INCREASING SOFTWARE DEVELOPMENT VELOCITY –

A GUIDE TO USING API TEST AUTOMATION TO MAKE YOUR SOFTWARE FASTER, BETTER AND MORE SECURE

optimus
information

TABLE OF CONTENTS

I INTRODUCTION

If you're developing software then you know that APIs (Application Programming Interfaces) are essential to your success. APIs are the digital framework that allow all of us to enjoy seamless connectivity between discrete services and software packages. Without APIs, we would need to recreate monolithic software stacks for each unique deployment.

Neither fun nor agile. However, many organizations are still doing just that. Faced with supporting older legacy applications, or even developing new software using older methods, some enterprises are not fully capitalizing on the benefits that APIs offer.

In this guide, we will begin by explaining the two main benefits of using APIs, which are as follows:

1. How your software will level the playing field against your competitors;
2. How it will increase your software adoption.

Then we will dive deeper into some guidelines to help you implement best practices around API development.

THE TEST AUTOMATION ADVANTAGE

Deciding to use APIs only solves half of the problem. The next step is to automate the API testing process.

Why?

API test automation will introduce the highest standards in your software development. It will ensure that your APIs are not leaving security vulnerabilities and that your APIs are working as intended.

The suggestions in this guide will introduce you to best-in-class standards that are designed to reduce future software maintenance costs by building a reusable library of API calls, and describe several testing technologies that can be tailored to your company's needs.

Let's get started.

CHAPTER 1

| THE PROS AND CONS OF APIs

In this section, we want to explore both the advantages and the vulnerabilities of APIs. Truly understanding why you are going the API route will enable you to use APIs to your best advantage.

LEVEL THE PLAYING FIELD

The first benefit of APIs is that they provide a high level of standardized instructions to third parties that want to access any organization's applications, middleware and services.

For example, imagine there are no longer any APIs for web-based data-gathering companies like Google, Facebook and Twitter. Your software will now be required to develop the application, logic and presentation layer that mimic the functions of the APIs instead of having your software do a simple API call. This increases your company's software costs and maintenance.

INCREASE SOFTWARE ADOPTION

The second benefit of having APIs is that it will increase your software adoption.

Companies have financial constraints and/or limited man-hours dedicated to a specific software project. The saying "Why would anyone reinvent the wheel?" is applicable here. Wouldn't it be better to spend your limited resources on building features and additional services if the foundation is provided through an API call?

If your software has a sound foundation, then adding APIs is a vital step toward encouraging widespread adoption. By providing a programmatically efficient method for accessing your software features, other companies will rely on your software to innovate and create additional services.

Real-estate, mapping and social media mash-ups on the web are well known examples of how new aggregated services are built over others' APIs. Once companies are dependent on the foundation you created, you'll be perfectly positioned to democratize your software across specific target markets.

If your software has a sound foundation, then adding APIs is a vital step toward encouraging widespread adoption.

DRAWBACKS OF APIS

APIs are extremely beneficial to your software, but there are certain drawbacks. The main drawbacks are security implications, maintenance commitments and the testing requirements associated with each.

SECURITY IMPLICATIONS AND TESTING

Security tests are critical due to APIs' programmatic nature because APIs expose your software code to the outside world. This security exposure will add to development, maintenance and testing commitments that your company may not be willing to make. This exposure makes APIs vulnerable to malicious exploits that can affect the confidentiality or integrity of data sent by third-party software that relies on the now-vulnerable APIs.

For example, a bug in an API may increase the risk of unauthorized program data access – and, by proxy, access to user data. This security risk is much higher than that associated with having a bug in the app's GUI. Due to these concerns and how APIs give access to certain parts of your code, implementing an API requires a comprehensive test plan that includes unique aspects of API testing. Another benefit of API-specific testing is that it reduces the level of difficulty in tracking down root causes after deployment due to error propagation.

Exposure makes APIs vulnerable to malicious exploits that can affect the confidentiality or integrity of data sent by third-party software.

MAINTENANCE COMMITMENTS AND TESTING

In our experience, many companies approach API testing in a manner similar to GUI testing. They often think that API testing is easier than GUI testing because interactions are programmatically performed automatically and without screen interactions.

It is true that API and GUI share one fundamental characteristic in common from a testing standpoint: the code runs as the developer intended and the edge cases are accounted for. Each test case is tailored, with rules and procedures that test against a specific function written by the developer to see if the function is working as intended and that there are no undocumented defects.

But here is where we see mistakes made.

APIs introduce an external variable compared to GUI. This means that API testing is more difficult because it requires white-box testing to ensure internal mechanisms are functioning as expected. You really need to shake out external API behaviours in the test lab, and this is something that is often overlooked in GUI testing requirements.

