# API Testing Micro-Credential

## Syllabus


# Table of Contents

## API Testing



## References

# General Information

## STUDY TIME – 190 MINS.

### KEYWORDS

API, database, integrated connections, endpoints, API mock, Token, data exchange, messaging, data connectivity, data transfer, request/response, get, put, post, open API, composite API, partner API, internal API, JWT, token, token authority, OWASP, Postman, GET, POST, PUT, request, response, data validation, Developer Tools, error testing, test automation, data format, JSON, XML, grey-box testing, Security tokens, server, tools, test maintenance

# LEARNING OBJECTIVES FOR API TESTING

### Introduction

### API Test Strategy & Planning
(K2) Explain the importance of test planning or strategy for API testing

### The Software Tester's Role
(K1) Recall the software tester's typical role in API testing

### Benefits - Why do API Testing?
(K1) Recall the benefits of API Testing and why it is needed

### API Types & Functions
(K1) Recall the four API types and functions
(K2) Understand the different types and functions of an API

### Definition of an Endpoint
(K1) Recall the definition of an API endpoint

### Principles of Data Exchange & Messaging
(K2) Explain the concept of data exchange and messaging as it applies to an API

### API Security
(K1) Recall the importance of API security using tokens

### Error Messaging
(K1) Recall the 5 error status codes levels
(K2) Explain the parts of the API response that indicates the status code

### Functional Testing
(K1) Recall the definitions of GET, POST, and PUT
(K2) Understand how the response tests for data validation
(K3) Apply understanding to evaluating a test response within the response code
(K3) Apply understanding to evaluating a test response using Developer tools in a browser

### Error Testing
(K1) Recall the importance of error testing to ensure all status codes are triggered

### API Test Automation
(K1) Recall the existence of API test automation options available

### Server Access & Security Tokens
(K2) Explain the need and purpose for server access to test API both for authentication and functional testing

### Tools
(K1) Recall the availability of tool options for API testing

### Test Maintenance
(K1) Recall the importance of test maintenance and its impact on API tests

# Introduction

The acronym API stands for Application Programming Interface. APIs are programming interfaces that connect applications to each other, a database, or a messaging system.

API test development and execution is the most important grey-box functional testing a tester performs if the application(s) shares data, messaging, and relies on the input and output of various API endpoints. API testing is performed to ensure API endpoints and data sharing functions work as expected. API testing ensures connecting data feeds are sending the data expected as well as how, when, and where expected. API testing guards against data corruption in transit or sudden failures in database or endpoint connectivity. API testing ensures your application functions perform as intended for applications designed for web or mobile environments.

Testing APIs is an essential skill for the modern-day software tester. Mobile and web applications use data continuously and share it amongst applications making the software tester role more complex. Testing the API connections for an application is a critical skill in today's professional software tester job market.

# What will you learn?

## API Test Strategy & Planning

Start API testing efforts by developing an API test strategy or test plan. The plan doesn't have to be a long, or formal document; it can be as simple as a list. For example, for API testing the business needs to determine and define the following:

- Who is developing the tests?
- Who is executing the tests?
- How often are the tests executed?
- Where are the tests being executed, or what server is planned for testing?
- How are the tests created? For example, determine if the business plans on using a tool, developing a custom tool, or using browser Dev Tools.
- Define an API test priority list for all API's used by the application(s) under test.
- Define the main types of testing performed during API testing:
  - Access security
  - Endpoint security
  - Data security
  - Error Messaging
- Functional Testing using request/response verification
- Get security access established for those who develop and execute the tests, along with the appropriate user rights.
- Train testers on how to access the APIs directly, through tools, or through the application for testing.
- Determine if the tester will be testing partner APIs directly or by developing API mocks for testing.

A testing strategy for APIs is necessary because it helps to ensure the critical items needed for a particular API testing effort have been considered. There's more to API Testing than knowing the endpoint is functional. It's complex, which is a good reason to test as thoroughly as possible on a regular basis.

