**SGAutoPro Automation Framework User Manual**

A blue and grey logo

Description automatically generated

A logo with a pink and blue circle

Description automatically generated

SGAutoPro Automation Framework User Manual

Version 1.0

September 2025

**Table of Contents**

[1 Introduction 7](#_Toc211249239)

[2 Project Configurations 7](#_Toc211249240)

[3 Global Values for a Testcase 10](#_Toc211249241)

[4 Operations 12](#_Toc211249242)

[4.1 Common Keyword 12](#_Toc211249243)

[4.2 Assert 13](#_Toc211249244)

[4.3 API Operations 15](#_Toc211249245)

[4.3.1 CALLAPI 15](#_Toc211249246)

[4.3.2 PARSEAPIRESPONSE 15](#_Toc211249247)

[4.4 CALCULATE 15](#_Toc211249248)

[4.5 CALLTESTCASE 16](#_Toc211249249)

[4.6 CLEAR 17](#_Toc211249250)

[4.7 CLICK 17](#_Toc211249251)

[4.7.1 Clearing on Mobile Screen by 18](#_Toc211249252)

[4.7.2 Clicking Image 18](#_Toc211249253)

[4.8 CLICKBYOFFSETPERCENTAGE 18](#_Toc211249254)

[4.9 CLICKANDTYPE 19](#_Toc211249255)

[4.10 CLICKIFVISIBLE 19](#_Toc211249256)

[4.11 CONDITION 20](#_Toc211249257)

[4.12 Database Operations 22](#_Toc211249258)

[4.13 Double Click 22](#_Toc211249259)

[4.14 DRAGANDDROP 22](#_Toc211249260)

[4.15 GENERATERANDOM 23](#_Toc211249261)

[4.16 GETELEMENTCOUNT 25](#_Toc211249262)

[4.17 GETTEXT 26](#_Toc211249263)

[4.18 GETTITLE 27](#_Toc211249264)

[4.19 GETURL 28](#_Toc211249265)

[4.20 HOVERMOUSE 28](#_Toc211249266)

[4.21 ISPRESENT 28](#_Toc211249267)

[4.22 ISSELECTED 29](#_Toc211249268)

[4.23 LOAD 30](#_Toc211249269)

[4.24 WAIT 30](#_Toc211249270)

[4.25 GLOBALWAIT 32](#_Toc211249271)

[4.26 GETARRAYSIZE 33](#_Toc211249272)

[4.27 SLEEP 33](#_Toc211249273)

[4.28 STRINGMANIPULATION 33](#_Toc211249274)

[4.29 SPLIT 35](#_Toc211249275)

[4.30 LOOP 35](#_Toc211249276)

[4.31 SCROLL 37](#_Toc211249277)

[4.32 SCROLLTOELEMENT 38](#_Toc211249278)

[4.33 SWITCHFRAME 40](#_Toc211249279)

[4.34 Selecting a value in a locator 41](#_Toc211249280)

[4.34.1 SELECTBYINDEX 41](#_Toc211249281)

[4.34.2 SELECTBYTEXT 41](#_Toc211249282)

[4.34.3 SELECTBYVALUE 41](#_Toc211249283)

[4.35 SENDSPECIALKEY 42](#_Toc211249284)

[4.36 SETPROPERTY 42](#_Toc211249285)

[4.37 Mobile Specific Tags 43](#_Toc211249286)

[4.38 TYPE 43](#_Toc211249287)

[4.39 GETATTRIBUTE 44](#_Toc211249288)

[4.40 JAVAScript Execution 45](#_Toc211249289)

[4.41 Handling multiple Windows 45](#_Toc211249290)

[4.41.1 GETWINDOWHANDLES 45](#_Toc211249291)

[4.41.2 GETCURRENTWINDOWHANDLE 46](#_Toc211249292)

[4.41.3 SWITCHTOWINDOWHANDLE 46](#_Toc211249293)

[4.42 Date Functions 47](#_Toc211249294)

[4.42.1 DATEOPERATIONS 47](#_Toc211249295)

[4.42.2 DateOperation2 48](#_Toc211249296)

[4.43 Browser Operations 52](#_Toc211249297)

[4.43.1 NAVIGATE BACK 52](#_Toc211249298)

[4.43.2 NAVIGATE FRONT 52](#_Toc211249299)

[4.43.3 REFRESH 52](#_Toc211249300)

[4.44 Get Element Size 52](#_Toc211249301)

[4.44.1 GET WIDTH 52](#_Toc211249302)

[4.44.2 GET HEIGHT 52](#_Toc211249303)

[5 Appendix 1 53](#_Toc211249304)

[5.1 Array 53](#_Toc211249305)

[5.2 Rank 55](#_Toc211249306)

[6 APPENDIX 2 56](#_Toc211249307)

[6.1 KEY VALUE MAPPING 56](#_Toc211249308)

[6.2 DATABASE 56](#_Toc211249309)

[6.3 REST API 58](#_Toc211249310)

[6.4 Working with QTEST 61](#_Toc211249311)

[6.4.1 Configuring the Project Property to Connect with qTest 61](#_Toc211249312)

[6.4.2 Configuring the testcase 62](#_Toc211249313)

[6.4.3 Creating Test plan in qTest Project 63](#_Toc211249314)

[6.4.4 Creating Test Suite in qTest for Automation Results 64](#_Toc211249315)

[6.5 RUNNING AUTOMATION JAR WITH EXTRA PARAMETERS 65](#_Toc211249316)

[6.5.1 Rerun of failure or skipped Testcases 65](#_Toc211249317)

[6.5.2 Running automation using extra parameters 65](#_Toc211249318)

[6.5.3 Preparing data for iLottery CICD 66](#_Toc211249319)

[6.6 CHECK DUPLICATE KEYS 67](#_Toc211249320)

[6.6.1 Checking duplicate key within a property file: 67](#_Toc211249321)

[6.6.2 Checking duplicate keys within multiple properties files 67](#_Toc211249322)

[7 APPENDIX 3 68](#_Toc211249323)

[7.1 Setting browser capabilities from external file 68](#_Toc211249324)

[7.2 Running Appium with image plugin 69](#_Toc211249325)

[7.2.1 KEYS USED IN MOBILE.PROPERTIES 69](#_Toc211249326)

[7.2.2 METHODS SUPPORTED 69](#_Toc211249327)

[7.3 Parameterizing Testcases Based on external Inputs 70](#_Toc211249328)

[7.3.1 PREPARING THE DATA 70](#_Toc211249329)

[7.3.2 EXECUTION STEPS 70](#_Toc211249330)

**Documentation Policy**

**Document Name: SGAutoPro Getting Started**

**Date: February 08, 2024**

This document contains information confidential and proprietary to Scientific Games International, Inc. (SGI), a wholly owned subsidiary of Scientific Games Corporation (SG) and may be protected by other forms of intellectual property. The information contained herein may not be reproduced, paraphrased, translated into another language, or otherwise disclosed or duplicated by any means without the prior written consent of SGI. Violators may be prosecuted.

