TABLE OF CONTENTS

Preface

In recent years, many testers who have applied for QA jobs would undoubtedly have to go through an interview process. For your benefit as you prepare ready for a new position, the API testing interview questions below have been compiled from my many resources and my interview experiences in the past 7 years. This list of topics is useful for more than just API interviews; it will also be helpful to new and experienced testers who want to learn both fundamental and complex concepts about web API testing.

I hope this note could help fresh graduates who seek a career as a Software Test Engineer or someone who doesn't have much time to prepare for the software Test Engineer Interview.

Lamhot Siagian
Software Engineer in Test Consultant

Common Question and Answer for API Testing Interview

What is an API?

API stands for ***Application Programming Interface*** which is useful for communication between different software systems. It facilitates data exchange between systems located in different remote places. API is a collection of functions that are executable by other functions of the applications (Web, Desktop, iOS App, and Android Apk).

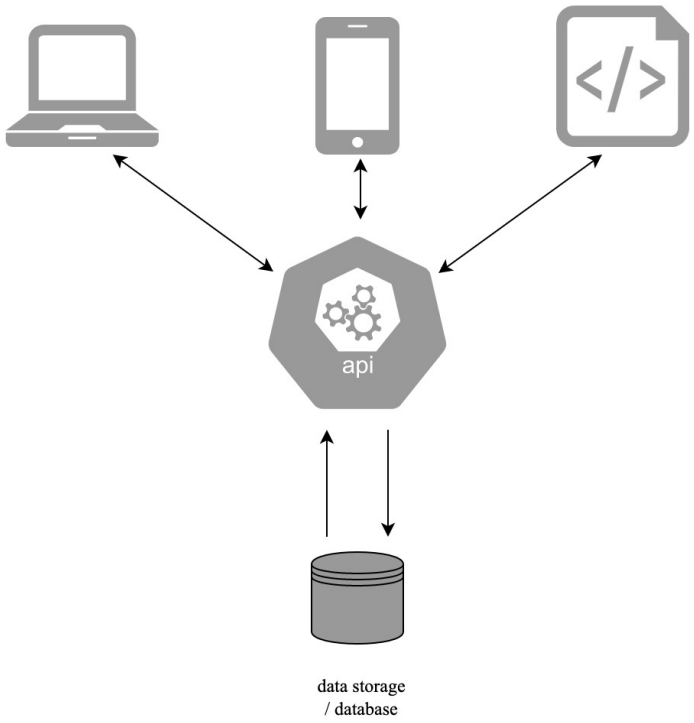


Figure 1.1 API Diagram

When you use a mobile application, browser, or desktop application, it connects to the Internet and sends information to a server. The data is subsequently retrieved, interpreted, and sent back to your phone by the server. The application then analyzes the data and displays the information you requested in a comprehensible manner. All of this occurs through an API.

Here sample of API requests:

curl --location --request GET 'https://dummyjson.com/products/1'

Sample response:

```
{
  "id": 1,
  "title": "iPhone 9",
  "description": "An apple mobile which is nothing like apple",
  "price": 549,
  "discountPercentage": 12.96,
  "rating": 4.69,
  "stock": 94,
  "brand": "Apple",
  "category": "smartphones",
  "thumbnail": "https://dummyjson.com/image/i/products/1/thumbnail.jpg",
  "images": [
    "https://dummyjson.com/image/i/products/1/1.jpg",
    "https://dummyjson.com/image/i/products/1/2.jpg",
    "https://dummyjson.com/image/i/products/1/3.jpg",
    "https://dummyjson.com/image/i/products/1/4.jpg",
    "https://dummyjson.com/image/i/products/1/thumbnail.jpg"
  ]
}
```


What is API Testing?

API (Application Programming Interface) is a computing interface that enables communication and data exchange between two separate software systems. A software system that executes an API includes several functions/subroutines that another software system can perform. API defines requests that can be made, how to make requests, data formats that can be used, etc.

