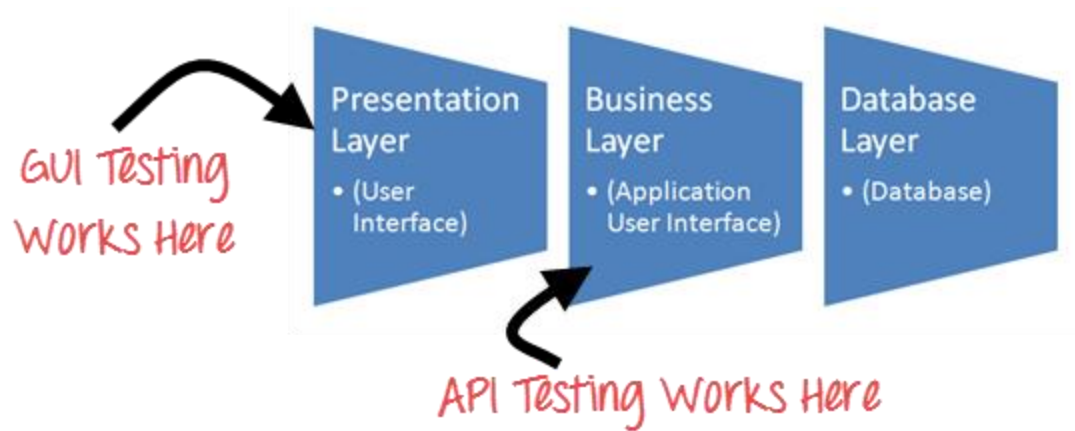# API TESTING

API testing is a type of [software testing](#) that analyzes an application program interface (API) to verify that it fulfills its expected functionality, security, performance and reliability. The tests are performed either directly on the API or as part of [integration testing](#).

An API is code that enables the communication exchange of data between two software programs. An application typically consists of multiple layers, including an API layer. API layers focus on the [business logic](#) in applications, defining requests such as how to make them and the data formats used.

As opposed to user interface (UI) testing, which focuses on validating the application's look and feel, API testing focuses on analyzing the application's business logic as well as security and data responses. An API test is generally performed by making requests to one or more API endpoints and comparing the responses with expected results.

Web Services
Web Services are (like Web API) the services that serve from one machine to another. But the major difference that arises between API and Web Services is that the Web Services uses a network.

It is safe to say that all Web Services are Web APIs but all Web APIs are not Web Services (explained in the latter part of the article). Thus, Web Services are a subset of Web API. Refer to the below diagram to know more about Web API and Web Services.

Web API vs Web Services
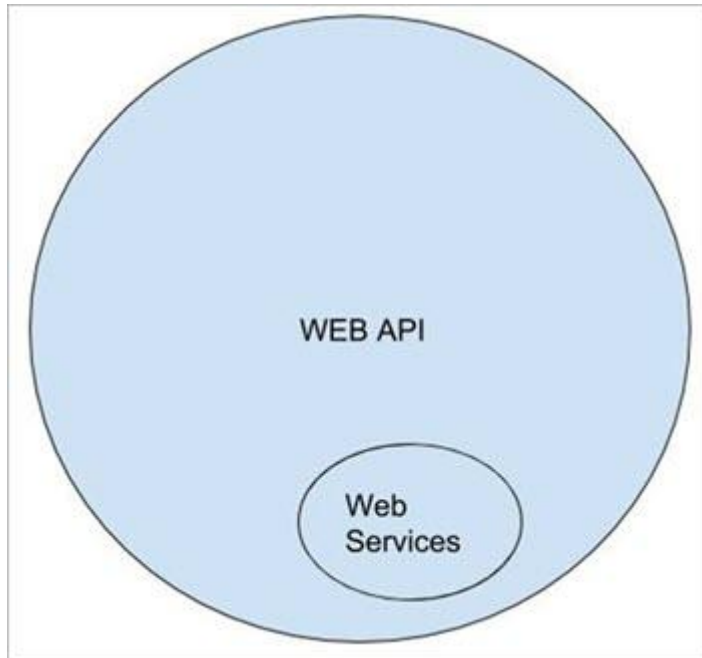
Web Services vs Web API:

Both Web API and Web Services are used to facilitate the communication between the client and the server. The major difference comes only in the way they communicate.

Each of them requires a request body that is acceptable in a specific language, their differences in providing a secure connection, their speed of communicating to the server and responding back to the client, etc.

The differences between Web Services vs API

As discussed above, we have known about Web Services. You're probably wondering "What are the differences between APIs and Web Services?". In order to answer this question, let's take a look at what an API is and the analogy between APIs and Web Services.

API stands for Application Programming Interface which is a protocol used as an interface by software components to communicate with each other. An API serves as an interface between two different applications so that they can communicate with each other.

APIs and Web Services both are means of communication between service providers and service consumers. They are usually mistaken for each other but there are many differences between them:

WEB SERVICE	API
All web services are APIs.	All APIs are not web services.
It can only be hosted on IIS.	It can be hosted within an application or IIS.
It is not open source but can be used by any client that understands XML.	It is open source and it can be used by any client that understands JSON or XML.
It requires a SOAP protocol to receive and send data over the network, so it is not a	It is light-weight architectured and good for devices

light-weight architecture.	which have limited bandwidth, like mobile devices.
A Web service uses only three styles of use: SOAP, REST and XML-RPC for communication.	API may use any style of communication.
It only supports the HTTP protocol.	It supports the HTTP protocol: URL, Request/Response Headers, caching, versioning, content formats.

Here are some test cases you might want to consider:

- **Test API authentication:** Ensure that the API requires authentication to access it. Verify that a valid user can log in and receive an authentication token, while an invalid user is denied access.