Due to the confidentiality of the subject matter, revisions or alterations are undertaken only by written agreement with SGI. The information contained in this document is subject to change by SGI without notice. Distribution of this document or any portion thereof may only be conducted by SGI or through its designated officials.

The software / systems described in this document are protected by copyright law. Unauthorized duplication or use of any software described herein is prohibited. Software features may be described herein which are not implemented in every system. Similarly, ongoing enhancements to this system may result in the presence of features on a particular system, which are not included in this edition of the document. Authorized users will be notified of software / systems modifications through the release of new editions of this documentation.

SGI MAKES NO WARRANTY, EXPRESSED OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY OF FITNESS FOR ANY PARTICULAR PURPOSE. SGI ACCEPTS NO LIABILITY FOR ERRORS CONTAINED IN THIS DOCUMENT OR FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES ASSOCIATED WITH ITS PROVISION, CONTENTS, OR USE.

© 2024 by Scientific Games International, Inc.

All rights reserved.

Revision History

| **Version** | **Date** | **Description** | **Authored By** |
| --- | --- | --- | --- |
| 1.0 | 02/08/2024 | Initial version of Automation Framework Documentation. | Manoj |
| 1.1 | 05/31/2024 | Running with additional parameters, click by offset, implementation of dependent TCs | Manoj |
| 1.2 | 07/25/2024 | test Testcase update. | Manoj |
| 1.3 | 08/01/2024 | As per Release Note of Sprint 14 | Mukesh |
| 1.4 |  | Sprint 16 (Date Comparison) | Manoj |
| 1.5 | 11/26/2024 | Sprint 17 & 18 Changes added | Manoj |
| 1.6 | 01/22/2025 | Sprint 20 Changes | Mukesh |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

This document provides an overview of SGAutoPro Automation Framework.

# Project Configurations

|  |  |
| --- | --- |
| ***Keyword*** | ***Values*** |
| driver\_choice | Ensure successful execution of automation scripts by selecting the appropriate browser based on your testing requirements. The supported browsers for running automation are:   * Mozilla Firefox: firefox\_browser * Internet Explorer: ie\_browser * Google Chrome: chrome\_browser * Microsoft Edge: edge\_browser * Android Web Browser: android\_web * Android App: android\_app * iOS App: ios\_app * iOS Web Browser: ios\_web * API Testcases: no\_browser |
| testcase\_path | This key points to the path where testcases files are stored for execution.  The test cases are stored in JSON format for easy management and execution. |
| run\_all\_case | * When the parameter is set to true, all test cases will be executed regardless of the 'run' tag assigned to each testcase. * Conversely, when the parameter is set to false, only those test cases with the 'run' tag explicitly set to true will be executed.   This approach allows users to choose specific test cases for execution based on their 'run' tag values should be used during creating and debugging testcases. |
| ignore\_case\_driver | When the parameter is set to “true”, automation will override individual testcase-specific driver configurations. Instead, it will run all test cases using the driver specified at the project level. This provides a centralized approach to manage the execution environment, ensuring consistency across all test cases. |
| ignore\_screenshot\_key | * When the parameter is set to true, automation will refrain from capturing screenshots at the step level. This provides a streamlined execution process without generating screenshots for each individual step. * Conversely, when the parameter is set to false, automation will capture screenshots at the end of each step based on the 'screenshot' flag. If the 'screenshot' flag is set to true for a particular step, a screenshot will be taken at the conclusion of that step. |
| elements\_path | This key points to the directory path for object repository, containing all elements or locators essential for test automation. |
| globaldata\_path | This key points to the directory path for the user-defined properties files, instrumental for storing and retrieving variables during automation and runtime modification.  Variables in these files can be accessed in the automation script using the syntax **${filename.key},** where 'filename' is the name of the properties file (excluding the extension). |
| testdata\_path | The **testdata.properties** file, housing static test data that remains unchanged during automation runtime, should be stored in the path specified by this key. |
| mobile\_path | The mobile-specific files, containing static data that remains unchanged throughout the automation runtime, should be stored in the path specified by this key. |
| environment\_path | he properties file containing environment variables for display in the Extent Report can be found in the path specified by this key. |
| tag\_specific\_run | When the parameter is set to true, test cases with tags specified in the 'tags\_to\_run' configuration will be executed. This allows for a targeted approach, running only those tests cases associated with specific tags as defined in the 'tags\_to\_run' configuration. |
| tags\_to\_run | For example – Smoke, Regression |
| default\_wait\_timer | The default timer is applied whenever the automation process involves clicking or typing into a control. This timer is designed to introduce a delay between actions, ensuring proper synchronization with the application under test. |
| default\_locator\_type | The automation system considers the default identifier type for all identifiers inside the test script if the **identifier\_type** is not explicitly provided. The default value is set to 'xpath'. |
| maxium\_scroll\_counts | Maximum number of scrolls required until required element is found. This is used inside ScrollToElement operation. |
| iisServer | Used for preparing the data for Jenkins. Provide the server address of the web server where reports are hosted. If kept blank or commented, link will be created with localhost IP. |
| tolerance\_to\_failure\_percent | When set, Automation will skip the execution of testcases, once the failure testcase percentage exceeds beyond this level. To keep it disabled, set to 101. |
| retry\_count | When set, automation will retry the failure testcase for times set in this key. All the failures will be reported as skipped until the final try.  Please note, this will increase the testcase count in the report as there will be additional skipped testcases. |
| print\_to\_csv | To dump the testcase properties to csv file. Should be used for development purposes only. |
| skip\_on\_main\_test\_fail | When **true**, the current test script is skipped, if any of the dependent testcase is marked as failed or skipped. If not provided, default value is considered as **false**. |
| qTest | For uploading data to qTest, please follow [Working with qtest](file:///C:\Users\suma.b\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\5TDD4QTD\SGAutoPro%20User%20Manual%20(Sprint%2020).docx#_Working_with_qtest) |
| fails\_on\_console\_error | When true, console errors are checked at the end of the TC and TC is marked as failed, if there is any testcase. |

***Note:*** *All the keywords are case sensitive and should be provided with lowercase in any of the operation.*

A screenshot of a computer screen

AI-generated content may be incorrect.

