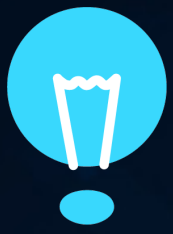
PARASOFT®
Automated Software Testing

How to Choose the Right API Testing Solution



Key Capabilities to make your Organization Successful in Today's Testing Environments

Ease-of-Use and Core Capabilities

- ✓ VISUAL AND SCRIPT-LESS FUNCTIONALITY
- ✓ A CUSTOM EXTENSIBILITY FRAMEWORK
- ✓ AUTOMATED ASSERTIONS AND VALIDATIONS
- ✓ DATA-DRIVEN TESTING
- ✓ TEST RE-USABILITY
- ✓ ABILITY TO RAPIDLY CREATE TESTS BEFORE A SERVICE IS AVAILABLE
- ✓ AUTHENTICATION

✓ VISUAL AND SCRIPT-LESS FUNCTIONALITY

Your API testing tool should not require you to have any experience writing code. A visual testing tool with an intuitive and scriptless user interface will empower a large body of testers (at a variety of experience levels) to use the tool productively. API testing can be overlooked when developers push it to QA, and QA focuses on manual testing. Having an API testing tool that is visual and script-less will enable testers to adopt this critical testing practice without having to spend lots of time on training and enablement.

✓ A CUSTOM EXTENSIBILITY FRAMEWORK

Should you need to write code to accomplish tasks such as generating a proprietary token or a unique identifier, your API testing tool should have the capability of using the scripts, but it should not be limited to a single language. Different testers and developers use different scripting languages depending upon the level of expertise and preference, so at a minimum, your API testing tool should support Java, Jython, JavaScript, and groovy.

Your API testing tool should also include a framework that allows you to extend the tool's capabilities, built-in transports, and protocols. This will enable you to support any transport or protocol that your organization is using, whether it is an industry standard or custom tailored to you. Examples of these might be DDS, plain socket, TCP, file-based messaging, etc.

✓ AUTOMATED ASSERTIONS AND VALIDATIONS

It is important for your API testing tool to help you define success criteria for response validation. This process should enable the tests to run automatically as a batch, validate messages, and eliminate the need for you to manually inspect your traffic. Your tool should be able to automatically compare responses to those expected, in bulk, as well as enable you to surgically validate a single element. Additionally, your tool should be able to validate the schema of a request or response payload to ensure the service is complying to it service definition.

✓ DATA-DRIVEN TESTING

For maximum flexibility when working with your API testing tool, it should be able to data-drive your test cases. This means that your framework should enable you to swap out static values in your API calls with dynamic values derived from data sources. It should support a wide range of data sources including CSV, Excel, JSON, as well as simple in-project tables.

You should be able to leverage dynamic data sources as well. An example of this would be connecting to a live database and pulling dynamic data at runtime, as well as a writable data source that can be updated on the fly. Additionally, you should be able to aggregate data sources together and have a mechanism for rapidly switching between them.

✓ TEST RE-USABILITY

Your API testing solution should be able to re-leverage any API test you have previously created. This gives you the ability to define scenarios such as web UI logins, complex authentication, or a repeated set of actions once and then bring them into subsequent test cases as a reference. This will help you maintain your library of test cases by providing a location where you can update critical paths and have child test suites inherit that information.

✓ ABILITY TO RAPIDLY CREATE TESTS BEFORE A SERVICE IS AVAILABLE

It's important to ensure that your API testing solution helps you rapidly create tests early in the software development process. This is critical because in order to adequately take advantage of the benefits of agile, you will need to focus on new features and functionality introduced into each Sprint. Often this takes the form of modifications to UIs and APIs. Being able to rapidly test this functionality before it's available ensures that you will be able to maximize your test coverage and ensure your critical use cases.

To do this your API testing solution must be able to consume service definitions such as Open API/Swagger, WSDL, RAML, as well as schema definitions.

✓ AUTHENTICATION

Your API testing solution also needs to work with authentication, encryption, and access control. A large number of your services will be deployed via an encrypted protocol such as SSL, as well as having a security policy such as Oauth, Basic auth, Kerberos, payload encryption, SAML, Signatures, etc.