- **Test API response codes:** Check that the API returns the expected HTTP status codes for various requests, such as success (200), bad request (400), unauthorized (401), and not found (404).

- **Test CRUD operations:** Verify that users can create, read, update, and delete to-do lists and their associated tasks. For example, create a new to-do list, verify that it appears in the list of user's to-do lists, update an existing task, and confirm that the changes are reflected in the backend.

- **Test input validation:** Ensure that the API properly handles invalid input, such as empty fields or incorrect data types. Verify that the API returns appropriate error messages for these scenarios.

- **Test API security:** Verify that the API is secure and protected from common security threats such as SQL injection and cross-site scripting (XSS) attacks.

- **Test API performance:** Test the API's response times and verify that it can handle a high volume of requests without performance degradation.

By testing these and other relevant scenarios, you can ensure that the API backend of the mobile app functions as expected and provides a seamless user experience.

Why Testing APIs?

API testing is important to ensure the quality of your web and mobile apps. API testing can be done by developers, QA engineers and business analysts.

API testing is not just about testing the API endpoint and response; it is also about testing the backend of your mobile app.

You can also perform API testing on a mobile device or emulator. You can use your own device to test the app, or you can use a cloud-based tool such as Google Cloud Test Lab.

API testing is also an important part of mobile app development. When you are building a mobile app, you need to make sure that the API endpoint works well with your backend.

Real world examples

Example 1: Let's compare it with a real-world example: You go to a restaurant you sit at your table and you choose that you need ABC. You will have your waiter coming up and taking a note of what you want. You tell him that you want ABC. So, you are requesting ABC, the waiter responds back with ABC he gets in the kitchen and serves you the food. In this case, who is your interface in between you and the kitchen is your waiter. It's his responsibility to carry the request from you to the kitchen, make sure it's getting done, and you know once it is ready he gets back to you as a response.

Example 2: Another important example that we can relate is travel booking systems. For instance, take Kayak the biggest online site for booking tickets. You enter your destination, once you select dates and click on search, what you get back are

the results from different airlines. How is Kayak communicating with all these airlines? There must be some ways that these airlines are actually exposing some level of information to Kayak. That's all the talking, it's through API's

Example 3: Now open UBER and see. Once the site is loaded, it gives you an ability to log in or continue with Facebook and Google. In this case, Google and Facebook are also exposing some level of users' information. There is an agreement between UBER and Google/Facebook that has already happened. That's the reason it is letting you sign up with Google/Facebook.
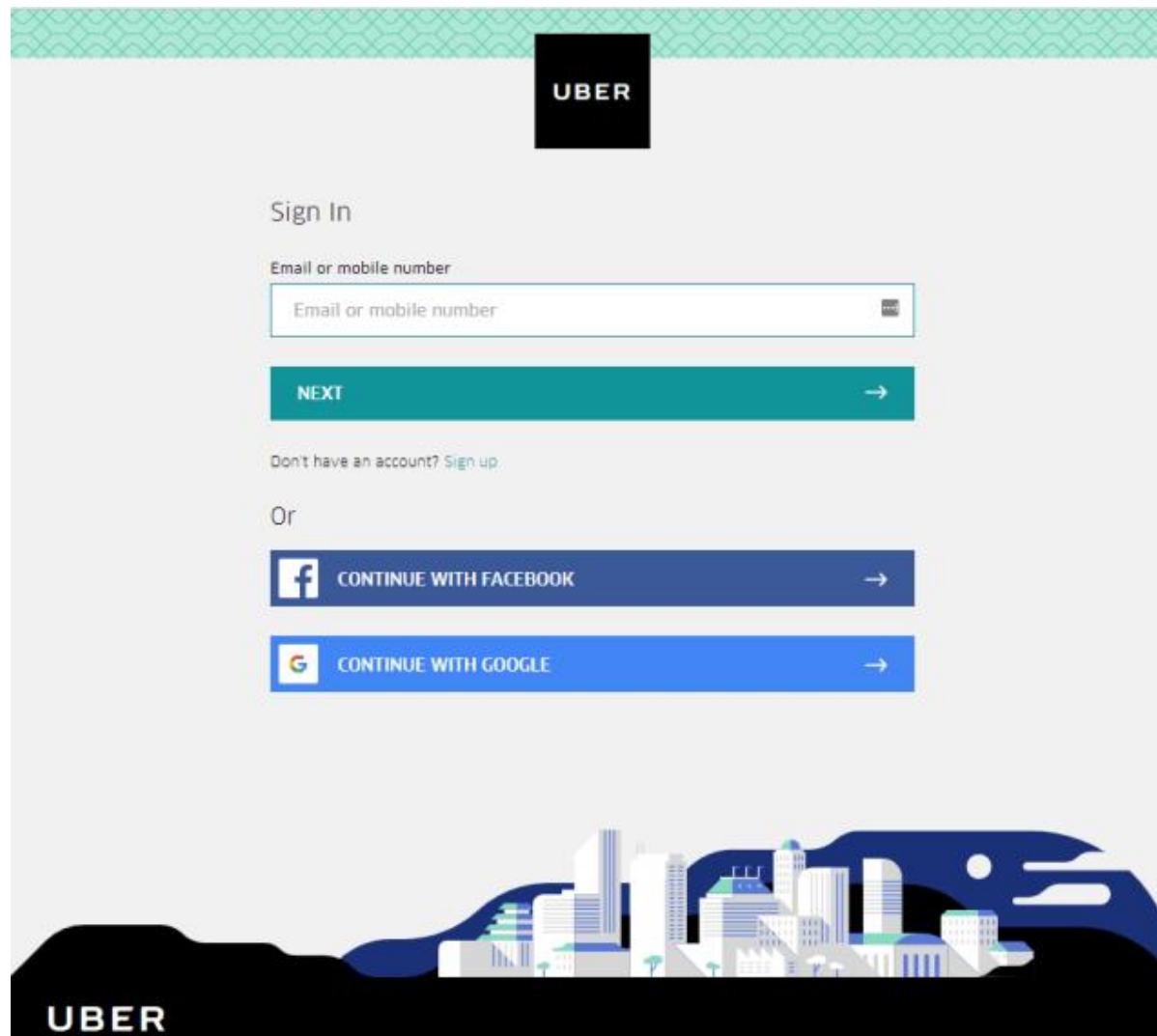
What is the use of API's?