# Global Values for a Testcase

The values mentioned here are applied to a testcase.

|  |  |
| --- | --- |
| ***Keyword*** | ***Values*** |
| run | The **run** parameter plays a crucial role while developing testcases.  Set the value to **true** while developing a script and make **run\_all\_case=false** along with **tag\_specific\_run=false** inside project property.  On completing the development of the scripts, **run** can be set to **False.**  Once all the scripts are developed and need to be run as part of automation tests, the running of these scripts can be controlled using  **Run\_all\_case=true** or **tag\_specific\_run=true**.  **Note:** Once enabled, these two settings will ignore the **run** parameter. |
| name | Name of the testcase. This will be part of the report.  It is recommended to use a testcase id as part of name and as part of name of the Json file to relate the testcase in report and the associated script file. |
| desc | To provide a meaningful purpose for the testcase. |
| testCaseIds | A test automation script may fulfil multiple testcases. This field is in an array of testcase IDs from qTest to link the script and related testcases. Multiple testcases can be provided with comma separated. For single testcase, just provide the testcase ID.  For more details refer to [Appendix 6.1.4](#_Working_with_QTEST) Working with qTest. |
| dependentTestCaseIds | This tag holds a testcase ID or an array of testcase IDs. If those testcases are failed or skipped during execution, current test script will also be skipped. All the testcases mentioned in current script inside tag testCaseIds, will be marked as skipped in qTest. |
| properties | The **Properties** variable is a local variable of **HashMap** type, confined to the scope of the testcase. Users can utilize this variable to store and retrieve variables dynamically during the execution of the test script. Variables stored in the **Properties** can be accessed and reused across different steps within the same test script, providing a flexible and efficient way to manage and share values during automation. |
| rank | The **rank** parameter is utilized to determine the order of execution within your automation process. By assigning a specific rank value, you can control the sequence in which testcases, or steps are executed. Lower rank values are executed first, followed by higher rank values. For details refer to Rank in [Appendix 5.2](#_Rank) |
| tags | * Tags like "SmokeTest","RegressionTest","AccountsTests" * Multiple tags can be associated with testcases. * This will group the testcases and a group of testcases can be run by using the key **tags\_to\_run** from project properties. |
| array\_properties | The **array\_properties** is a feature designed to create arrays, offering a functionality similar to **Properties**. More detailed information on how to leverage **array\_properties** can be found in Section 5 of this user manual. This feature provides a flexible way to manage and manipulate arrays within your automation scripts, enhancing the versatility and efficiency of your test implementations. |
| driver | Takes the same input as in driver\_choice for project. property. This value will override project level driver input if **ignore\_case\_driver=false** in project. property file. |
| dataset | The dataset will have two parameters. The first parameter denotes the location of the test data spreadsheet. The second parameter denotes the sheet name from which data is extracted. A dataset tag has the format:  "dataset":["location of the excel spreadsheet", "Sheet name"],  This is used for parameterization of testcase. |

A computer screen shot of a program

AI-generated content may be incorrect.

# Operations

Below mentioned operations and associated keywords are used to form steps for the testcase. Properties of each operation are described below:

## Common Keyword

Below are the common keywords which can be applied in any operation.

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| name | The 'Name' property serves as a unique identifier for each operation within a testcase. This property not only provides a descriptive label for the operation but also contributes to the clarity of the testcase flow when reviewing automation reports.  When designing your testcases, assign meaningful names to operations. These names will be reflected in the automation reports, making it easier for users to comprehend and analyze the flow of test execution. Choose informative and concise names for operations to enhance the overall readability of your automation reports." |
| skip | When the 'Skip' parameter is set to 'True', the framework will skip the execution of the associated step during automation runs. This feature provides a convenient way to selectively exclude specific steps from execution without altering the overall testcase structure while debugging or developing the test script. |
| desc | The 'Desc' property is utilized to provide a descriptive annotation or explanation for a step within a testcase. Users can use this property to add contextual information, notes, or any other details that help in understanding the purpose or functionality of the step |
| log | The 'Log' property serves as a mechanism to log user-defined errors or messages after the execution of a step. This feature is particularly useful during debugging or script creation, providing testers with insights into the script's behavior. |
| screenshot | When the 'screenshot' property is set to 'True', the framework will capture an on-demand screenshot after the execution of the step. This functionality provides users with the ability to include screenshots for specific steps as needed while debugging or creating the test script.  To keep the automation light weight, it is suggested to use it during development of the test scripts. |

## Assert

**Mandatory Keywords:** **Name**, **operand**\_**lhs**, **operand**\_**rhs**, **compare**\_**by**

**Optional Keywords:** Desc, Log, skip, screenshot

**Use**: "The **ASSERT** operation is employed to confirm comparison conditions within a testcase. When the comparison does not match, the operation fails the testcase and logs an error. If a custom log message is not provided, a default message is printed. The default error message format is as follows:

'<<Custom value>>' is not matching with the actual condition '<<Custom value>>' when compared by the operator '<<operator provided>>.'

Users can customize this message by providing their own log message for better clarity on the nature of the assertion failure. The **ASSERT** operation enhances the reliability of testcases by validating expected conditions and highlighting discrepancies during automation runs."

Values for the Keywords:

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| operand\_lhs | The 'operand\_lhs' property represents the value of the left-hand side operand used for comparison within an operation. |
| operand\_rhs | The 'operand\_rhs' property represents the value of the right-hand side operand used for comparison within an operation. |
| compare\_by | EQUALS, NOT\_EQUALS, CONTAINS, STARTS\_WITH, ENDS\_WITH, EQUALS\_IGNORE\_CASE, LESS\_THAN, LESS\_THAN\_EQUALS, GREATER\_THAN GREATER\_THAN\_EQUALS  Default value for compare\_by is EQUALS, if no value is provided. |
| Operation\_type | Soft\_assert |

When **Operation\_type** is “**soft\_assert**”, testcase will continue to execute even if the step fails. On completion of testcase, failed testcases will be reported. If no operation\_type is present, default is normal assert. Execution will be halted at the same step.

**Note:**

* Values are not case sensitive.
* “Log keyword” when provided in other steps, prints the log to the report step. Only for Assert, it will be printed when assert fails.

**Example:**

A computer screen with text and images

AI-generated content may be incorrect.

## API Operations

### CALLAPI

CALLAPI operation is used for creating a call to API.

### PARSEAPIRESPONSE

Parse API response is used after the CALLAPI operation to parse the response and get required result.

Please refer to [Appendix 2](#_APPENDIX_2) for more details.

## CALCULATE

Performs all types of mathematical operations.

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | Calculate |
| operation\_type | add, multiply, divide, subtract, reminder, absolute, min, max, ceil, floor, sqrt, percentage, power |
| Operand\_hls | First operand for the calculation |
| Operand\_rhs | Second operand for the calculation.  Some of the operation types do not need second operand. Anything provided in operand\_rhs will be ignored. |
| output | To store the output from the manipulation operation. |

**Implementation:**

Below table demonstrates with an example how this operation works.

\*Few of the operation if one of the operands is decimal, result is returned as decimal.

Others will always return decimal.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation** | **operation\_type** | **operand\_lhs** | **operand\_rhs** | **output** |
| CALCULATE | \*ADD | 10 | 15 | 25 |
|  |  | 10.0 | 15 | 25.0 |
|  | \*MULTIPLY | 10 | 12 | 120 |
|  |  | 10.0 | 12 | 120.0 |
|  | \*DIVIDE | 100 | 25 | 4.0 |
|  |  | 100.0 | 25 | 4.0 |
|  | \*SUBTRACT | 100 | 10 | 90 |
|  |  | 100 | 10.0 | 90.0 |
|  | \*REMINDER | 22 | 7 | 6 |
|  |  | 22 | 7.0 | 6.0 |
|  | \*ABSOLUTE | -22 | NA | 22 |
|  |  | -22.0 | NA | 22.0 |
|  | \*MIN | 10 | 9 | 9 |
|  |  | 10.0 | 9 | 9.0 |
|  | \*MAX | 10 | 9 | 10 |
|  |  | 10.0 | 9 | 10.0 |
|  | CEIL | 10.6 | NA | 11.0 |
|  | FLOOR | 10.6 | NA | 10.0 |
|  | SQRT | 10 | NA | 3.1622776601683795 |
|  | POWER | 2 | 3 | 8.0 |

A screen shot of a computer code

AI-generated content may be incorrect.

