XML and JSON are nowadays the most popular representations of resources.

28. What is a RESTFul Web Services?

Mostly, there are two kinds of Web Services which should be remembered in your next API testing interview:

1. SOAP (Simple Object Access Protocol) – an XML-based method to expose web services.
2. Web services developed in the REST style are referred to as RESTful web services. These web services use HTTP methods to implement the concept of REST architecture. A RESTful web service usually defines a URI, Uniform Resource Identifier a service, provides resource representation like JSON and a set of HTTP methods.

29. What is a “Resource” in REST?

REST architecture treats any content as a resource, which can be either text files, HTML pages, images, videos or dynamic business information. REST Server gives access to resources and modifies them, where each resource is identified by URIs/ global IDs.

30. What is the most popular way to represent a resource in REST?

REST uses different representations to define a resource like plain text, CSV, HTML, JSON, and XML. XML and JSON are the most popular representations of resources.

31. Which protocol is used by RESTful Web services?

RESTful web services use the HTTP protocol as a medium of communication between the client and the server.

32. What are some key characteristics of REST?

Key characteristics of REST are likely asked in a Web API Testing interview. So please get the answer ready in your mind with these 2 ones:

- REST is stateless; therefore, the SERVER has no status (or session data). With a well-applied REST API, the server could be restarted between two calls, since all data is transferred to the server.
- Web service uses POST method primarily to perform operations, while REST uses GET for accessing resources.

33. What is messaging in RESTful Web services?

RESTful web services use the HTTP protocol as a communication tool between the client and the server. The technique that when the client sends a message in the form of an HTTP Request, the server sends back the HTTP reply is called Messaging. These messages comprise message data and metadata, that is, information on the message itself.

34. What are the core components of an HTTP request?

An HTTP request contains five key elements:

1. An action showing HTTP methods like GET, PUT, POST, DELETE.
2. Uniform Resource Identifier (URI), which is the identifier for the resource on the server.
3. HTTP Version, which indicates HTTP version, for example-HTTP v1.1.
4. Request Header, which carries metadata (as key-value pairs) for the HTTP Request message. Metadata could be a client (or browser) type, format supported by the client, format of a message body format, cache settings, and so on.
5. Request Body, which indicates the message content or resource representation.

35. What are the most commonly used HTTP methods supported by REST?

- GET is only used to request data from a specified resource. Get requests can be cached and bookmarked. It remains in the browser history and has length restrictions. GET requests should never be used when dealing with sensitive data.
- POST is used to send data to a server to create/update a resource. POST requests are never cached and bookmarked and do not remain in the browser history.
- PUT replaces all current representations of the target resource with the request payload.
- DELETE removes the specified resource.
- OPTIONS is used to describe the communication options for the target resource.
- HEAD asks for a response identical to that of a GET request, but without the response body.

36. Can GET request to be used instead of PUT to create a resource?

The PUT or POST method should be used to create a resource. GET is only used to request data from a specified resource.

37. Is there any difference between PUT and POST operations?

PUT and POST operation are quite similar, except the terms of the result generated by them.

PUT operation is idempotent, so you can cache the response while the responses to POST operation are not cacheable, and if you retry the request N times, you will end up having N resources with N different URIs created on server.

In a Web API Testing interview, you should give a specific example for PUT and POST operations to make crystal clear to the interviewer. Below is an example:

Scenario: Let's say we are designing a network application. Let's list down few URIs and their purpose to get to know when to use POST and when to use PUT operations.

GET /device-management/devices : Get all devices

POST /device-management/devices : Create a new device

GET /device-management/devices/{id} : Get the device information identified by "id"

PUT /device-management/devices/{id} : Update the device information identified by "id"

DELETE /device-management/devices/{id} : Delete device by "id"

38. Which purpose does the OPTIONS method serve for the RESTful Web services?

The OPTIONS Method lists down all the operations of a web service supports. It creates read-only requests to the server.

39. What is URI? What is the main purpose of REST-based web services and what is its format?

URI stands for Uniform Resource Identifier. It is a string of characters designed for unambiguous identification of resources and extensibility via the URI scheme.

The purpose of a URI is to locate a resource(s) on the server hosting of the web service.