If I expose a free API, it's not that I am exposing all the information that there is. I have my own criteria like what level of information that I should expose to my user or to the outside world. So I tie that information and I just expose it, now people use it. Tomorrow, they want some extra level of information, they comfort me and ask me for that. Why should I give it for free? That's when I charge based on the demand, I do make money out of my extra services.

Think about the development perspective, UBER has an ability to share location where it uses Google Map API there. UBER has to focus on what it is actually doing rather than already existing functionality. It can just plug and play. There is no reinvestment of time to develop the same functionality. That's how people expose some level of API's to the outside world and some of them will be charged based on how you are using it.

How to test APIs?

There are many ways to test APIs, including using tools like **Postman** and curl command line tools, or even an API testing framework like **SoapUI**. The most popular method for web and mobile API testing is to use a tool like **JMeter, Katalon, Apigee, or Karate** to automate API testing.

API testing with Postman:

- **Install Postman**: First, you need to download and install Postman. Postman is a popular API testing tool that allows you to test APIs by sending HTTP requests to the server and receiving responses.

- **Create a new request**: To create a new request in Postman, click on the "New" button on the top-left corner of the application. Then, select the type of request you want to send (e.g. GET, POST, PUT, DELETE) and enter the URL of the API endpoint you want to test.

- **Add request parameters**: Depending on the API endpoint you're testing, you may need to add request parameters to your request. These parameters can include headers, query parameters, and request body data. To add parameters in Postman, click on the "Params" or "Body" tabs, depending on the type of parameter you want to add.

- **Send the request**: Once you've added all the necessary parameters to your request, click on the "Send" button to send the request to the server. Postman will display the response from the server, including the status code, response headers, and response body.

- **Validate the response**: After you've received the response from the server, you need to validate it to ensure that it meets your expectations. You can use Postman's built-in assertions to validate the response, such as checking the status code, response headers, and response body.

- **Create test suites**: Once you've created a few requests and validated their responses, you can create test suites to automate your API testing. Test suites allow you to group related tests together and run them all at once. You can also set up assertions for each test to ensure that they pass or fail based on specific criteria.

Overall, [Postman](#) allows you to test APIs quickly and easily. It's easy to get started with, and you can create more advanced tests and test suites as you become more familiar with the tool.

API testing with SoapUI:

SoapUI is another popular tool for API testing that provides a comprehensive set of features to help testers create and execute automated functional, regression, compliance, and load tests for REST and SOAP APIs. Some of the key features of SoapUI include:

- **Easy API Discovery**: SoapUI allows users to easily discover APIs by importing WSDL, Swagger, or RAML definitions, making it easy to get started with testing.

- **Automated Testing**: SoapUI allows testers to create automated tests using its intuitive drag-and-drop interface, without needing to write any code. It also supports scripting languages like Groovy and JavaScript for more advanced test scenarios.

- **Test Data Generation**: SoapUI provides tools for generating test data and creating data-driven tests, allowing testers to cover a wide range of test scenarios.

- **Mock Services**: SoapUI allows testers to create mock services that simulate the behavior of the actual API, allowing testing to proceed even when the actual API is not yet available.

- **Integration with CI/CD**: SoapUI integrates with popular CI/CD tools like Jenkins, Bamboo, and TeamCity, allowing teams to include API testing as part of their automated testing pipeline.

Overall, [SoapUI](#) offers a wide range of features and customization options. With SoapUI, testers can easily create and execute automated tests for their REST and SOAP APIs, ensuring that their APIs are working as expected and meeting the requirements of their users.

API testing with the Katalon Platform

The Katalon Platform is a powerful test automation solution that allows you to easily create and run tests for web and mobile applications, including APIs.

Some additional features of Katalon that can be useful for API testing include:

- **Data-driven testing**: Katalon allows you to use data from external sources, such as CSV or Excel files, to drive your API tests. This can help you test your API with different inputs and scenarios.

- **Assertions**: Katalon provides a range of built-in assertions that you can use to verify the responses from your API endpoint. This includes checking response codes, response headers, and response content.

- **Reports**: Katalon generates detailed reports for each test case, including screenshots and logs. This can help you identify any issues or errors that occur during your API testing.

- **Seamlessly integrations**: Katalon allows users to automate API testing without depending on third-party tools or configurations while leveraging all the benefits of the Katalon Platform.

- **Flexible Parametrization**: with Katalon, users can parametrize any value within the API interface and use external data sources like excel, text files, databases, etc. to inject values into the API parameters. This technique enables users to use the same test case to test on multiple environments and streamline the testing cycle.

Overall, Katalon is a powerful tool for API testing that can help you ensure the quality and reliability of your web and mobile applications.

Web API Testing vs Mobile API Testing?

Web API testing is the process of testing the backend of a mobile app, while mobile API testing is the process of testing the actual app itself.

Web API testing is more complex than mobile API testing because it involves more components like databases and servers.

Mobile developers can start with web API testing and then move on to mobile API testing once they're comfortable with their understanding of how data flows through their system.

How does API Testing work?