How does SOAP compare to REST?

REST or representational state transfer is an architectural style, rather than a protocol. What this means is that REST provides much more flexibility in terms of how you structure your message, which format you use, and how the client and the server scale. SOAP, on the other hand, requires tight coupling between client and server. If either side changes something, things go wrong, hence its protocol nature.

	SOAP	**REST**
Nature	Strict protocol	Architecture pattern
State	Stateful, stateless	Stateless
Format	XML	XML, JSON, HTML, plain text

Message	Evenvelope	Postcard
Transfer protocol	HTTP/HTTPS, FTP, TCP, SMTP, XMPP	HTTP/HTTPS
Logic exposure	WSDL	WADL
Caching	Caching module	HTTP-caching
Security	WS-Security, ACID, HTTPS, SSL	HTTPS, SSL
Speed	slow	Fast
Learning Curve	Difficult	Easy
Community	small	Large

REST was introduced in 2000 – it's not much younger than SOAP – with the idea of making servers care less about what's happening to the client.[1]

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 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

[1] https://www.altexsoft.com/blog/engineering/what-is-soap-formats-protocols-message-structure-and-how-soap-is-different-from-rest/

test automation requires less code so it can provide better and faster test coverage compared to GUI test automation. These will result in cost savings 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 coding 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.

What Is The Difference Between Web Services and APIs

Web Serviced	**API**
All web services are APIs.	All APIs are not web services.
It supports XML.	Responses are formatted using Web API's MediaTypeFormatter into XML, JSON, or any other given format.
You need a SOAP protocol to send or receive data over the network. Therefore it does not have lightweight architecture.	API has a lightweight architecture.
It can be used by any client who understands XML.	It can be used by a client who understands JSON or XML.
Web service uses three styles: REST, SOAP, and XML-RPC for communication.	API can be used for any style of communication.
It provides support only for the HTTP protocol.	It provides support for the HTTP/s protocol: URL Request/Response Headers, etc.

What are HTTP status codes?

Browsers and servers interact via the Hypertext Transfer Protocol (HTTP) status code. The browser transmits your request to a server, which then delivers a response when you use the internet to seek information from a server. The HTTP status code, which informs the browser of the request's status and whether or not something went wrong, as well as the reason for the error, is a component of that response. The requested page, the client's browser, and the server are among the details carried by HTTP headers for requests and answers, and these details can determine the status codes.

What are common HTTP status codes?

Numerous HTTP status codes are available. Wikipedia now lists 75 different status codes, the majority of which you've probably never heard of.[2]

1xx informational response

Communicates transfer protocol-level information.

1. ***100 Continue***, this response means everything is functioning properly and can proceed. The code indicates the server is processing the request and can complete it. The 100 code is an interim response and tells you the request has not caused an error.
2. ***101 Switching Protocol***, the server received the data request and can comply. The server, however, will change protocols based on information in the Upgrade header field. This status code does not indicate an error.
3. ***102 Processing***, the server received the request, but there isn't any response yet. The server is processing the request, which is ongoing. This code also indicates an error has not occurred.

2xx success

The client accepts the request and is processed it successfully at the server.

1. ***203 Non-Authoritative Information***, the returned data does not match the data available on the server. This often can indicate the data arrived via a proxy source of the server that contained the original data. This code means the server received a 200 OK from its origin, but it's returning a modified version of the response.
2. ***204 No Content,*** the server sends this code when it received and understood the request. With this code, though, the server is signaling it has no data to return. Even though there's no content to send, this code indicates the HTTP headers are proper.
3. ***205 Reset Content***, the server has completed the request, but there was a problem. This status code represents a correctable error. You can solve this by resetting the document view and trying again.

3xx redirection

Indicates that the client must take some additional action in order to complete their request. Most of the codes related to this series are for URL Redirection.