## CALLTESTCASE

**Mandatory Keyword:** Name, value

**Optional Keywords:** Desc, skip

**Use:** The **CALLTESTCASE** operation is employed to execute a JSON file specified in the value parameter. This feature allows users to modularize their test automation by calling and executing separate test cases defined in external JSON files.

Values for the key words:

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Value | Path of the json file for which testcase to be called and executed. |

**Note:**

* The root folder for this file will be as specified in the “Testcase\_path” tag in project properties file.
* Values are not case sensitive.

**Example:**

A screenshot of a computer

AI-generated content may be incorrect.

## CLEAR

**Mandatory Keywords:** Name, identifier

**Optional Keywords:** Desc, identifierType, skip

**Use:** CLEAR removes the texts present in a textbox. If this element is a form entry element, this will reset its value.

Values for the key words:

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Identifier | Path of the locator (textbox) |
| IdentifierType\*\* | NAME, ID, XPATH, CLASS\_NAME, LINK\_TEXT, PARTIAL\_LINK\_TEXT, TAG\_NAME, CSS  \*\* Default value for IdentifierType is XPATH, if not provided. |

## CLICK

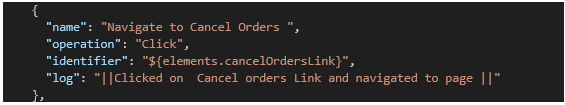
**Mandatory Keywords:** Name, identifier

**Optional Keywords:** Desc, IdentifierType, skip

**Use:** To click on the provided identifier. There are some preconditions for an element to be clicked. The element must be visible, and it must have a height and width greater than 0.

Values for the key words:

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Identifier | Path of the locator (textbox) |
| IdentifierType\*\* | NAME, ID, XPATH, CLASS\_NAME, LINK\_TEXT, PARTIAL\_LINK\_TEXT, TAG\_NAME, CSS  \*\* Default value for IdentifierType is XPATH, if not provided. |



### Clearing on Mobile Screen by

This is currently implemented for Mobile Devices through Appium Image Option. This will clear the input fields wherever the focus is.

Appium server should be started with image plugin for this to work. For details follow [Appendix-3](#_APPENDIX_3).

Use "operation\_type": "ON\_MOBILE\_SCREEN".

A black screen with orange text

AI-generated content may be incorrect.

### Clicking Image

This is currently implemented for Mobile Devices through Appium Image Option.

Appium server should be started with image plugin for this to work. For details follow [Appendix-3](#_APPENDIX_3).

Use "operation\_type": "ON\_MOBILE\_SCREEN",

**Example:**

A black background with white text

AI-generated content may be incorrect.

## CLICKBYOFFSETPERCENTAGE

**Use:** To click on the provided identifier at a particular position.

Values for the key words:

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Identifier | Path of the locator (textbox) |
| IdentifierType\*\* | NAME, ID, XPATH, CLASS\_NAME, LINK\_TEXT, PARTIAL\_LINK\_TEXT, TAG\_NAME, CSS  \*\* Default value for IdentifierType is XPATH, if not provided. |
| x\_offset | For click at Left or Right from center of the object. If the Value is -ve, it is towards left. If the value is +ve, it is towards right. |
| y\_offset | For click at Up or Down from center of the object. If the Value is -ve, it is towards Up. If the value is +ve, it is towards down. |

**Example:**

A black screen with white text

AI-generated content may be incorrect.

## CLICKANDTYPE

**Mandatory Keywords:** Name, identifier

**Optional Keywords:** Desc, IdentifierType, skip

**Use:** Single method to click on a locator and type the provided value. There are some preconditions for an element to be clicked. The element must be visible, and it must have a height and width greater than 0.

Values for the key words:

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Identifier | Path of the locator (textbox) |
| Value | Texts to be typed to the locator |
| IdentifierType\*\* | NAME, ID, XPATH, CLASS\_NAME, LINK\_TEXT, PARTIAL\_LINK\_TEXT, TAG\_NAME, CSS  \*\* Default value for IdentifierType is XPATH, if not provided. |

## CLICKIFVISIBLE

**Mandatory Keywords:** Name, Identifier

**Optional Keywords:** Desc, IdentifierType, skip

**Use:** Method to click an element if it is visible. No exception will be raised if the element is not visible. Automation will move to the next step.

Values for the key words:

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Identifier | Path of the locator. |
| IdentifierType\*\* | NAME, ID, XPATH, CLASS\_NAME, LINK\_TEXT, PARTIAL\_LINK\_TEXT, TAG\_NAME, CSS  \*\* Default value for IdentifierType is XPATH, if not provided. |

**Example:**

A screen shot of a computer

AI-generated content may be incorrect.

## CONDITION

**Mandatory Keywords:** Name, operand\_lhs, operand\_rhs, compare\_by, if\_true

**Optional Keywords:** Desc, Log, if\_false, skip

**Use:** The **condition** operation is used to verify a specified condition and execute steps based on the outcome. If the condition is true, the **if\_true** steps will be executed. **If\_false** will be executed otherwise.

Steps inside the **If\_false** condition are not mandatory. However, it is essential to maintain the structure by including a blank step. Failing to do so may result in a null exception being raised. The presence of a blank step ensures proper syntax and execution, even when no specific steps are intended for execution when the **condition\_expression** evaluates to false.

Values for the key words:

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| operand\_lhs | The "operand\_lhs" property represents the value of the left-hand side operand in a comparison operation. |
| operand\_rhs | The "operand\_rhs" property represents the value of the right-hand side operand in a comparison operation. |
| compare\_by | EQUALS, NOT\_EQUALS, CONTAINS, STARTS\_WITH, ENDS\_WITH, EQUALS\_IGNORE\_CASE, LESS\_THAN, LESS\_THAN\_EQUALS, GREATER\_THAN, GREATER\_THAN\_EQUALS |
| If\_true | The "If\_true" property specifies the steps to be performed if the associated condition is satisfied or evaluates to true. |
| If\_false | The "If\_false" property specifies the steps to be performed if the associated condition is satisfied or evaluated to false. |

**Example:**

A screen shot of a computer program

Description automatically generated

## Database Operations

Database operations include 3 major operations as explained below. For details how to use these functions refers to [Appendix 2](#_APPENDIX_2).

|  |  |
| --- | --- |
| **Query** | **Operations** |
| RUN\_DB\_QUERY | To run a database query |
| PARSE\_DB\_RESULT | To get results for a particular column and row |
| GET\_DB\_RESULT\_SIZE | To get the result size |

## Double Click

**Mandatory Keywords:** Name, identifier

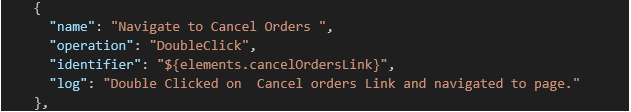
**Optional Keywords:** Desc, IdentifierType, skip

**Use:** To double click on the provided identifier. There are some preconditions for an element to be clicked. The element must be visible, and it must have a height and width greater than 0.