It is important to be able to communicate using these authentication mechanisms, so your API tool must support all of the common standards. Additionally, you will need to validate your security is working properly, so your API testing tool should have a mechanism to ensure that the standards are implemented properly and work flawlessly.

Optimized Workflows

- ✓ TEST FLOW LOGIC
- ✓ AI-POWERED TEST CREATION
- ✓ TEST DATA MANAGEMENT / GENERATION
- ✓ EVENT MONITORING
- ✓ BDD SUPPORT (CUCUMBER)

✓ TEST FLOW LOGIC

Your API testing tool should have a mechanism for controlling test flow based on conditions. Not all test scenarios will execute in a linear fashion, so you may need to make automatic decisions at runtime that will affect how your test executes. An example of this might be ensuring that a response contains a specific element prior to moving to the next test step. Additionally, you may want to pause execution and poll a web service for a while to ensure a process has taken place, so your API testing solution must have the ability to analyze responses for key criteria and then use that information to control the rest of the test execution.

✓ AI-POWERED TEST CREATION

Your API testing solution must be able to work in an agile context. A key capability for success here is intelligent test creation, that brings significant efficiency gains. Your API testing tool should help you build test cases by understanding actions that you do repeatedly and learning how to do them for you. By leveraging artificial intelligence, your API testing solution will be able to monitor actions you're already testing manually with your applications, and automatically infer the relevant API calls from these actions. By streamlining the activities you do often and learning how to optimize the test creation process, you will be able to achieve a more successful API testing rollout.

✓ TEST DATA MANAGEMENT / GENERATION

Testers can spend a lot of time gathering adequate test data. Your API testing tool should support you in this activity by providing workflows for connecting to various data sources as well as generating test data itself. Your solution should have the ability to understand the types of data you require for given scenarios, and build on that test data with additional use cases so that your test cases can be as flexible as possible.

✓ EVENT MONITORING

To enable end-to-end testing, your API testing solution must be able to monitor events as they flow through your system, so you can validate inputs and expected outputs, and understand how transactions transform as they move through your application. With multi-step validation by plugging into your application internals via JMS messaging, database monitoring, etc., your solution will be able to provide greater levels of test coverage.

✓ BDD SUPPORT (CUCUMBER)

To support BDD, your API testing solution should give you the foundation for Cucumber step definitions that your QA team needs to build the required test steps for business analysts to leverage in their BDD. This will broaden the usage of your solution across key stakeholders and ensure that your testing is done at the earliest stage possible.

Supported Technologies

- ✓ REST API TESTING
- ✓ SOAP API TESTING
- ✓ MQ / JMS TESTING
- ✓ IOT AND MICROSERVICE TESTING
- ✓ DATABASE TESTING
- ✓ WEB-BASED TESTING
- ✓ PERFORMANCE TESTING
- ✓ TESTING NON-STANDARD MESSAGE FORMATS

✓ REST API TESTING

Your testing tool must be able to interface with Representational State Transfer protocol (REST) APIs. This includes support for service definitions such as Open API OAS3, Swagger, or RAML. Your tool must be able to send URL, Method, Path, Parameter, Query, and JSON payload information, as well as Headers, Mime Types, Attachments, and so on. Additionally, it needs to be able to consume and validate request and response for simple regression or schema validation.

✓ SOAP API TESTING

Simple Object Access Protocol (SOAP) is still extremely relevant in most applications, so your API testing solution must be able to interface with SOAP APIs. This includes support for service definitions such as WSDL and schema XSD. It must be able to send SOAP-compliant requests that include SOAP action, attachments, WS policy, and relevant SOAP headers. Additionally, it needs to be able to consume and validate SOAP XML responses for simple regression or schema validation.

✓ MQ / JMS TESTING

Your API testing solution must support queuing technology such as MQ and JMS, so you can simulate various patterns including point-to-point and publish/subscribe. This will allow you to do complete end-to-end testing, and validate message systems that leverage these technologies.