[2] https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

1. ***300 Multiple choices,*** The request has more than one potential response. The server often sends a report with this code that includes a list of the potential choices. To get the data, you can choose from the responses the server provides.
2. ***301 Moved permanently,*** This code indicates the URL no longer exists or is broken. This message often includes the new URL for you to use. It also signifies to the user to update all references to this URL.
3. ***302 Found***, The Uniform Resource Identifier (URI) changed, but the change is not permanent. A 302 Found means more changes might occur in the future, so consider using the new URI to complete requests. This code is one way to create a temporary redirect.
4. ***305 Use proxy,*** The request needs to use a server proxy to complete successfully. This message tells the user to send the request again. The second request will go to the server proxy, which can provide the data.
5. ***307 Temporary redirec***t, This response means the Uniform Resource Identifier is different. The server has to use a new, temporary URI to return data. This code also indicates future requests likely won't use the temporary URI and will function as normal.

4xx client errors

This category of error status codes points the finger at clients (these are specific to client-side errors.)

1. ***400 Bad request***, this code indicates invalid syntax prevented the server from receiving a request. Servers return this code when the requester made the error and should correct it before trying again. This could mean the client used invalid message framing or improperly routed the message.
2. ***401 Unauthorized***, the server cannot complete the request. The server believes the user does not have the authorization to receive the data. After sending this code, a server won't complete the request until user authentication happens.
3. ***403 Forbidden,*** the server understood the request, but it is refusing to fulfill it. Sometimes, the user does not have access to the requested content or data, and the server rejects the request. This also can mean the client needs an account to request data or the request would create a duplicate record.
4. ***404 Not found***, the 404 error is one of the most common codes servers send. The error means the server can't find information in that location now, but it might be able to in the future. It also can occur when a page has a new location, but a user didn't provide the old one with a redirect.
5. ***408 Request timeout***, this code means a request took too long to process. It also could indicate the request didn't make it through to the server. Sometimes, internet problems can interrupt the process, but the user can try again.
6. ***410 Gone***, the information a user requested from the server no longer exists. This code means the information existed at some point, but it doesn't anymore. The server also can't tell you whether the information exists somewhere else.
7. ***422 Unprocessable Entity***, the request was well-formed but was unable to be followed due to semantic errors.
8. ***429 Too Many Requests***, The user has sent too many requests in a given amount of time.

5xx server errors

The browser makes a valid request, but an error happened on the server side of the process (these are specific to the server-side error).

1. ***500 Internal Server Error***, a generic error message, is given when an unexpected condition was encountered and no more specific messages are suitable.
2. ***501 Not Implemented***, the server either does not recognize the request method or lacks the ability to fulfill the request. Usually, this implies future availability (e.g., a new feature of a web-service API).
3. ***502 Bad Gateway,*** the server was acting as a gateway or proxy and received an invalid response from the upstream server.
4. ***503 Service Unavailable***, the server cannot handle the request (because it is overloaded or down for maintenance). Generally, this is a temporary state
5. ***504 Gateway Timeout,*** the server was acting as a gateway or proxy and did not receive a timely response from the upstream server
6. ***505 HTTP Version Not Supported***, the server does not support the HTTP protocol version used in the request.

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 the following nine categories that you should remember before attending 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

What are the Limits of API Usage?