It is all about requests and responses between the client and server. You want something, you request for something so there is a way it gets carried to the server and you get the response back from the server. In other words, it's all about your requesting and you know getting some responses from the provider.

Top 15 API Testing Tools

SoapUI · Katalon · POSTMAN · Assertible · Swagger · apigee · APACHE jMeter · REST-assured · TRICENTIS · Karate · REST Console · APIFORTRESS · Pyresttest · Fiddler · Airborne

SOAP vs REST

SOAP has been used since many years ago but these days it's all about REST (Representational State Transfer). If you just go to any Job Portal and just click on 'REST API Automation Tester' or "REST API Testing Jobs, there are tons out there.

REST is kind of API which is actually efficiently being used by many companies like eBay, Amazon, Google. When such big companies are using definitely they have already transitioned from SOAP to REST. It means that you know REST is in high demand now. There are few advantages from the development perspective, REST supports a number of data formats. The most important ones are the way the request is sent and how we read the response. There is a specific format SOAP always uses XML format whereas REST uses JASON. Now, because JASON is being used JavaScript object notation, it's a form of representation. If you click on a JASON format on Google, objects will be inverted commas followed by the

parameters. So whatever, JSON is a way of our communication, a format which is being used for the information exchange such as request versus response.

REST API & RESTful web services explained

REST API, which stands for representational state transfer application programming interface. Some of the simplest questions that might come up when you're first starting to understand REST API is, what is being represented exactly, what is a state, and what is being transferred. So here in the article, let's look at all these words individually and learn what they all mean.

The word 'representational' means there is a transfer of representations of resources and the resources can be pretty much anything that can be named on the Internet like a list of users or a list of photos, comments, posts, articles, pages, videos, books, profiles, etc. To understand how exactly we get a representation of resources, you need to look at how everyone interacts with web pages, the client-server model, and the HTTP. Here's the HTTP protocol:

Everytime you type a URL in a browser, you're sending a request to the web server and the web server responds with a

resource. What's important to understand here is, everytime you type a URL in a browser or click on a webpage link, you're sending a request for a specific resource or resources from a web server. And the web server responses and delivers the request your resource to your browser as in a formal web page or whatever format that the resource is in. The second important point to understand is that web server doesn't actually deliver the resources. It doesn't send you the database that it has, but rather a representation of that resource in a format that is readable, for example, HTML or image. Think of actual resources, physical things located on a web server database or stored a web server hard drive, and representations of those resources as copies in either readable format for human beings like HTML or image or easy-to-work-with formats for programmers like JSON and XML. It's also helpful to think of the whole web as a bunch of resources. We always request the resource when clicking on a link or type in a URL, no wonder that URL stands for Uniform Resource Locator.

So what's about the word 'state'? Take an example, whenever I click a link, the application state changes and I am presented with another resource, which is a page with all the other content of that resource. And as I click through all these different pages, the application state keep changing from one state to the next. When the resource gets transferred from the web server, we send a request for the resource and we get a representation back and whatever formats the resources presented. This is what is meant by the word 'transfer'. However, it can also refer to the transfer of the application state when we click on the next link and get transferred to another page.

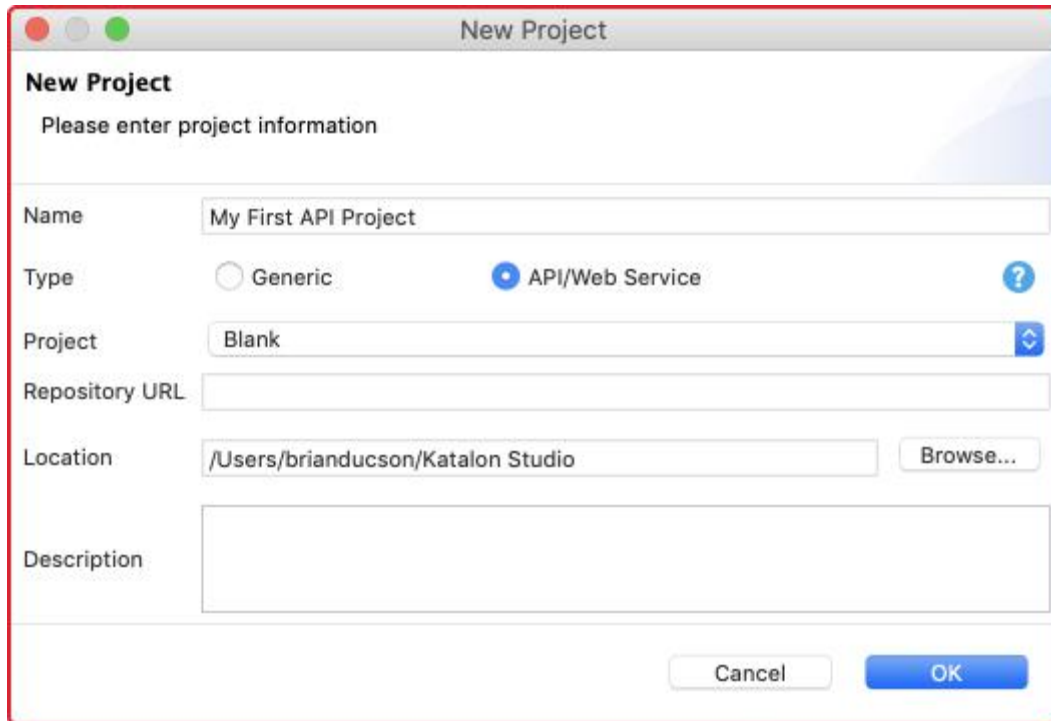Implementing REST API & RESTful web services testing using Katalon Studio

Installing and setting up Katalon Studio

Please refer to the guide for all the basic steps, from downloading to activating the build *__Installing and Setting up Katalon Studio__*.

