



Achieve Peak Web Performance

An Introduction to API Performance Testing

WHITE PAPER

ApicaSystems.com 

Stochholm, New York, London, Santa Monica

It takes a developer to create an application and an API.
Does it require a developer to test them?

Applications are ubiquitous now and drive a lot of the value found in today's websites. A Google search for an address or a venue returns a page with an interactive map embedded. A click on a 'forgot password?' link makes an API signal the target application to send an email to the predefined address with a link to the password reset process. The flexibility and ease-of-use of APIs allows programmers to open up applications and make their processes available to other applications. The utility of APIs allows companies to sell the use of their API to other businesses. For companies that are very data-focused, APIs may become the primary product the company is banking on.

Application developers are an expensive resource. They need to be proficient in a number of programming languages and able to break down all the processes their application will perform into the individual steps it takes to perform them. They then have to code each of the processes, step by step, in the language and format that their application will use. After that, they must create the interfaces necessary to make the application available to the outside world.

Common Tests performed on API's

Verify if the API's does not return anything.

Verify if the API triggers some other event or calls another API. The Events output should be tracked and verified.

Verify if the API is updating any data structure.

When a person interacts with an application, transactions are initiated in the Graphical User Interface (GUI). This is a human-friendly interface that is easy to navigate and use. When an application interacts with another application, whether internally or externally facing, Application Program Interfaces (APIs) are used. The API specifies to the application what task is to be done and the information required to complete the task.

However, good development process dictates that the person or team that has written the API should not be the ones who test it. They can be available to help the test process, but fresh eyes and minds should be doing the testing. Does that mean that creating an application requires a large development staff so they can test each others' work? Not if the test team has the right set of tools.

What does API testing entail?

Testing APIs, like any other software test, starts small with functionality tests and progresses on through full capability, stress and security tests. Considerations to keep in mind when testing APIs include having a full understanding of what the API is supposed to do, creating test scenarios including the API call and interpreting the results.

The two most common API types are REST (Representational State Transfer) and SOAP (Simple Object Access Protocol). Both are equally capable of enabling applications to pass information back and forth, however:

SOAP APIs use their own specific protocol and they focus on exposing operations or application logic, as opposed to exposing actual data, as a service. An example of a SOAP API function is “switch category (User, oldCategory, newCategory)” – SOAP because it is a request to perform an operation, switching a category. SOAP APIs are generally written in Extensible Markup Language (XML), which is something of a sibling to HTML, a language in wide use on the internet. It is used to describe data.

Rest APIs are best used to handle Create, Read, Update and Delete (CRUD) operations on data and focus on accessing named resources through a single consistent interface. An example of a REST API function is “getUser (User)” – REST because it is a request for data, the User. REST APIs are generally written in JavaScript Object Notation (JSON), a lightweight format for structuring data that is easy for people to read and write and is easy for machines to parse and generate. It is based on a subset of JavaScript and is a text format that is language independent but uses conventions familiar to the ‘C’ family of languages.

So taking into account REST and SOAP, JSON and XML, requests and responses, you need to have a wide pool of knowledge to draw from to create a comprehensive test suite and to interpret the results. Having a tester at your disposal that is preprogrammed with that knowledge helps your QA team to quickly run tests that produce results with insights into where an API falls short or could be improved.

API Testing Resources

[API Testing Series Blog](#)

[API Testing Webinars](#)

[API Testing eBook](#)

Does this mean your QA staff must be well versed in XML and JSON to fully exercise the API?

When well-crafted test tools are used, it doesn't take a developer to write tests suites that fully exercise APIs. A well designed, intuitive GUI can steer a test engineer through the process of API building and test construction. A great test tool makes building thorough test suites measurably easier.

The challenge is that API testing tools need to support a minimum set of features to be usable. Sending a request to an application is easy, but displaying the response in a "decoded" way, related to the protocol being used, is not a feature found in simple limited products. After "decoding" the response the user needs to be able to extract some of the response data and store them as variables that will be used in a later request. This is a feature not usually available in simple products. Beyond that, there are also big differences in how easy the product can be to use.

What does an API test look like?

At minimum, a test should ensure that the API:

- Returns the expected responses for both expected and unexpected input
- Functionality works as expected even with multiple concurrent clients
- Is not vulnerable to common attacks

API tests should also be built to represent functional use. Unit testing APIs may prove that each request and response is functioning properly as an individual but your business relies on them all functioning together so that the response to a query for an inventory item can be coupled with a subsequent request to put that item in a shopping cart followed by a request to check out and a response with the correct price being reflected.





API tests should also address unexpected activity such as stalled sessions - where the application is expecting a response and there is none – or disconnects during a process, checking to see if the session is left open and vulnerable to being hijacked.

API tests should also look at third-party integrations. If your API has to work with another company's API, then testing only your API leaves it with a potential problem spot over which you have no control. Not performing API integration tests can end up, as we used to say, 'leaving landmines in your application'.

API tests should also track intermittent failures. A process may pass 99% of its tests and that 1% can be written off as 'expected deviation' or an acceptable error rate, but in the real world, where applications handle hundreds of thousands of transactions a day, that 1% error rate represents thousands of missed transactions and upset customers. User experience is everything on a website and it pays big dividends to go through logs to catch these failures before they cause untold damage to the company's reputation.