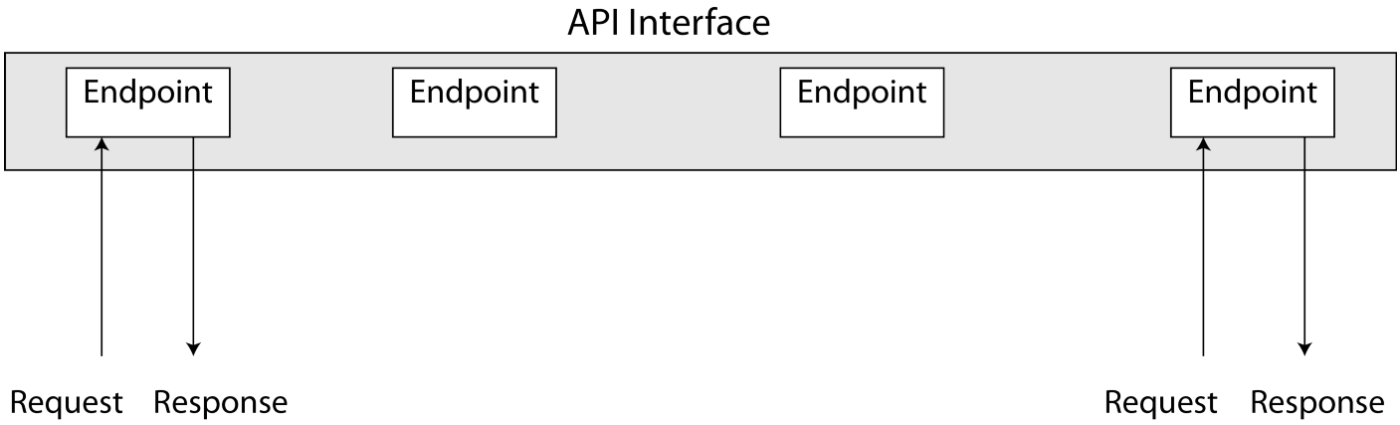
Many APIs have a certain limit set up by the provider. Thus, try to estimate your usage and understand how that will impact the overall cost of the offering. Whether this will be a problem depends in large part on how data is leveraged. Getting caught by a quota and effectively cut off because of budget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

What is an API, and what is an Endpoint?

An API is a set of protocols and tools that allow two applications to communicate. The two applications can sit on the same machine (your current Application, that is your Web Browser, communicates with the Operating System on your machine to display this article).

The use of an API means that an interface is public so programs can easily communicate with each other and act in expected ways (forming a contract across the interface).

On the other hand, an Endpoint is a URL that enables the API to access resources on a server, often through a RESTful API interface.



The interface can (as shown above), provide a series of Endpoints that can be called at any time.[3]

How do you validate the Endpoints?

- Find out which endpoints your web service provides
- Each collection endpoint needs to be tested

[3] https://stevenpcurtis.medium.com/endpoint-vs-api-ee96a91e88ca

- At least one single resource endpoint needs to be tested for each resource type

Negative tests: Try to request an endpoint that does not exist

How do you validate Search, Filtering, Sorting Endpoint?

Endpoints that support search, filtering, sorting, etc. need to be verified with supported parameters separately and in combinations. Use boundary values, equality partitioning, pairwise testing, check special characters, max length parameters, etc.

http://example.com/api/products?q=name:Keyboard&maxPrice:200

http://example.com/api/products?year:2018

http://example.com/api/products?sort:name,asc

Negative tests: Try to request search/filtering/sorting with wrong parameter/value

How do you validate the Request Methods?

Positive tests

- For each collection endpoint and at least one single resource endpoint for each resource type verify supported request methods
- Verify concurrent access to resources (DELETE and GET for example)

Negative tests

For each collection endpoint and at least one single resource endpoint for each resource type try to verify the behavior for not supported request methods

How do you validate the Request Header?

Positive

- For each collection endpoint and at least one single resource endpoint verify all supported request methods with correct header values - with required headers only first, then verifying optional ones one at a time and combinations
- Use boundary values, equivalence partitioning, pairwise testing. Verify special characters, Unicode text for headers, and max length values.

Negative

- Verify behavior in case of missing required header, one at a time
- Verify behavior in case of wrong/unsupported/empty header value
- Verify behavior in case of not supported header

How do you validate Request Body?

Possible negative tests for POST/PUT requests:

- Field contains an invalid value (not equal allowed value, out of bounds, etc) • Field of wrong data type
- Field value is empty object/string
- Field value is null
- Required field is absent
- Redundant field (“Add to custom fields” example)
- Empty object {}
- Invalid XML/JSON
- No data
- Post already existing resource
- Possible negative tests for DELETE requests: Delete non-existing resource

How do you validate Rate Limits and Caching in API?