Creating a new project and setting up API automation test

Step 1: Create a new project
Go to File -> New -> Project and enter a project name and its location to start a new project.

Once the project is confirmed to be created, we will see a folder structure in the Test Explorer This folder system is responsible to keep all the test resources and is also the place where we start our first API test.

Step 2: Create the first API test

Before creating our first API test, let's have a look at the format we use to set up a testing project.

Object Repository

- Object Repository is a place which stores all the Web service endpoints along with all information of Request method, URL, Header, Content, and Authentication.

- Web service test objects in Object Repository are integrated by a folder system for better management.

Test Cases

- Test Cases stores all test scenarios and is grouped by a folder system. Each test case includes a few steps illustrating a test scenario.
- We can execute a test case individually with a specified execution profile.

Test Suites

- Test Suites is the place where all test suites are stored. A test suite is a collection of test cases verifying a specific target.
- Test cases at the test suite level can be executed with a data-driven approach.
- Test reports are also generated at the test suite level

Test Suite Collection

- Test Suite Collection is a collection of Test Suites verifying a larger target.
- Test Suite at Test Suite Collection level has specific Test environments specified.
- Set-up API testing project
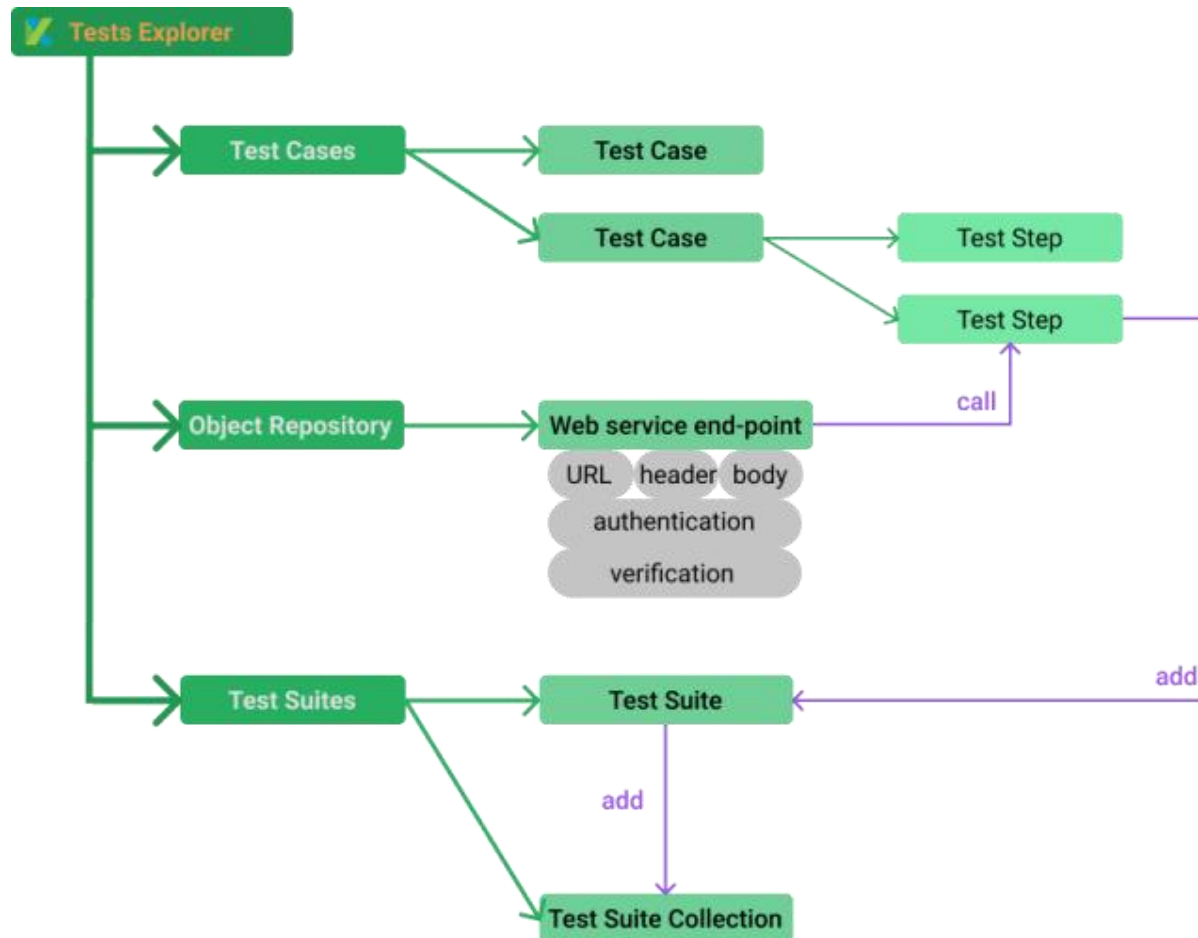
Step 3: Create a new RESTful endpoint at Object Repository
Object Repository -> New -> Web Service Request
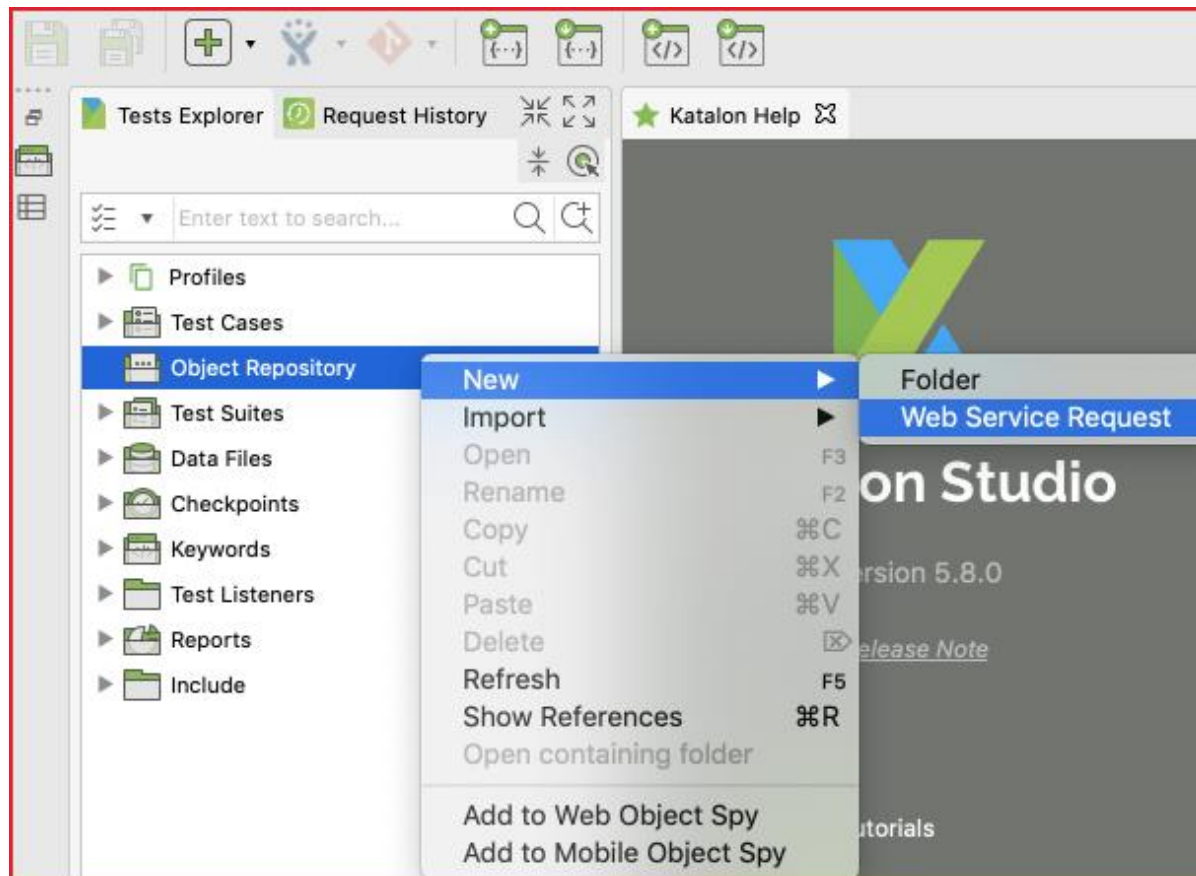
Katalon Studio stores Web service endpoint for testing at Object Repository, which is similar to Test Object in UI Test. At the Create New Web Service Request dialog, you can either choose to create a RESTful or a SOAP request.