## The Software Tester's Role

API testing is another type of testing performed by a person in the software tester role within the development team or as part of an independent test team. Frequently, the software tester works with a developer to get the endpoint URLs or other API documentation to enhance the quality of the tests. Developers are an excellent reference point for how the API works, what part of the application functionality it impacts, as well as the data passed or returned by the API. In addition, the developer is often the provider of API access keys. The software tester is tasked with finding defects and failures in the API by testing the application UI, and testing the actual API directly.

## Benefits

**Why execute API Testing?**

The simple reason for performing API testing is that without API's most modern mobile and web applications cannot function. APIs are the bridge to functionality delivered by services and microservices.

For example, if the application consumes or shares data, messages outbound, or saves to a database, API testing is needed.

One benefit of API testing is it provides test coverage to ensure the application's API's are communicating and saving as expected. A common goal of API test coverage is to test each API and each functional area of the application impacted by the API.

Another critical benefit of API testing is that it provides a layer of security testing by testing the authentication, authorization and encryption of data. Messaging and data can be intercepted or re-routed, which can result in a data breach. Testing the security of API endpoints is essential to stop unauthorized users from accessing application data in transit.

Another benefit of API testing is the protection against defects arising from unexpected code changes made by connecting partners or database code updates. Unfortunately, such unexpected changes often occur and have their impact in the production (live) environment. For this reason, API testing may need to be conducted as a monitoring activity for applications in live use.

Finally, the risk of the disruption of an application's functional workflow is reduced when API testing occurs regularly on either a live or test server. API testing ensures consistent functional performance of the application for customers on the live server.

# Fundamentals

## API Functional Types & Protocols

There are several types of API's used in software development. Each has a specific use depending on the function the API performs.

- **Open** – These are open source API's accessible by using a simple HTTP protocol. They are public API's that are not secured, so nearly anyone can connect to their endpoints.

- **Internal** – These are for internal use only, and not shared with partners or the public. Internal APIs communicate and share data between divisions or departments.

- **Partner** – These are typically owned and secured APIs whose connection information is controlled and allowed only with business partners. Application development companies interact with each other, and if allowed they share APIs. An example of partner APIs is healthcare record data exchange systems. Data is exchanged using APIs across multiple connected partner entities like a doctor's office, pharmacy, and healthcare network.

- **Composite** – These are a combination of data and service functions in a single API. Services allow access to multiple endpoints with a single call. They are used when information needs to be shared between resources without having to develop and support multiple API instances.

APIs have different protocols. Protocols are literally the architectural style and format in which the API is developed. The reason for establishing protocols is having a shared set of defined rules for the use of data types and commands. It is the protocol that defines the formatting of data or information that is exchanged by the API. It is possible to connect one API to multiple protocol types, but doing so increases the complexity of the API. The following are the commonly used API protocols:

- **REST -** REST or RESTful API stands for Representational State Transfer. REST API's function is based on a known set of architectural rules that control communication and data transfer.

- **JSON –** JSON (JavaScript Object Notation) is a lightweight format for transferring and storying data. JSON is self-describing and typically easier to understand coding language. JSON is frequently used with a REST/RESTful API.

- **XML -** XML stands for eXtensible Markup Language. XML is designed to store and transport data. XML was designed to be both human- and machinereadable. XML is a formatting code language that uses remote procedure calls to perform data transfer.

- **SOAP -** SOAP stands for Simple Object Access Protocol. SOAP is built using XML to perform data transfer operations. SOAP has the ability using XML to share data between applications and components even if written in different coding languages. In other words, it uses XML to exchange information in a header/body (envelope) format.

## Definition of an Endpoint

The endpoint of an API is a connection point to a digital location where requests are sent to retrieve data. An endpoint requires a set of connection data for access, and possibly a security token. APIs typically require the following for an endpoint to process a request: URL, method, header(s) and body. The header contains the metadata describing the request and the body contains the message data.

Endpoints are essentially server locations or access points to a server between two applications or services. Without endpoints, API's cannot communicate with each other.

## Principles of Data Exchange & Messaging

Data exchange systems and messaging are related methods that use APIs. A data exchange system is developed for a domain. Routines or mapping is done to translate data in and out of the domain. A group of data domains is an exchange system.