- Verify that cached responses are received faster
- Verify cache expiration time (right before and after) • Find out rate limits for different methods/endpoints • Try to reproduce max allowed rate limit
- Try to reproduce more than max allowed rate limit

How do you validate Pagination, Cursors?

- Endpoints that support pagination, and cursors need to be verified with supported parameters separately and in combinations. Use boundary values, equality partitioning, and pairwise testing.
- For negative tests try to use a limit more than max one, offset out of bounds, incorrect values

http://example.com/api/products?limit=20&offset=100

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 communication

Who can use a Web API?

Web API can be consumed by any client which supports HTTP verbs such as GET, PUT, DELETE, and 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.

What are the request methods for API?

1. 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.
2. 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.
3. PUT replaces all current representations of the target resource with the request payload.
4. DELETE removes the specified resource.
5. OPTIONS is used to describe the communication options for the target resource.

6. HEAD asks for a response identical to that of a GET request, but without the response body.

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 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 cost savings 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 coding 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.

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.

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.

What are the 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

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 the following nine categories that you should remember before attending 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

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

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

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.

What tools could be used for API testing?

There are several tools to test the APIs. When a tester gets to test API, they must ask for its document, whether it is a REST or SOAP API or its not-web based API there should always be a document where the details should be written. To approach API testing-

1. Ask for Doc

2. Write functional or service level cases first
3. Write integration tests
4. When API is stable enough and passes most of the above tests, perform security, performance, and load testing.

- A typical API doc has all the information related to the API like its request format, response, error codes, resource, mandatory parameters, optional parameters, headers, etc. The doc can be maintained in various tools like swagger which is open source, Dapperdox, ReDoc, etc.
- After that try to write service-level cases for API. For example, if an API takes n parameters to get the response in which m is mandatory params and others are optional, then one test case should be to try different combinations of parameters and verify the response. Another test case might verify the headers and try to run API without passing authentication and verify the error code.
- Next comes to the step of integration test, where you need to test the API and all its dependent APIs or functions. This also includes testing API response, the data it should return to another API or method, and what happens if this API fails.
- Once the API is stable and functional testing is almost done, the tester can perform load, security, and performance testing.

Automation Test tools for API:

We often need to automate the test cases which are repeatedly executed. For eg- Regression cases. Similarly, in the case of API testing, there might be some cases that we need to execute before every release and those cases can be automated.

There are many tools for API automation that are quite popular-

1. SOUP UI
2. Katalon studio
3. Postman
4. Jmeter
5. RestAssured
6. CloudQA TruAPI

What are the 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.

What are the differences between API Testing and UI Testing?

- API enables the 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 the look and feel of an application.

What are the 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
- Another important challenge is providing input values, which is very difficult as GUI is not available in this case.

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

What are common API errors that are often found?

Not only API fundamental questions, but the interviewer also determines 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.

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

What is the API documentation?

The API documentation is 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.

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

APIs, especially modern RESTful APIs, is 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.

What is REST?

REST (Representational State Transfer) is an architectural style for developing web services that exploit the ubiquity of HTTP protocol and uses the HTTP method to define actions. It revolves around resources where every component is 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 a 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.

What is a RESTFul Web Services?

Mostly, there are two kinds of Web Services that 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, that provides resource representation like JSON and a set of HTTP methods.

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.

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

REST uses different representations to define a resource like text, JSON, and XML.

XML and JSON are the most popular representations of resources.

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.

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.

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.

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, and 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.

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.

Is there any difference between PUT and POST operations?

PUT and POST operations are quite similar, except for 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 the 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 a 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"

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

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

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>.

What is the 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.

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 the POST method. But, if we consider a real use case, then sending a POST with a 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.