Request type is a required field. You need to specify it exactly at this step. In contrast, URL is not required. You can set this value later in the next step.

Click OK, then we are ready to input more details to the first RESTful test.
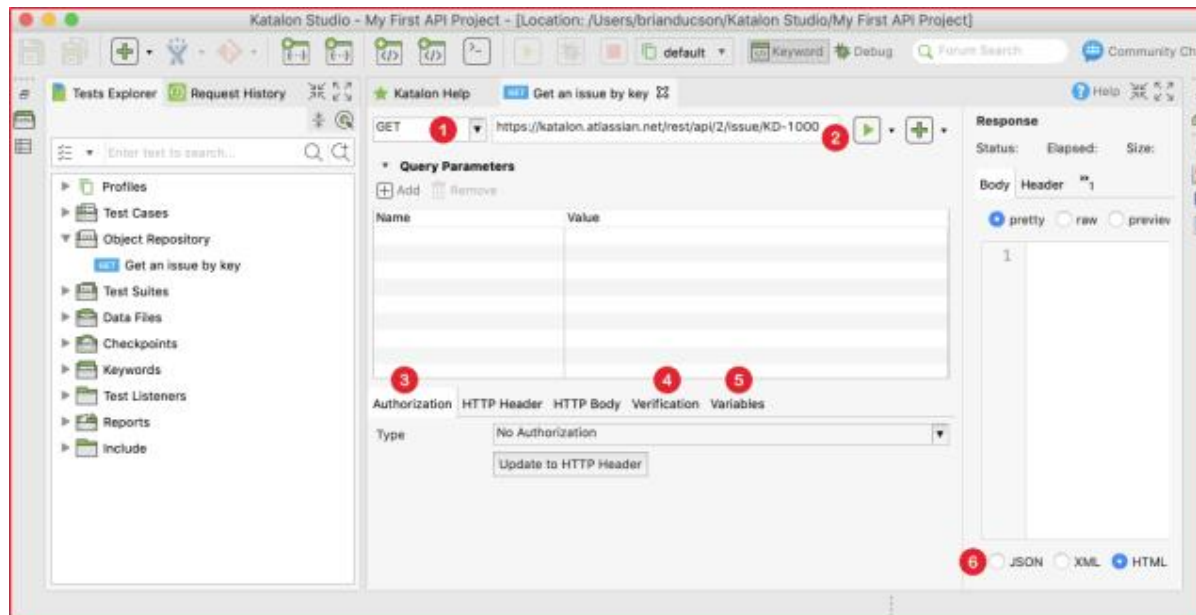
There are some important concepts needed to specify when testing a RESTful request:

(1) Request method

We can choose one of these following methods for your first request test: GET, POST, PUT, DELETE. The method needs to match with the URL to have a valid request. For instance, let's assume that our first test is a public API from Jira Cloud version. We should use the GET method to receive information on an existing ticket using its ID.