Each domain is responsible for adhering to the API protocol and type used. Then, the system exchanges data through secured API endpoints. The data exchange itself is the communication between APIs using a protocol like JSON, or XML as examples.

Messaging is the actual request sent to the API. The "message" contains the endpoint, security token (if used), metadata, and the body. The metadata is a description of the contents of the body, and the body contains the data itself in code or other defined document type, such as PDF or JPG.

## API Security

APIs expose endpoints that handle object identifiers, creating a wide attack surface to steal data or intercept requests. Numerous methods exist for securing API endpoints and the servers connected to those endpoints. As a software tester, become familiar with API security so problems can be recognized during the development cycle.

When API endpoints are not secured properly, the risk of a data and system breach is high. Most APIs use access tokens to authenticate user interactions. However, testing access to the token granting authority is secure and that the assets are both stored and communicated securely is critical. API security is a deep and substantial topic.

In this syllabus, we'll discuss the basics of token authentication and JWT, verifying expiration dates, and securing storage locations.

For more information on API security, or application security in general, read the OWASP site regularly or sign up for updates: https://owasp.org/www-project-api-security/

## Error Messaging

As Software Testers, we are interested in both successful requests and those that produce an error. It's essential for testing API's to test both the response that is successful, and those that present errors. When testing APIs, errors are returned when a request fails instead of producing a correct response. Typically, the error message returns a description of the error, an error code, HTTP error response message, and an HTTP error response code.

An API error response includes the following:

- A textual error message
- Details of the error, if any
- General description of the error
- Error response code value
- HTTP response with HTTP response textual message and an HTTP response code.

The following are response codes that APIs use to communicate the status of a request with the sender:

- 100 level - Informational - acknowledges the request was received
- 200 level - Success - request received and completed
- 300 level - Redirection - request is pending, client action needed
- 400 level - Client error - request is invalid
- 500 level - Server error - internal server failure consuming a request

As the API tester, create API tests to test for each of these responses to the request. Some exceptions may apply as 500 errors are generally internal server errors requiring IT assistance or IT intervention. The quality of the error text description is relative to the developer of the API.

# Testing Techniques

## Using Tools

In this micro-credential course, examples shown are in the Postman tool. There are numerous tools that provide API testing frameworks in addition to, or instead of Postman.

Software testers will likely use a tool for API testing. However, the tool used is typically chosen by the development team or the business entity. In this course, we are not covering how to use a specific tool. Postman was chosen because it has a free version that can be downloaded and used to practice API testing. Postman also offers an abundance of online tutorials and reference resources for learning more about API functionality.

## Functional Testing

The majority of testing with APIs is functional testing. A software tester performs some security testing but the main or primary type of testing is functional.

Functional testing ensures the API functions as expected and generates the expected responses with the expected data and format. API testing is defined as a grey-box testing technique. It is grey box because the tester have internal knowledge of the API and application functions related to it.

### 3.2.1 GET, POST, PUT

GET, POST, PUT, and DELETE are common HTTP (and HTTPS) methods used when testing APIs. In this micro-credential we'll cover GET, POST, and PUT only as those are the ones software testers primarily use.

GET retrieves data from the API. In other words, the request is "getting" data back in the response. Similar to running a SQL query on the database, with a GET request the data requested comes back in the form of a response. The GET method is often used to retrieve a security authentication token or spot check data in the database.

POST sends new data to the API. The request is "posting" data or adding data.

PUT updates existing data. PUT or PATCH do the same function, they update the data value that already exists in the database. Use PUT to update existing database values.

## 3.2.2 REQUEST/RESPONSE

The request is the message created and sent to the API endpoint. Within the request, designate the data the tester is interested in adding, retrieving, or editing based on the HTTP method used (GET, POST, PUT). Once the request is sent, the API responds with a textual response message viewable in a variety of formats including JSON, XML, and HTML. If a tool is used, the response is typically shown in a "test results" area of the tool.

## 3.2.3 API DATA VALIDATION

Once a response message is received, test or validate the data received based on the parameters of the request. There are multiple ways to validate the API data. The tester can use the API response message and step through it line by line. Or consider using the browser developer tools option to view the API and data associated to it.