Values for the keywords:

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Identifier | Path of the locator (textbox) |
| IdentifierType\*\* | NAME, ID, XPATH, CLASS\_NAME, LINK\_TEXT, PARTIAL\_LINK\_TEXT, TAG\_NAME, CSS  \*\* Default value for IdentifierType is XPATH, if not provided. |

**Example:**



## DRAGANDDROP

**Mandatory Keywords:** Name, identifier

**Optional Keywords:** Desc, IdentifierType, skip

**Use:** DRAGANDDROP will drag the parent element (parent\_identifier) and drop on the element (Identifier) provided.

DRAGANDDROPBYOFFSET will do the same but instead of dropping on the element, it will drop on the coordinate provided.

Values for the keywords:

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| parent\_identifier | Path of the element which you want to drag |
| Identifier | Path of the element where you want to drop the parent element |
| operation | DRAGANDDROP |
|  |  |

Another way to drag and drop is to use the offset. Here Just pass the element as identifier and provide the x and y offset

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| parent\_identifier | Path of the element which you want to drag |
| identifier | Path of the element where you want to drop the parent element |
| operation | DRAGANDDROP |
| operation\_type | DRAGANDDROPBYOFFSET |
| x\_offset | X axis |
| y\_offset | Y axis |

**Example:**

        {

            "name": "Drag an element and drop on another element",

            "operation": "DragAndDrop",

            "parent\_identifier": "${elements.manage\_subscription\_link}",

            "identifier": "${elements.loginadminOKButton}"

        },

        {

            "name": "Drag an element and drop at certain offset",

            "operation": "DragAndDrop",

            "operation\_type": "DragAndDropByOffset",

            "identifier": "${elements.loginadminOKButton}",

            "x\_offset": "100",

            "y\_offset": "200"

        }

## GENERATERANDOM

**Mandatory Keywords:** Name, operation\_type, value, output

**Optional Keywords:** Desc, skip

**Use:** Creates a random number or string of provided type.

Values for the key words:

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| operation\_type | STRING, ALPHA\_NUMERIC |
| value | Length of the string or number |
| output | Variable where it is to be saved |

A screen shot of a computer program

Description automatically generated

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| operation\_type | EMAIL |
| value | Domain name like scientificgames.com |
| output | Any random email id followed by scientificgames.com |

A screen shot of a computer code

AI-generated content may be incorrect.

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| operation\_type | NUMBER |
| value | Any integer or decimal value |
| output | Any random integer between 0 and Provided Value. |

A screen shot of a computer code

Description automatically generated

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| operation\_type | NUMBER\_IN\_RANGE |
| value | Any two integers or decimal values.  First integer should be less than second integer |
| output | Any random integer between the two numbers provided in value |

A screen shot of a computer

Description automatically generated

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| operation\_type | PHONE |
| value | Field Not Needed. |
| output | Any 10-digit number |

A screen shot of a computer code

Description automatically generated

## GETELEMENTCOUNT

**Mandatory Keywords:** Name, identifier, output

**Optional Keywords:** Desc, skip, identifierType, parent\_identifier

**Use:** When the locator is an array of elements, “GETELEMENTCOUNT” functions return the count of elements.

If parent\_identifier is provided, element count will be taken from the parent element. If no parent identifier provided, element count will be taken from the complete web page.

Values for the key words:

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| identifier | Path of the locator |
| parent\_identifier | Path of the parent locator under which element counts to be found out. |
| output | Stores count of elements in the variable provided |

{

"name": "Get the row count from a page",

       "operation": "GetElementCount",

       "identifier": "//\*[@id="CustomerGrid"]/div[3]/table/tbody/tr",

       "output": "${rowCount}"

},

{

      "name": "Get the row count from custoemer grid",

      "operation": "GetElementCount",

      "identifier": "//\*[@id="CustomerGrid"]/div[3]/table/tbody/tr",

      "parent\_identifier": "//\*[@id="CustomerGrid"]",

      "output": "${rowCount}"

}

## GETTEXT

**Mandatory Keywords:** Name, identifier

**Optional Keywords:** Desc, skip, identifierType, output, text, compare\_by

**Use:** Get the visible (i.e. not hidden by CSS) text of this element, including sub-elements.

Also compare with the key “Text” if provided and assert if fails.

**Use Case 1: When only “output” is provided**.

Values for the key words:

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| identifier | Path of the locator. |
| output | Stores extracted text in the variable provided for further use. |

A computer screen shot of text

Description automatically generated

**Use Case 2: When “text” & “compare\_by” is provided.**

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| identifier | Path of the locator. |
| Text | Text to be compared with getText |
| Compare\_by | Text to be compared with the getText using the specified operator.  If no operator is provided, value will be considered as equals. |
| output | Stores extracted text in the variable provided for further use. |
| Operation\_type | Soft\_assert |

When **Operation\_type** is “**soft\_assert**”, testcase will continue to execute even if the step fails. On completion of testcase, failed testcases will be reported. If no operation\_type is present, default is normal assert. Execution will be halted at the same step.



A black and orange text

AI-generated content may be incorrect.

A computer screen shot of text

AI-generated content may be incorrect.

## GETTITLE

**Mandatory Keywords:** Name, identifier, output

**Optional Keywords:** Desc, skip, identifierType

**Use:** Get the title of the current page.

Values for the key words:

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| identifier | Path of the locator |
| output | Stores extracted text in the variable provided |

A screen shot of a computer program

AI-generated content may be incorrect.

## GETURL

**Mandatory Keywords:** Name, identifier, output

**Optional Keywords:** Desc, skip, identifierType

**Use:** Get a string representing the current URL that the browser is looking at.

Values for the key words:

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| identifier | Path of the locator |
| output | Stores extracted text in the variable provided |

**A computer screen with text

Description automatically generated**

## HOVERMOUSE

**Mandatory Keywords:** Name, identifier

**Optional Keywords:** Desc, skip, identifierType

**Use:** Moves the mouse to the middle of the element.

Values for the key words:

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| identifier | Path of the locator |

## ISPRESENT

**Mandatory Keywords:** Name, identifier

**Optional Keywords:** Desc, skip, identifierType, output, value

**Use:** Is this element displayed or not? This method avoids the problem of having to parse an element's "style" attribute. This returns Boolean value to true or false. Automation framework asserts when element is not present, and output is not provided. If output is provided stores the value of True or False for further use.