A URI's format is <protocol>://<service-name>/<ResourceType>/<ResourceID>.

40. What is payload in RESTFul Web services?

The "payload" is the data you are interested in transporting. This is differentiated from the things that wrap the data for transport like the HTTP/S Request/Response headers, authentication, etc.

41. What is the upper limit for a payload to pass in the POST method?

<GET> appends data to the service URL. But its size shouldn't exceed the maximum URL length. However, <POST> doesn't have any such limit.

So, theoretically, a user can pass unlimited data as the payload to POST method. But, if we consider a real use case, then sending POST with large payload will consume more bandwidth. It'll take more time and present performance challenges to your server. Hence, a user should take action accordingly.

42. What is the caching mechanism?

Caching is just the practice of storing data in temporarily and retrieving data from a high-performance store (usually memory) either explicitly or implicitly.

When a caching mechanism is in place, it helps improve delivery speed by storing a copy of the asset you requested and later accessing the cached copy instead of the original.

43. What are SOAP Web services?

This is one of the fundamental Web services testing questions that you must know the answer. The SOAP (Simple Object Access Protocol) is defined as an XML-based protocol. It is known for designing and developing web services as well as enabling communication between applications developed on different platforms using various programming languages over the Internet. It is both platform and language independent.

44. How does SOAP work?

SOAP is used to provide a user interface that can be accessed by the client object, and the request that it sends goes to the server, which can be accessed using the server object. The user interface creates some files or methods consisting of server object and the name of the interface to the server object. It also contains other information such as the name of the interface and methods. It uses HTTP to send the XML to the server using the POST method, which analyzes the method and sends the result to the client. The server creates more XML consisting of responses to the request of user interface using HTTP. The client can use any approach to send the XML, like the SMTP server or POP3 protocol to pass the messages or reply to queries.

45. When to use SOAP API?

Use the SOAP API to create, retrieve, update or delete records, like accounts, leads, and user-defined objects. With more than 20 different calls, you can also use the SOAP API to manage passwords, perform searches, etc. by using the SOAP API in any language that supports web services.

46. How users utilize the facilities provided by SOAP?

- PutAddress(): It is used to enter an address in the webpage and has an address instance on the SOAP call.
- PutListing(): It is used to allow the insertion of a complete XML document into the web page. It receives the XML file as an argument and transports the XML file to XML parser liaison, which reads it and inserts it into the SOAP call as a parameter.
- GetAddress(): It is used to get a query name and gets the result that best matches a query. The name is sent to the SOAP call in the form of text character string.
- GetAllListing(): It is used to return the full list in an XML format.

47. What is the major obstacle users faced when using SOAP?

When using SOAP, users often see the firewall security mechanism as the biggest obstacle. This block all the ports leaving few like HTTP port 80 and the HTTP port used by SOAP that bypasses the firewall. The technical complaint against SOAP is that it mixes the specification for message transport with the specification for message structure.

48. What are the various approaches available for developing SOAP based web services?

There are two different methods available for developing SOAP-based web services, which are explained below:

- Contract-first approach: the contract is first defined by XML and WSDL, and then Java classes are derived from the contract.
- Contract-last approach: Java classes are first defined, and then the contract is generated, which is normally the WSDL file from the Java class.

"Contract-first" method is the most popular approach.

49. What are the elements of a SOAP message structure?

It is a common XML document that contains the elements as a SOAP message

Envelope: It is an obligatory root element that translates the XML document and defines the beginning and end of the message.

Header: It is an optional item which contains information about the message being sent.

Body: It contains the XML data comprising the message being sent.

Fault: It provides the information on errors that occurred while during message processing.

50. What are the syntax rules for a SOAP message?

- Must use encoded XML
- Envelope namespace must be used
- Encoding namespace must be used
- Must not consist of a DTD reference
- Must not have XML processing instruction

51. What is the transport method in SOAP?

Application layer and transport layers of a network are used by SOAP; HTTP and SMTP are the valid protocol of the application layer used as the transport for SOAP. HTTP is more preferable, since it works well with the current Internet infrastructure, in particular with firewalls. The SOAP requests can be sent using an HTTP GET method while the specification only contains details about HTTP POST.