These tests are important because the consequences of API failures or misbehaviours from a defective API call results in security vulnerabilities and hidden financial costs.

In summary, testing APIs and GUIs is the only way to reveal these unintended and undocumented defects - but API testing does require more man-hours, increasing your financial cost.

However, an increased financial cost in testing is usually preferable to releasing defective APIs that will make people question your software's API reliability and security. Once the reputation of your software is questioned, the adoption of your software through third-party software will be hindered. This may cause third-party software vendors to seek other solutions, often provided by your competitors.

CHAPTER 2

| API TESTING AND WEB SERVICES

Typically, testing APIs that communicate over networks involves testing functionality at the high level; e.g. does the API call perform the expected action, and is it at the lower levels of the messaging protocol? This aspect of testing crosses the threshold between black-box and white-box testing because the tester must ensure the messages being passed back and forth are correctly formed. That's not always the case even if they meet functional requirements.

Many testing tools and frameworks have the capability to extract message details, or the tester may write code, often in Java, to accomplish this. Messages may be represented in JSON, XML, GraphQL or other possible formats that require advanced programming skills from testers. Usually, either the testing tool or the tester's code performs string extractions and reformatting of the message contents to verify correct results.

AUTOMATING THE API TESTING PROCESS

While specific steps in automating your API tests depend on the type of API you are testing and the testing tools and framework you are employing, in our experience, we recommend that you follow the same general steps used to build your test cases, case collections and scripts.

1. Test Preparation

a) You may choose a test tool, framework or test manager specific to your API's protocol, such as REST or JSON, but we recommend testing tools able to handle APIs that support multiple protocols.

b) Be sure to obtain the most recent API documentation and up-to-date documentation on supported API messaging protocols.

c) Create a test plan with case templates used for every type of test, such as functionality, security, usability, performance and so on. We recommend that each test case should specifically detail what is being tested, how it will be tested (including API endpoints), expected results, pass/fail criteria plus a link to specific API requirements.

2. Build the Tests

a) Build a reusable library of API calls for each web protocol required. For instance, you will likely need a reusable API login call plus calls to retrieve key information about the API's state to use in evaluating test results.

b) On top of your test call library, build test case code directly linking back to specific test cases previously created.

c) Build collections of test cases that are called from a single script. We recommend that each script pertain to a specific API feature or a particular area of testing such that it can be used to prioritize the testing of recent changes.

d) Check the happy paths first; for example, that the basic request/response functions correctly, that all possible API input combinations are covered, plus parameter combinations and input values. Then move on to testing for graceful failure and load testing.

SAVE HEADACHES BY USING A TEST MANAGER

The steps outlined above are made infinitely easier by using a test manager, such as TestRail, to build your call library, specific test cases, test code and case collections. Many test managers also support the ability to create test configurations for each testing environment, such as development, QA, integration and production testing.

API TESTING DOS AND DON'TS

What You Must Do

- Plan ahead for test suite maintenance. We frequently see test plans go awry because the stakeholders underestimated both the amount of testing and the amount of effort required to maintain the testing environment.
- Focus automation on stable code when possible, as this reduces test case/script thrashing. This "do" is harder to accomplish in agile environments due to their inherent rapid pace.
- Be sure all API endpoints are tested, especially with regard to security, from the most remote endpoint in the cloud to any local endpoints.
- Before any test run, ensure that the system is "plugged in." That is, make sure all dependent web services are up and running.
- Make your testing more efficient by storing persistent data, such as login tokens, locally.

What You Must Not Do

- Never perform tests with HTTP calls. Use only HTTPS.
- Do not create brittle tests that must be changed frequently, as this adds to maintenance costs.
- Do not create monolithic test scripts. These are harder to maintain and slow down testing.
- Do not treat all test cases and scripts as equal. Classify and prioritize them by their value according to how quickly they are able to pick up defects in the API.
- Do not allow API documentation to lag when defects in the specs are found or when API changes are made due to shifting requirements or as a result of testing.

CHAPTER 3

API TESTING TOOLS AND FRAMEWORKS

The best API testing tools and frameworks help you build your test cases and test code in a manner that ensures tests are reusable, maintainable and that they promote stability in the testing environment. These are vital because API testing, unlike GUI testing, is more challenging, in that test writers need white-box knowledge of the API and a reasonable level of programming skills.