Values for the key words:

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| identifier | Path of the locator |
| Value | True / False |
| output | Stores the value of true or false in the variable provided. |
| Operation\_type | Soft\_assert |

**When value tag is present**, compare if the element is present or not.

True → Check Element should be present and assert if it is not present.

False → Check Element should not be present and assert if it is present.

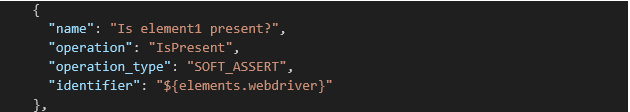
**When output tag is present**, Stores the value of true or false based on element present or not in the variable provided.

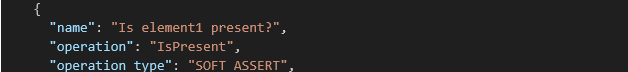
**When both Value & Output is not present**, compare if the element is present or not.

If not present→ Assert

If present → Go to next step

When **Operation\_type** is “**soft\_assert**”, testcase will continue to execute even if the step fails. On completion of testcase, failed testcases will be reported. If no operation\_type is present, default is normal assert. Execution will be halted at the same step.







## ISSELECTED

**Mandatory Keywords:** Name, identifier, output

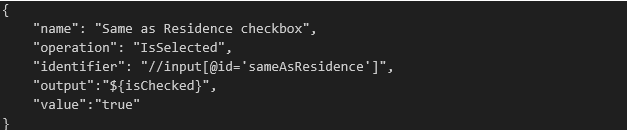
**Optional Keywords:** Desc, skip, identifierType

**Use:** Determine whether this element is selected or not. This operation only applies to input elements such as checkboxes, options in select and radio buttons. True if the element is currently selected or checked, false otherwise.

Values for the key words:

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| identifier | Path of the locator |
| output | Stores the value of true or false in the variable provided. |
| value | True or false |

\*\* Sprint 14 Onwards, if a value tag is provided, isSelected status of the element can be compared with the value provided and an assert can be thrown if it is not matching.



## LOAD

**Mandatory Keywords:** Name, value

**Optional Keywords:** Desc, skip

**Use:** Load a new web page in the current browser window.

Values for the key words:

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| value | URL of the page which needs to be loaded to browser |



## WAIT

**Mandatory Keywords:** Name, operation, identifier, value, Wait\_type, text

**Optional Keywords:** Desc, skip, identifier\_type, PollingTime (if the operation is FluentWait)

**Use:** This is the implementation of different wait types like Explicit Wait or Fluent Wait of selenium webdriver. Here automation waits for the element mentioned until the provided value.

Values for the key words:

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| operation | WAIT |
| identifier | Xpath of the element |
| Value | Waiting time. Default is 30 sec, if not provided. |
| Wait\_type | This is the condition used for Explicit or Fluent wait.  Default value is VISIBILITYOFELEMENTLOCATED, if no value is provided.  Please find the table below for the conditions that can be supplied for Wait\_types key. |
| Text | Default value is True. This key is used for the following wait-type   * invisibilityOfElementWithText * textToBePresentInElementLocated * textToBePresentInElementValue * titleIs * titleContains * elementSelectionStateToBe (True or False) |
| Polling\_Time | Interval at which automation will check for the element.  If Polling time is provided it will be Fluentwait. Default is 2sec.  If no polling time is provided it will be Explicitwait. |
| identifier\_type | Type of the locator |

|  |  |
| --- | --- |
| **Wait\_type** | **Usage** |
| visibilityofelementlocated | An expectation for checking that an element is present on the DOM of a page and visible. Visibility means that the element is not only displayed but also has a height and width that is greater than 0. |
| alertIsPresent\*\* | Wait till the alert is present |
| elementSelectionStateToBe | An expectation for checking if the given element is selected. |
| elementToBeClickable | An expectation for checking an element is visible and enabled such that you can click it. |
| elementToBeSelected | An expectation for checking if the given element is selected. |
| frameToBeAvaliableAndSwitchToIt | An expectation for checking whether the given frame is available to switch to. |
| invisibilityOfTheElementLocated | An expectation for checking that an element is either invisible or not present on the DOM. |
| invisibilityOfElementWithText | An expectation for checking that an element with text is either invisible or not present on the DOM. |
| presenceOfElementLocated | An expectation for checking that an element is present on the DOM of a page. |
| textToBePresentInElementLocated | An expectation for checking if the given text is present in the element that matches the given locator. |
| textToBePresentInElementValue | An expectation for checking if the given text is present in the specified elements value attribute. |
| visibilityOfAllElementsLocatedBy | An expectation for checking that all elements present on the web page that match the locator are visible. |
| visibilityOfAllElements | An expectation for checking that an element is present on the DOM of a page and visible. |
| titleIs\*\* | An expectation for checking the title of a page. The expected title must be an exact match |
| titleContains\*\* | An expectation for checking that the title contains a case-sensitive substring |

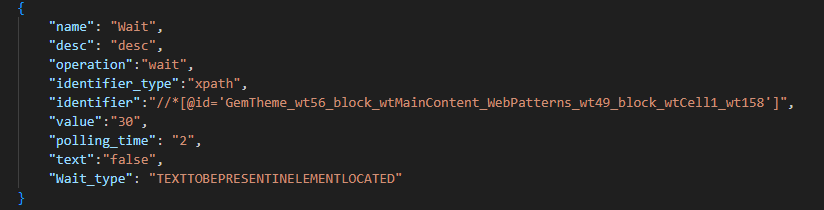
**Note:**

\*\* identifier\_type and identifier are not required for

* titleIs
* titleContains
* IsalertPresent

A computer screen with text

AI-generated content may be incorrect.



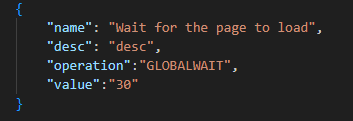
## GLOBALWAIT

**Mandatory Keywords:** Name, operation, value

**Optional Keywords:** Desc, skip

**Use:** This is the implicit wait of selenium webdriver. Here automation waits for elements till the mentioned time. Once the value is set, it will remain until the end of the script.

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| operation | GLOBALWAIT |
| Value | Waiting time. Default is 30 sec, if not provided. |



## GETARRAYSIZE

This is used to get the number of elements in an array. Please refer to [Appendix 5.1](#_Array) for more details.

## SLEEP

**Mandatory Keywords:** Name, operation, value

**Optional Keywords:** Desc, skip

**Use:** Sleep usages thread.sleep function of java program to halt the execution for provided time.

Halt the automation execution till specified time.