52. What are some important characteristics of a SOAP envelope element?

- SOAP message has a root Envelope element
- Envelope is an obligatory part of the SOAP message.
- If an envelope includes a header element, it should not contain more than one.
- Envelope version will change if the SOAP version changes.
- The SOAP envelope is indicated by the prefix ENV and the envelope element.
- The optional SOAP encoding is also specified using a namespace and the optional encoding style element.

53. What are the major functionalities provided by the SOAP protocol class?

The SOAP protocol is used to provide simple access methods for all the applications available on the Internet, providing the following functionalities:

- ***Call***: A class which provides the main functionality for a remote method for which a call is needed. It is used to create the call() and to specify the encoding style of the registry that will be assigned when if necessary. This call() function is used by the RPC call, which represents the options of the call object.

- ***Deployment Descriptor***: A class used to provide the information about the SOAP services. It enables easy deployment without the need for other approaches.
- ***DOM2 Writer***: A class that serializes and uses DOM node as XML string to provide more functionalities.
- ***RPC Message***: A class used as the base class that calls and replies to the request submitted to the server.
- ***Service Manager***: A class that provides, lists and then outputs all SOAP services.

54. What are the web relation functionalities provided by SOAP protocol?

- ***HTTPUtils***: This provides the functionality of the POST method to safely meet the requirements.
- ***Parameter***: It is an argument for an RPC call used by both the client and the server.
- ***Response***: It is an object that represents an RPC reply from both client and server, but the result will not be displayed until after the method call.
- ***TCPTunnel***: It is an object that provides the ability to listen on a specific port and to forward all the host and port names.
- ***TypeConverter***: It helps to convert an object of one type into another type and this is called using the class in the form object.

55. How does the message security model allow the creation of SOAP more secure to use?

The security model includes the given security tokens. These tokens comprise digital signatures for protection and authentication of SOAP messages. Security tokens can be used to provide the bond between authentication secrets or keys and security identities. Security token uses the authentication protocols and an X.509 certificate to define the relationship between the public key and identity key. The signatures are used to verify the messages and their origin, generate knowledge to confirm the security tokens to bind the identity of a person to the identity of the originator. Security model prevents different attacks and can be used to protect the SOAP architecture.

56. What is the difference between top down & bottom up approach in SOAP Web services?

- Top down SOAP Web services include creating WSDL document to create a contract between the web service and the client, with a required code as an option. This is also known as Contract-first approach. The top-down approach is difficult to implement because classes must be written to confirm the contract defined in WSDL. One of the benefits of this method is that both client and server code can be written in parallel.
- Bottom up SOAP web services require the code to be written first and then WSDL is generated. It is also known as Contract-last approach. Since WSDL is created based on the code, bottom-up approach is easy to implement and client codes must wait for WSDL from the server side to start working.

57. What are advantages of SOAP?

- SOAP is both platform and language independent.
- SOAP separates the encoding and communications protocol from the runtime environment.
- Web service can retrieve or receive a SOAP user data from a remote service, and the source's platform information is completely independent of each other.
- Everything can generate XML, from Perl scripts through C++ code to J2EE app servers.
- It uses XML to send and receive messages.
- It uses standard internet HTTP protocol.
- SOAP runs over HTTP; it eliminates firewall problems. When protocol HTTP is used as the protocol binding, an RPC call will be automatically assigned to an HTTP request, and the RPC response will be assigned to an HTTP reply.
- Compared to RMI, CORBA and DCOM, SOAP is very easy to use.
- SOAP acts as a protocol to move information in a distributed and decentralized environment.

- SOAP is independent of the transport protocol and can be used to coordinate different protocols.

58. What are disadvantages of SOAP?

SOAP is typically significantly slower than other types of **middleware** standards, including CORBA, because SOAP uses a detailed XML format. A complete understanding of the performance limitations before building applications around SOAP is hence required.

SOAP is usually limited to pooling and not to event notifications when HTTP is used for the transport. In addition, only one client can use the services of one server in typical situations.

If HTTP is used as the transport protocol, firewall latency usually occurs since the firewall analyzes the HTTP transport. This is because HTTP is also leveraged for Web browsing, and so many firewalls do not understand the difference between using HTTP within a web browser and using HTTP within SOAP.