TRICENTIS

The Tricentis web service API testing tool easily integrates into continuous testing environments with support for test case design, optimization and test data management. Its model-based test automation approach reduces maintenance while increasing re-usability. It supports a large set of API protocols: HTTP/HTTPS, SOAP, REST, JMS, AMQP, Rabbit MQ, TIBCO EMS, IBM MQ and NET TCP.

PARASOFT SOATEST

Parasoft's design and features are tailored for true end-to-end, GUI-less SOAP API testing. It supports complex multi-component scenarios involving databases, message layers and enterprise service buses, and is compatible with over 120 message types. Many tests can be created without coding skills. It is adept at uncovering Open Web Application Security Project vulnerabilities during security testing.

SOAPUI

SoapUI NG Pro was built specifically for SOAP and REST API testing, plus it includes support for JSON, XML, JMS and IoT testing. Its unique point-n-click/drag-n-drop interface lets you create tests quickly and simplifies examining JSON and XML data. Test data may be loaded from Excel, databases or files. It quickly turns functional tests into load or security tests.

JMETER

Jmeter is free and open-source with powerful features such as GUI-created tests, data and parameter import from CSV files, an easy-to-use extractor for validating API responses with JSON, XML or regular expressions, plus plug-ins for test execution during code builds. This is one of the tools that we have standardized on at Optimus and we consider it best-in-class.

CHAPTER 4

ROLE OF THE CLOUD IN APIs AND API TESTING

Increasingly, web service APIs are executed on virtualized hardware and software stacks. In general, cloud-based API testing is much the same as for non-cloud APIs, but there are some distinct differences and challenges.

For instance, we find that shared cloud hardware, software and network resources are typically not under our control and may impact performance testing in opaque ways. Overcoming this facet of API cloud testing typically requires continuous monitoring to average out the differences.

Increasingly, we are recommending Microsoft Azure Platform as a service (PaaS) for organizations looking to build software in the cloud. It is a complete development and deployment environment with resources that enable you to deliver everything from simple cloud-based apps to sophisticated, cloud-enabled enterprise applications.

With Azure PaaS, you can expect a great infrastructure environment —servers, storage and networking—but also middleware, development tools, business intelligence (BI) services, database management systems and more, as PaaS is designed to support the complete web application lifecycle: building, testing, deploying, managing and updating. For example, SQL is PaaS and as such reduces the need to maintain, patch and license a VM. It can be spun up or spun down quickly, depending on project demands.

As an organization, Optimus developed the "Azure Test Harness" to help companies that want to automate API testing; after exploring several cloud options, we made the decision to run the Test Harness in Azure.

We take special care to account for security test cases that include the possibilities of data or access controls being exposed to other applications sharing the same cloud infrastructure, even if the intrusion were accidental.

Additionally, since an API package may depend on other components within an existing software stack or through third-party apps and APIs, we find it prudent to test the interfaces with which the API interacts as much as possible to ensure they meet functional, performance and security requirements.

ENGAGE KNOWLEDGEABLE SPECIALISTS TO INCREASE CHANCES OF SUCCESS

As you have seen, API testing is one of the more complex tasks a tester can sign up for. That is due to four main factors:

1. The need for white-box testing, which requires in-depth knowledge of web protocols and message contents.
2. A more rigorous focus on testing for security issues than would be present in a stand-alone client app.
3. A testing ecosystem, especially for cloud-based APIs, that is dependent on hardware and software resource sharing, which complicates load and performance testing.
4. API interactions with 3rd-party software components that extend the sphere of testing beyond just the API itself.

BUILD YOUR API TEST COMPETENCY – WORK WITH AN EXPERT

APIs improve decoupling of software components, making these components available to other projects and to innovate use cases. There are many examples of new web services that went nowhere because of a lack of an API. APIs lead to more usable, reliable and maintainable software components that are better able to make profitable use of cloud resources.

APIs bring strong advantages to your software development, and effective API testing efforts - despite the additional complexities and issues inherent in API testing - is simply one more area of expertise you need to master today to truly compete.

By mastering API testing competence, you will have put in place a more robust testing environment than your competitors, who are probably still relying on GUI or browser testing. You will find that this discipline will extend to other software projects and is especially useful for fast-paced CD/CI/CT environments.

For companies that want to get started, Optimus makes it easy. We can act as a guide and help you develop a plan for a robust, automated test sequence. We will give you the Azure test harness licence for free and show you how to use it. And if you need more help, we can work with your team on an ongoing basis, ensuring that your software benefits from our experienced team of test experts.

CONTACT US TODAY

for a free virtual assessment and to learn how to get our Azure Test Harness for free.

Optimus Information Inc.
510-900 Howe Street. Vancouver, BC, Canada.
Phone: +1 604-736-4600 | Email: info@optimusinfo.com