(2) Request URL

Along with the request method, request URL is used to tell the web server which API is utilized under test. Any mismatch between method and URL will lead to invalid request exception at runtime or wrong data response.

(3) Authorization

Authorization is an essential part of an API. It is used to get the correct data under permission (unless the data is public). Katalon Studio supports common authentication methods:

The basic method requires username and password. Don't forget to click Update to HTTP Header so that the authentication can be applied to HTTP Header.

(4) Verification

Verification is the place where you define the assertion to ensure the response will contain the expected information.
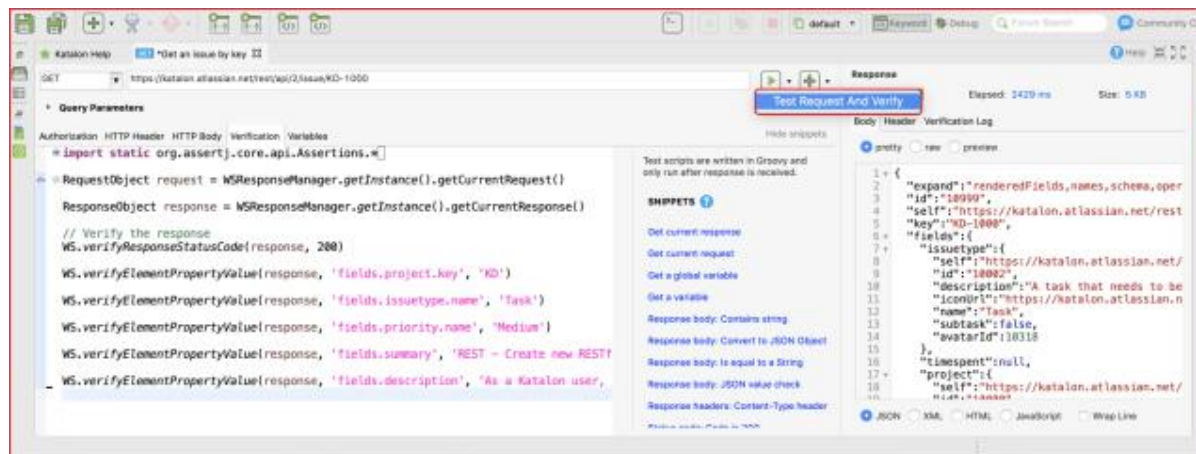
The verification tab of a request is similar to the Script tab of a test case. In other words, you can write custom scripts with built-in keywords or Groovy/Java scripts to verify the response data. Besides built-in keywords, Katalon Studio also supports built-in snippets, which help you to generate assertions with a single click. It is useful for testers who might find it difficult to deal with parsing and to assert with JSON data format.

The right panel of the request is the response displayed in nice format automatically and the result of verification at Verification Log. To include verification script when sending the request, you need to choose the Test Request and Verify option from the execution button.

The Verification script helps you have quick feedback of the request status rather than an actual test. You can add more assertions to the test case level in the next step.

(5) Variables

Variables make API testing more robust and dynamic with the data-driven approach. In Katalon Studio, every part of the request can be parameterized. In other words, dynamic data can be used for: URL, Authentication, HTTP Header, and HTTP Body to maximize the capability of data-driven testing. Following setup works the same with the above example:
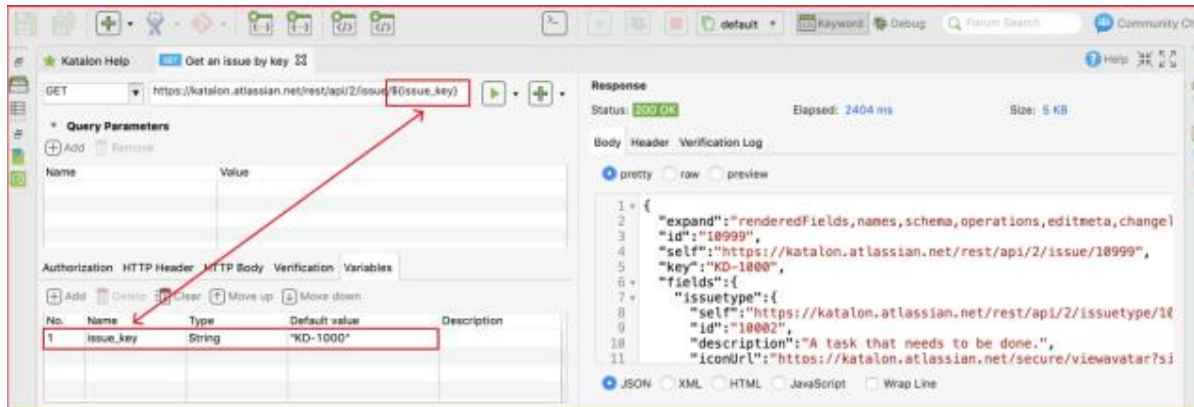
(6) Formatter

The response will be automatically displayed in a neat format: JSON, XML, HTML, and JavaScript. It is helpful for a quick view of the response status.

**Step 4:** Create a new test case with an existing request

While the request at Object Repository is helpful for fast testing, we can add the request verification at the test case level for better managing and reporting.