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| operation | SLEEP |
| Value | Waiting time. Default is 30 sec, if not provided. |

## STRINGMANIPULATION

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | STRINGMANIPULATION |
| operation\_type | LENGTH, CONCATENATE, TRIM, UPPERCASE, LOWERCASE, INDEX\_OF, CHAR\_AT, CONTAINS, ENDS\_WITH, STARTS\_WITH, SUBSTRING, REPLACE |
| value | Actual String where manipulation needs to be performed |
| text | Additional inputs |
| output | To store the output from the manipulation operation. |

***Note*** *– Since this is string operation any whitespace in value or text will not be trimmed automatically.*

**Illustration-**

Below table demonstrates with an example how this operation works.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **operation\_type** | **Value** | **Text** | **output** |  |
| LENGTH | Hello | NA | 5 |  |
| CONCATENATE | Hello | Word | HelloWorld |  |
| TRIM |  | NA |  | Removes the white space |
| UPPER\_CASE | Hello World | NA | HELLO WORLD | Convert everything to upper case |
| LOWER\_CASE | Hello World | NA | hello world | Convert everything to lower case |
| INDEX\_OF | Hello World | Wor or W | 7 | First occurrence of the word or char |
| CHAR\_AT | Hello World | 7 | W |  |
| CONTAINS | Hello World | Hell | TRUE |  |
| CONTAINS | Hello World | XXX | FALSE |  |
| ENDS\_WITH | Hello World | rld | TRUE |  |
| ENDS\_WITH | Hello World | XXX | FALSE |  |
| STARTS\_WITH | Hello World | Hell | TRUE |  |
| STARTS\_WITH | Hello World | XXX | FALSE |  |
| STARTS\_WITH | Hello World | Wor, 7 | TRUE | with offset - Where to start looking |
| SUBSTRING | StringMANIPULATION | 6 | MANIPULATION | Truncate the first 6 characters.  If only one number N is provided, substring between 1 and N is removed. Output is rest of the string. |
| SUBSTRING | StringMANIPULATION | 7,18 | MANIPULATION | Find the chars starting from 7th Index to 18th index |
| REPLACE | Hello World | (Hell, XXX) | XXXo World | Replace all occurrences |
| REPLACE | Hello World | Hell | o World | Replace all occurrences |
| REPLACE | Hello World | o | Hell Wrld | Replace all occurrences |
| REPLACE | Hello World | (o,XX) | HellXX WXXrld | Replace all occurrences |
| REPLACE | Hello World | (1,4,XXX) | XXXo World | Replace first 4 letters or Replace char at 1 to 4 |
| REPLACE | Hello World | (9,11,XXX) | Hello WoXXX | Replace char at 9 to 11 |
| REPLACE | Hello World | (o,XX,,first) | HellXX World | Replace first occurrence |
| REPLACE | Hello World | (o,,,first) | Hell World | Replace first occurrence |

## SPLIT

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | SPLIT |
| operation\_type | Not Applicable |
| value | Actual String where manipulation needs to be performed.  For example, String:5:MANIPULATION:TEST |
| text | Splitter, based on which split to be happened  For example : |
| output | To store the output from the manipulation operation. This should be an array type output.  For the above example an array of elements [String, 5, MANIPULATION, TEST] |

***Note –*** *Since this is string operation any whitespace in value or text will not be trimmed automatically.*

**Illustration-**

Below table demonstrates with example how this operation works.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operation** | **Value** | **Text** | **output** |  |
| Split | String:5:MANIPULATION:TEST | : | ${array.PlayerDetails} | Splits the string into multiple strings based on the separator (text) provided and stores into an array. |

## LOOP

**Mandatory Keywords:** : Name, operand\_lhs, operand\_rhs, compare\_by, loop

**Optional Keywords:** Desc, skip

**Use:** Used to create a loop like for loop or while loop

Values for the key words:

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | loop |
| operand\_lhs | Value of the left-hand side operand for comparison. |
| operand\_rhs | Value of the right-hand side operand for comparison. |
| compare\_by | EQUALS, NOT\_EQUALS, CONTAINS, STARTS\_WITH, ENDS\_WITH, EQUALS\_IGNORE\_CASE, LESS\_THAN, LESS\_THAN\_EQUALS, GREATER\_THAN GREATER\_THAN\_EQUALS |
| Loop | Step or multiple steps |

A loop keyword above may have multiple steps inside and one of the mandatory steps to have loopindexupdate. Below explain how loopindexupdate to be formed.

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Operation | loopindexupdate |
| operation\_type | INCREMENT or DECREMENT |
| Value | Value by which want to update the loop condition |
| output | Mention whichever operand wanted to be modified |

Loop index update should have one of the operand variables for which the value should be incremented or decremented to compare and which order value has to be incremented or decremented.



## SCROLL

Scroll will simply scroll the page or element for once. Automation testers need to use additional logic to find or do require operation post scroll. This is not recommended unless there is any difficulty scrolling the page using Scroll to element operation.

**Scroll For Mobile**

This is applicable only for mobiles. For Web value needs a comma separated input with 2 or 4 parameters.

**Mandatory Keywords:** Name, Operation, Operation\_type, value

**Optional Keywords:** Desc, skip

**Use:** Used to scroll a page

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | scroll |
| Operation\_type | Up, down, left, right |
| Value | Numeric value to determine scroll percent |

A black screen with white text

AI-generated content may be incorrect.

**Scroll for Web**

Scrolling a web page is controlled by selenium function scrollFromOrigin. The starting point of scroll or the origin is decided by the first two parameters in the value tag. Next two parameters are the delta to do the scroll.

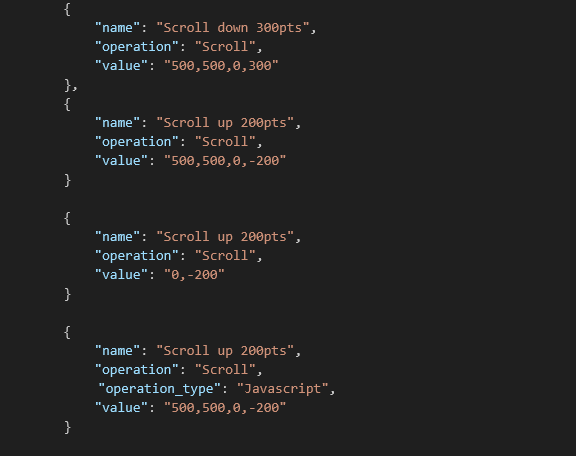
For web scrolling, operation\_type: should be JAVASCRIPT or should not be specified.

**Parameters:**

**scrollOrigin** Where scroll originates (viewport or element center) plus provided offsets.

**deltaX** The distance along X axis to scroll using the wheel. A negative value scrolls left.

**deltaY** The distance along Y axis to scroll using the wheel. A negative value scrolls up.



## SCROLLTOELEMENT

**Scroll To Element for Mobile:**

Scroll to element on the other hand uses the supplied element (identifier) and scroll the page until the element is found. Scrolls happen at the center of the page and amount of scroll can be controlled by using the percentage value from value tag.

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| name | Name of the step |
| operation | ScrollToElement |
| identifier | Path of the locator to be scrolled into the view port |
| parent\_identifier | Parent\_identifier is the element within which scroll needs to happen. This is an optional element. If provided, scroll will happen at the center of the frame. |
| operation\_type | Up, down, left, right |
| value | Max scroll before Exception is thrown |
| count | Max number of scrolls should be made before throwing error. If not provided then the value of maxium\_scroll\_counts key from project.properties will be used. |

A screen shot of a computer program

AI-generated content may be incorrect.