API tests should be scalable, taking all of the facets of testing and spreading them across hundreds or thousands of concurrent sessions. API testers have to be able to mimic the types of crowds that websites face on an hourly basis, mixing 'normal' traffic with traffic showing errors like long response times, malformed requests and unexpected content in responses

It should also not only be able to create tests but to examine results and flag tests that do not complete successfully. Once test suites have been created for the tester it must be able to archive those suites for test repeatability and regression testing. Test suites should be modifiable to meet the needs of a changing application environment and to keep up with newly-developed functionality.



Some target functionality to look for would be:

- The ability to send a number of API requests as part of the same test.
- The ability to test basic authentication
- The ability to include 'think time' as part of the request scenario to mimic real-world usage
- Validation of the server response across a variety of checkpoints including the HTTP Status, the Header information and the Body of the response (whether JSON or XML).
- Execution of API requests either sequentially or in parallel

Good API developers are hard to come by and development time is a significant cost. Those hours should be spent improving your application and your customers' experience. Your QA team can perform the lion's share of testing if they have tools that are up to the challenge.

7 Steps to API Testing in Apica LoadTest

Apica's new API Test feature is an entry-level solution. The user can create and run API tests without using ZebraTester. The ZebraTester script generated will be simple, while still providing value. The user can start to use ZebraTester for more advanced tests.

Step 1: Create New Test Page

There's an icon on the New Test page for API Tests.

Step 1: Choose which subscription to use

Choose which subscription you want to use for your test.
Custom Subscriptions

Default	Apica Administrators	OnDemand
Available Start: Unlimited End: Unlimited Max users: 1000000 Max duration: 4 hrs Max # of Tests: Unlimited Used in scheduled tests: See More <input checked="" type="checkbox"/>	Available Start: Unlimited End: Unlimited Max users: 1000000 Max duration: 4 hrs Max # of Tests: Unlimited Used in scheduled tests: See More <input type="checkbox"/>	Available Start: Unlimited End: Unlimited Max users: 5000 Max duration: 12 hrs Max # of Tests: Unlimited Used in scheduled tests: See More <input type="checkbox"/>


Step 2: Choose which script to use or create a new script


Use Existing Loadtest Script


☒ C# ☒ JS


Or

Create New Loadtest Script


URL


Selenium


ZebraTester


URL

Step 2: Set location from where the requests will be executed

Location

MyOnPremCluster

Step 3: Creating URL Request

The user then starts by entering a name for the request and the request URL for the API that is tested (example `api.zippopotam.us/us/90210`).

Optional the user can set authentication (basic authentication) before clicking "Send".

Request

Name

If the request is a HTTP POST then the request body can be entered into the Request Body field.

Request Body

Text JSON XML HTML

```
post code: "90210",
country: "United States",
country abbreviation: "US",
places: [
  {
    place name: "Beverly Hills",
    longitude: "-118.4065",
    state: "California",
    state abbreviation: "CA",
    latitude: "34.0901"
  }
]
}
```

Header Parameters can then be added to the request.

Header Parameters

Name	Value	
Name	Value	✗
Name	Value	✗
Name	Value	✗

Add Parameter

The last step is clicking on the Send button to send test to targeted tested application. The response from the request is presented with header and body.

The header is formatted in to parameters and the body is decoded as Text, JSON, XML or HTML.

Header Parameters

Name	Value	
Name	Value	✗
Name	Value	✗
Name	Value	✗

Add Parameter

Request Body

Text JSON XML HTML

```
{
  post code: "90210",
  country: "United States",
  country abbreviation: "US",
  places: [
    {
      place name: "Beverly Hills",
      longitude: "-118.4065",
      state: "California",
      state abbreviation: "CA",
      latitude: "34.0901"
    }
  ]
}
```

Step 4: Insert Assertions

The user can optionally add one or more an assertions that validates the response from the request.

Assertion

☐ JSON Body Equals ✗

In this example, the response from the server is JSON and the assertion is validating that the state value is equal to "California"

Step 5: Transaction Data Correlation

The user can set Parallel or Sequential execution of the requests and think-time for each request including variance.

API Test

Requests	
⊕ MyRequest_1	✗
⊕ MyRequest_2	✗
⊖ MyRequest_3	✗
Variable 1	
Variable 2	

Step 6: Test Run and Verification

All requests are listed in a three as an overview including variables. This part of the application is under construction.

Step 6: Test Run and verification

Before creating the test the user can save the API Test definition with a name for later use.

API Test Name

Save/Open API Test

SaveOpen

Step #7: Creating the API Test

Finally, the user clicks Create API Test and this creates a ZebraTester script. The user is then redirected back to the New Test page with the new ZebraTester script class file selected.

Create API Test

Create API Test

Summary - What are the hallmarks of a great API test device?

A truly great API tester will be easy enough to use that entry-level users can test with it. Making use of all your test personnel requires a test rig that is simple to operate but still provides value. At the same time, it should be flexible enough to be useful for intermediate and advanced test personnel to create complex test suites that can ferret out harder-to-create failures and test scenarios.

About Apica

Leading enterprises rely on the Apica Web Excellence Suite to test and monitor their mission critical business systems, APIs, web and mobile applications. Apica enables businesses to get detailed real-time performance, uptime and capacity insights, ensuring outstanding end user experience and optimized IT operations. Apica's suite – available as SaaS, on-premise and hybrid solutions – is trusted by 400+ leading brands globally. Apica has offices in Stockholm, New York, London and Santa Monica.

To learn more about Apica, visit apicasystems.com

Contact Us

North America: +1 (310) 776-7540
Nordics: +46 (0)8-400 273 27
UK/EMEA: +44 20 8396 4909
info@apicasystems.com