Step 5: Add an existing request to a test case

A request can be inserted into a test case with Web service built-in keywords. There are many keywords can be used to send the request, to verify the response, and to make the request as part of a bigger testing flow.
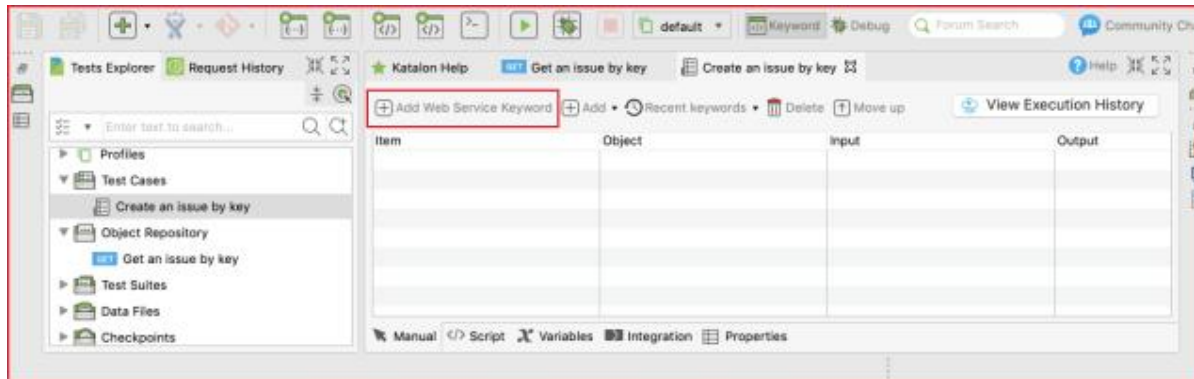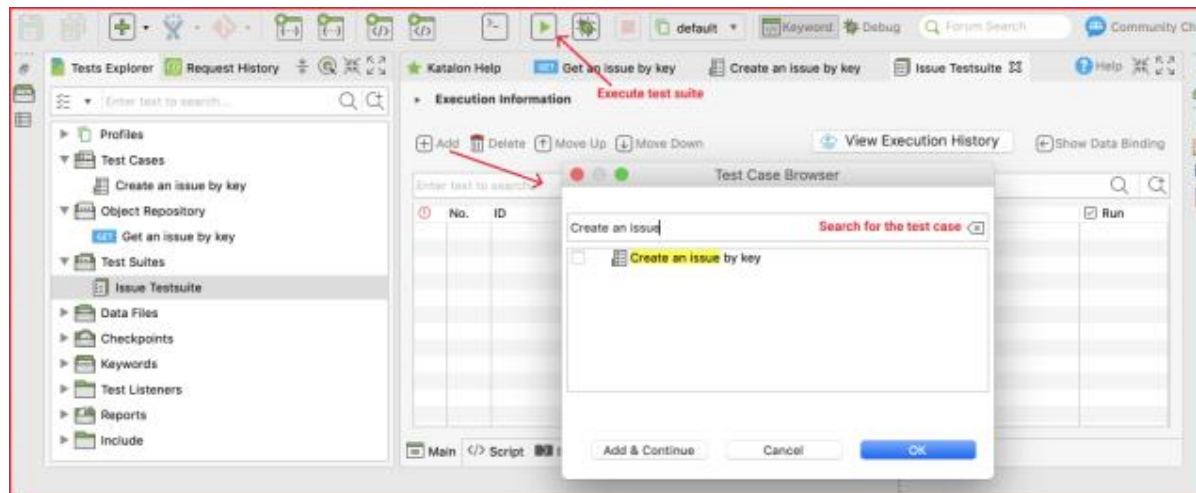
Following test case illustrates how we can call the request with verification steps from a test case:
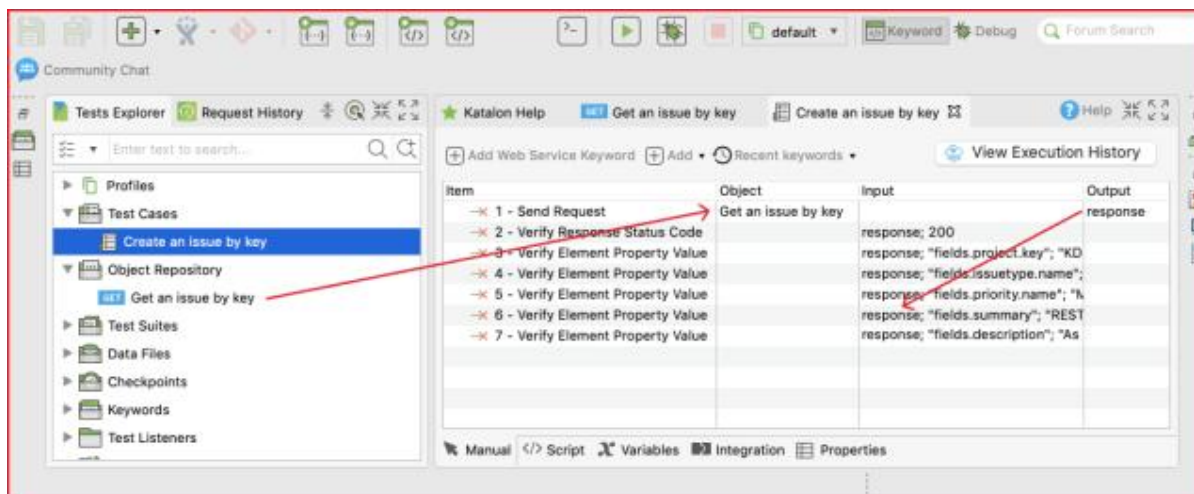
The test case can be executed as a normal test case in Katalon Studio. Each verification step can be viewed from the log.

Step 6: Add the test case to the test suite

A test case can be added to a test suite via either the drag-and-drop feature or the Add test case tool. Once the test case is added to the test suite, we can execute the whole test suite with the Run button (without selecting the browser as in UI testing).

https://testautomationresources.com/api-testing

[API Testing: What is it? | SmartBear](https://testautomationresources.com/api-testing)