✓ IOT AND MICROSERVE TESTING

IoT and microservices are bringing a host of new testing challenges. While still largely REST/JSON based, the future of IoT and Microservices will have interfaces deployed on nonstandard protocol (i.e. Websockets, MQTT, AMQP/Rabbit MQ, Kafka, and Protocol Buffers). Your API testing tool must be future-proof and have the ability to communicate via these new protocols, so that as they begin to be implemented at your organization, you are ready to test them.

A second testing need is the ability to isolate individual Microservices that are deployed via an asynchronous protocol. Your API testing tool must be able to emulate not only a provider to the queue or topic but also have the capability of consuming messages off of the topics, so you can isolate and test individual services.

✓ DATABASE TESTING

Your API testing tool must have the ability to communicate with databases. This will give you the ability to validate the contents of the database as you are interacting with your system via the APIs. This will enable end-to-end testing and ensure that your messages traverse through your system properly. Additionally, by being able to connect with your databases you will be able to pull data out of the database prior to API test execution. This will make your API test scenarios more dynamic by being able to use the most relevant and up-to-date data.

✓ WEB-BASED TESTING

Your API testing tool must have an integrated capability for testing your web UIs. This does not need to be your only web UI testing solution (i.e. Selenium), but should complement your API testing so that you can test across multiple interfaces exactly the same way your customers will interface across your business. An example of this would be using your web UI capability to log into an application and begin a transaction. You could then leverage your APIs to hit individual components to validate the transaction has seated into your application properly, and then finally you would be able to pull from the database to ensure the relevant information was stored appropriately.

✓ PERFORMANCE TESTING

Your API testing solution should allow you to shift left performance testing by leveraging the API tests you have been creating for individual component, smoke, and regression tests as a part of your nonfunctional load and performance testing strategy.

Your API testing solution should allow you to generate load via a built-in mechanism for controlling the number of transactions or number of virtual users. By having this performance testing capability integrated into your API testing solution, you will be able to conduct more performance tests earlier in the software delivery lifecycle.

✓ SUPPORT FOR NON-STANDARD MESSAGE FORMATS

Your API testing solution must be able to communicate over non-standard message formats and protocols, such as mainframe (copybook), fixed length messaging, or Electronic Data Interchange (EDI, which is often found in the exchange of computer to computer business documents), as well as industry standard message formats and protocols such as FIX and SWIFT. It is also important for your API testing solution to be extensible, to include additional proprietary message formats and protocols that your business demands. The extension of these protocols should be scriptless and modular.

Automation

- ✓ CI INTEGRATION
- ✓ BUILD SYSTEM PLUGINS
- ✓ COMMAND-LINE EXECUTION
- ✓ OPEN APIS FOR DEVOPS INTEGRATION

✓ CI INTEGRATION

Your API testing tool should have the capability to integrate into your existing CI process, so you can execute test cases via a command line interface or a series of open REST APIs, and select which type of test cases you want to execute with a specific configuration. Additionally, you should be able to retrieve the results and process them back into your CI pipeline to make an automated go or no-go decision in your deployment activities.

✓ BUILD SYSTEM PLUGINS

Your API testing solution should have built-in plugins for common CI systems such as Jenkins, Microsoft VSTS, Atlassian bamboo, and Jet Brains Team City. There are other build systems available, so your API testing solution should have an extensibility platform to allow you to build connectors for all future CI systems.

✓ COMMAND-LINE EXECUTION

Your API testing solution should have the ability to execute your API tests in batch via a command line interface. The command line interface should be easy to use so that you can set up a variety of batch execution scenarios across multiple interfaces without having a heavy reliance on scripting.

Your command line interface should be dynamic so that you can swap out variables, data sources, and test environments by simply modifying flags. This will ensure the command line interface is readily used by your organization.

✓ OPEN APIS FOR DEVOPS INTEGRATION

Your API testing solution should have open APIs that allow you to programmatically generate, configure, and execute test cases. This will allow you to set up a client/server configuration for your DevOps platform and your API testing platform. A series of open APIs will allow you to set up a scalable infrastructure and reduce overall licensing costs by programmatically making calls to the API testing server from multiple parts of your organization as needed to execute the proper test cases.