It is important to test for error conditions that may not produce error messages from the API. For example, data that may exceed the size of the data expected by the other service or application may be accepted, but later cause a system failure. These types of defects are not uncommon with APIs because input validation of the service or application at the endpoint is often inadequate to catch all forms of invalid input.

Likewise, applications that use APIs to receive data should also have strong input validation edits to prevent invalid or erroneous data from being applied.

An example of input validation from an API would be receiving a monetary conversion rate from a

currency conversion web service. Just checking for numeric input is not enough. A valid range of input must also be specified and applied to prevent greatly incorrect conversion rates for monetary calculations.

## Error Testing

When testing APIs, the response returns a status code as discussed previously in section 2.5 of this document. Also test, when possible, all the error codes possible for the API or status codes 200 to 400. Status 500 is testable, but requires additional IT involvement as it impacts the server the API lives on.

## Automating API Tests

Most tools offer a method to automate API tests. Typically, a base set, or “canned” set of code snippets are provided. Testers can enhance or edit these code snippets to check for specific data types or exact data matches returned in the response message. Automated API tests add depth and a repeatable test execution option that is low maintenance.

# Challenges

## Server Access & Security Tokens

API test execution requires planning not only for those actually executing the tests, but the IT team in charge of the servers the APIs live on, and accessing those servers. The tester will need their expertise to determine when tests can be executed without interfering with the server system when testing in the live or production server. If testing in a development or other non-production server, then development assistance is usually required. The tester also needs IT or Development help to schedule error tests because sending bad test data, without warning, is a bad idea. IT or Development may need to monitor the APIs during testing, in case a server-side failure occurs during the test.

Testers don't want to take down a non-test server and all its connections. Additionally, when testing on an internal server, with secured APIs, the tester needs to request token access. Without the security token access, tests cannot get past the authentication functionality. Tokens change constantly, so gaining access to the updated token is a constant concern.

It is preferred to testing APIs initially on a non-production environment. By testing on non-production servers the risk to the production server is significantly reduced.

## Tools

Tools are challenging to learn. Time for creating practice API tests is essential for fully learning the tool to create valid, and valuable API tests. Testers need to keep in communication with developers so they understand how an API works, and gain access to API documentation. The API documentation, written by developers contains the API structure as well as endpoint and security requirements. During test development, the tester may notice gaps or missing information in the API documentation. Check with the developer who wrote the documentation to request updates.

## Test Maintenance

API testing, like the majority of tests, requires maintenance. Whenever the API code changes, the tests need updated. The frequency of API test maintenance depends on the frequency new changes are made to your existing APIs.

# Trademarks

The following registered trademarks and service marks are used in this document.

AT*SQA® is a registered trademark of the Association for Testing and Software Quality Assurance Global Certification Body, Inc

Postman® is a registered trademark of Postman

# Purpose of this Document

This syllabus and body of knowledge form the AT*SQA micro-credential course for API Testing.

AT*SQA is an International Standards Organization (ISO) compliant certification body for software testers. AT*SQA provides this syllabus as follows:

1. To training providers–to produce courseware and determine appropriate teaching methods.

2. To certification candidates–to prepare for the exam (as part of a training course or independently).

3. To the international software and systems engineering community–to advance the profession of software and systems testing and as a basis for books and articles.

AT*SQA may allow other entities to use this syllabus for other purposes, provided they seek and obtain prior written permission.

# References


https://www.ibm.com/cloud/learn/api

https://learning.postman.com/docs/getting-started/sending-the-first-request/

https://www.baeldung.com/rest-api-error-handling-best-practices

https://owasp.org/www-project-api-security/

https://smartbear.com/learn/performance-monitoring/apiendpoints/#:~:text=Simply%20put%2C%20an%20endpoint%20is,of%20a%20server%20or%20service

https://www.freecodecamp.org/news/rest-api-best-practices-rest-endpointdesign-examples/

https://www.ibm.com/cloud/learn/api


www.atsqa.org