SOAP has different support levels, depending on the supported programming language. For instance, SOAP supported in **Python** and PHP is not as powerful as it is in Java and .NET

59. What are the differences between SOAP and REST?

SOAP	REST
1. SOAP is a protocol. SOAP was designed with a specification. It includes a WSDL file which has the required information on what the web service does in addition to the location of the web service.	1. REST is an Architectural style in which a web service can only be treated as a RESTful service if it follows the constraints of being: • Client Server • Stateless • Cacheable • Layered System • Uniform Interface
2. SOAP cannot make use of REST since SOAP is a protocol and REST is an architectural pattern.	2. REST can make use of SOAP as the underlying protocol for web services, because in the end it is just an architectural pattern.
3. SOAP only permits the XML format.	3. REST permits many different data formats.
4. SOAP based reads cannot be cached.	4. REST reads can be cached.
5. SOAP is like custom desktop application, closely connected to the server.	5. A REST client is just like a browser and uses standard methods. An application has to fit inside it.
6. SOAP is slower than REST.	6. REST is faster than SOAP.
7. It runs on HTTP but envelopes the message.	7. It uses the HTTP headers to hold meta information.

60. SOAP or Rest APIs, which method to use?

SOAP is the heavyweight choice for Web service access. It provides the following advantages when compared to REST:

- SOAP is not very easy to implement and requires more bandwidth and resources.
- SOAP message request is processed slower as compared to REST and it does not use web caching mechanism.
- WS-Security: While SOAP supports SSL (just like REST) it also supports WS-Security which adds some enterprise security features.
- WS-AtomicTransaction: Need ACID Transactions over a service, you're going to need SOAP.
- WS-ReliableMessaging: If your application needs Asynchronous processing and a guaranteed level of reliability and security. Rest doesn't have a standard messaging system and expects clients to deal with communication failures by retrying.
- If the security is a major concern and the resources are not limited then we should use SOAP web services. Like if we are creating a web service for payment gateways, financial and telecommunication related work, then we should go with SOAP as here high security is needed.

REST is easier to use for the most part and is more flexible. It has the following advantages when compared to SOAP:

- Since REST uses standard HTTP, it is much simpler.
- REST is easier to implement, requires less bandwidth and resources.
- REST permits many different data formats whereas SOAP only permits XML.
- REST allows better support for browser clients due to its support for JSON.
- REST has better performance and scalability. REST reads can be cached, SOAP based reads cannot be cached.
- If security is not a major concern and we have limited resources. Or we want to create an API that will be easily used by other developers publicly then we should go with REST.
- If we need Stateless CRUD operations then go with REST.
- REST is commonly used in social media, web chat, mobile services and Public APIs like Google Maps.
- RESTful service returns various MediaTypes for the same resource, depending on the request header parameter "Accept" as application/xml or application/json for POST and /user/1234.json or GET /user/1234.xml for GET.
- REST services are meant to be called by the client-side application and not the end user directly.
- ST in REST comes from State Transfer. You transfer the state around instead of having the server store it, this makes REST services scalable.

61. What are the factors that help to decide which style of Web services – SOAP or REST – to use?

Generally, REST is preferred due to its simplicity, performance, scalability, and support for multiple data formats.

However, SOAP is favorable to use where service requires an advanced level of security and transactional reliability.

But you can read the following facts before opting for any of the styles.

- ***Does the service expose data or business logic?*** REST is commonly used for exposing data while SOAP for logic.
- ***The requirement from clients or providers for a formal contract.*** SOAP can provide contract via WSDL.

- ***Support multiple data formats****.*
- ***Support for AJAX calls.*** REST can apply the XMLHttpRequest.
- ***Synchronous and asynchronous calls.*** SOAP enables both synchronous/ asynchronous operations whereas REST has built-in support for synchronous.
- ***Stateless or Stateful calls.*** REST is suited for stateless operations.
- ***Security.*** SOAP provides a high level of security.
- ***Transaction support.*** SOAP is good at transaction management.
- ***Limited bandwidth****.* SOAP has a lot of overhead when sending/receiving packets since it's XML based, requires a SOAP header. However, REST requires less bandwidth to send requests to the server. Its messages are mostly built using JSON.
- ***Ease of use****.* REST based application is easy to implement, test, and maintain.