Management and Maintenance

- ✓ INTEGRATION WITH REQUIREMENTS MANAGEMENT SYSTEMS
- ✓ BASIC AND ADVANCED REPORTING
- ✓ TEST ORCHESTRATION
- ✓ A PROCESS FOR MANAGING CHANGE
- ✓ ON-PREMISE AND BROWSER-BASED ACCESS

✓ INTEGRATION WITH REQUIREMENTS MANAGEMENT SYSTEMS

Your API testing solution should integrate with requirements management systems such as ALM, Bugzilla, and Jira. You should be able to identify to which requirements specific API tests are associated, and have a mechanism to understand how the results of that API test affect the individual requirements.

✓ BASIC AND ADVANCED REPORTING

Your API testing solution should have a rich reporting framework that allows you to understand individual API test results, as well as the entire project health. Your reporting framework should allow you to generate shareable documents that can be customized to individual stakeholders needs, and be machine-readable so they can be processed by your build systems and provide a simplified integration into your CI pipeline.

✓ TEST ORCHESTRATION

Your API testing solution should provide the ability to bundle together API testing scenarios for execution in multiple environments. This will enable you to create a test run that is customized to an application in an environment. Overall this will make your test cases easier to manage by allowing you to provide dynamic information into the scenarios such as data sources, environment endpoints, user access control, etc. while reusing the same test cases.

✓ A PROCESS FOR MANAGING CHANGE

One of the most critical capabilities of your API testing solution is a change-management process.

First and foremost, your API testing solution must natively integrate with your source control system. This will allow you to maintain several versions of your API tests, for forwards and backwards compatibility.

Additionally, your API testing solution must be able to understand the various versions of your APIs via service definitions, or schemas, and should your API service definitions change, your library of test cases should have an automated re-factoring process. This should be automatic in the case of small changes, or have an easy-to-understand process when dealing with larger changes.

✓ ON-PREMISE & BROWSER-BASED ACCESS

Your API testing solution must be broadly available. This means that you should be able to access your solution either on Prem or through a cloud provider, and your API testing solution must have a desktop application as well as a browser-based application. This flexible deployment will help broaden access and adoption of the solution, as well as help teams collaborate and reuse artifacts. As a part of enabling browser-based access, your API testing solution should have the proper user access controls in place so that your company's information is secure.

Looking for a tool that checks all of these boxes?

Choosing the right API testing solution for your organization can be a daunting challenge when you take into account all of the features and capabilities to consider. To get a solution that has all of these capabilities and more, you can check out Parasoft SOAtest. Learn more and get a free trial of the comprehensive functional test automation solution at https://software.parasoft.com/soatest.

Parasoft SOAtest

THE INDUSTRY-LEADING API TESTING SOLUTION

API, mobile, web UI, and database testing that's easy to use, even for beginners.

Parasoft SOAtest brings artificial intelligence and machine learning to automated functional testing, to help users test applications with multiple interfaces (i.e. mobile, web, API, and database). Its automated API testing mitigates the cost of re-work by proactively adjusting your library of tests as services change.

Parasoft SOAtest efficiently transforms test artifacts into security and performance tests, to increase re-usability and save time, all while building a foundation of automated tests that can be executed as part of Continuous Integration and DevOps pipelines.

LEARN MORE

ABOUT PARASOFT

From development to QA, Parasoft's technologies reduce the time, effort, and cost of delivering secure, reliable, and compliant software, by integrating static and runtime analysis; unit, functional, and API testing; and service virtualization. Powerful reporting and analytics help users quickly pinpoint areas of risky code and understand how new code changes affect their software quality, and groundbreaking technologies that add artificial intelligence and machine learning to software testing make it easier for organizations to adopt and scale an efficient software testing practice across development and testing teams.

www.parasoft.com

Parasoft Headquarters:
+1-626-256-3680

Parasoft EMEA:
+31-70-3922000

Parasoft APAC:
+65-6338-3628

PARASOFT®
Automated Software Testing