If none of the scroll is working, (X,Y) coordinates and pixel values can be passed in value tag. So that the scroll will start from the (X,Y) coordinates and automation will scroll up to the amount specified as 3rd parameter.

A screenshot of a computer code

AI-generated content may be incorrect.

**Scroll To Element for Web**

Recommended way to scroll and find an element:

Same ScrollToElement operation can also be used for scrolling any web page. Use the **wait** conditions for the page to be loaded properly before calling the scroll operation. This method uses selenium scrollToElement function and If the element is outside the viewport, scrolls the bottom of the element to the bottom of the viewport.

A screen shot of a computer program

AI-generated content may be incorrect.

Other ways to scroll if the above method is not working.

There is another way to scroll a page and find an element.

A parent identifier can be passed along with the starting point and delta to scroll. This can scroll within a web page.

A computer screen shot of a program

AI-generated content may be incorrect.

**Scrolling web page using javascript**

Some web pages having difficulty can be scrolled using javascript. The operation\_type parameter in the above operation needs to be set to javascript to use this. Scrolling value is controlled by the delta provided as part of value.

**deltaX:** The distance along X axis to scroll using the wheel. A negative value scrolls left.

**deltaY**: The distance along Y axis to scroll using the wheel. A negative value scrolls up.

A screen shot of a computer code

AI-generated content may be incorrect.

\*\* Testing in progress for javascript scroll.

## SWITCHFRAME

**Mandatory Keywords:** Name, Operation, identifier

**Optional Keywords:** : Desc, identifier\_type, skip

**Use:** Select a frame using its previously located WebElement. In return, the driver focused on the given frame.

[NoSuchFrameException](eclipse-javadoc:%E2%98%82=jsonframework/C:%5C/Users%5C/manoj.mohanty%5C/.m2%5C/repository%5C/org%5C/seleniumhq%5C/selenium%5C/selenium-api%5C/4.16.1%5C/selenium-api-4.16.1.jar=/maven.pomderived=/true=/=/maven.pomderived=/true=/=/javadoc_location=/jar:file:%5C/C:%5C/Users%5C/manoj.mohanty%5C/.m2%5C/repository%5C/org%5C/seleniumhq%5C/selenium%5C/selenium-api%5C/4.16.1%5C/selenium-api-4.16.1-javadoc.jar%5C!%5C/=/=/maven.groupId=/org.seleniumhq.selenium=/=/maven.artifactId=/selenium-api=/=/maven.version=/4.16.1=/=/maven.scope=/compile=/%3Corg.openqa.selenium(WebDriver$TargetLocator.class%E2%98%83TargetLocator~frame~Lorg.openqa.selenium.WebElement;%E2%98%82NoSuchFrameException) - If the given element is neither an IFRAME nor a FRAME element.

[StaleElementReferenceException](eclipse-javadoc:%E2%98%82=jsonframework/C:%5C/Users%5C/manoj.mohanty%5C/.m2%5C/repository%5C/org%5C/seleniumhq%5C/selenium%5C/selenium-api%5C/4.16.1%5C/selenium-api-4.16.1.jar=/maven.pomderived=/true=/=/maven.pomderived=/true=/=/javadoc_location=/jar:file:%5C/C:%5C/Users%5C/manoj.mohanty%5C/.m2%5C/repository%5C/org%5C/seleniumhq%5C/selenium%5C/selenium-api%5C/4.16.1%5C/selenium-api-4.16.1-javadoc.jar%5C!%5C/=/=/maven.groupId=/org.seleniumhq.selenium=/=/maven.artifactId=/selenium-api=/=/maven.version=/4.16.1=/=/maven.scope=/compile=/%3Corg.openqa.selenium(WebDriver$TargetLocator.class%E2%98%83TargetLocator~frame~Lorg.openqa.selenium.WebElement;%E2%98%82StaleElementReferenceException) - If the WebElement has gone stale.

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | SWITCHFRAME |
| identifier | frameElement |

## Selecting a value in a locator

### SELECTBYINDEX

**Mandatory Keywords:** Name, Operation, identifier, value

**Optional Keywords:** Desc, identifier\_type, skip

**Use:** Select an option from a dropdown or list by using index.

Select the option at the given index. This is done by examining the "index" attribute of an element, and not merely by counting.

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | SELECTBYINDEX |
| identifier | Locator to drop down |
| Value | Index value to be selected. |

### SELECTBYTEXT

**Mandatory Keywords:** Name, Operation, identifier, value

**Optional Keywords:** Desc, identifier\_type, skip

**Use:** Select an option from a dropdown or list by using text.

Select all options that display text matching the argument. That is, when given "Bar" this would select an option like: <option value="foo">Bar</option>

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | SELECTBYTEXT |
| identifier | Locator to drop down |
| Value | Text to be selected. |

### SELECTBYVALUE

**Mandatory Keywords:** Name, Operation, identifier, value

**Optional Keywords:** Desc, identifier\_type, skip

**Use:** Select an option from a dropdown or list by using value.

Select all options that have a value matching the argument. That is, when given "foo" this would select an option like: <option value="foo">Bar</option>

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | SELECTBYVALUE |
| identifier | Locator to drop down |
| Value | Text to be selected. |

## SENDSPECIALKEY

**Mandatory Keywords:** Name, Operation, identifier, value

**Optional Keywords:** Desc, identifier\_type, skip

**Use:** Use this method to simulate typing special keys. To get the list of special keys use the link below:

<https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/Keys.html>

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | SENDSPECIALKEY |
| identifier | Locator to where scrolling needed |
| Value | Value of the Key like ENTER or TAB |

A computer screen with text

Description automatically generated

## SETPROPERTY

**Use:** Set property is used to set any value for a variable.

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | SetProperty |
| output | Variable which needs to be set |
| Value | Value what needs to be set |

Example:

A black screen with orange text

AI-generated content may be incorrect.

## Mobile Specific Tags

1. Mobile locators
2. SWITCHCONTEXT
3. SWITCHTONATIVEAPP
4. SWITCHTOPARENTFRAME
5. SWITCHTOWEBVIEW
6. LISTCONTEXT

**Mandatory Keywords:** Name, identifier, output

**Optional Keywords:** Desc, skip, identifierType

**Use:** This operation is allowed only for Mobile drivers. Get the names of available contexts.

Values for the key words:

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| identifier | Path of the locator |
| output | Stores the value of true or false in the variable provided. |

## TYPE

**Mandatory Keywords:** Name, Operation, identifier, value

**Optional Keywords:** Desc, identifier\_type, skip

**Use:** Method to type text to the TextBox or TextArea or File Upload.

For uploading file, the control/identifier should be of file type.

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | TYPE |
| identifier | Locator to where scrolling needed |
| Value | Text to be typed/ path of the file to be uploaded |

A screen shot of a computer

AI-generated content may be incorrect.

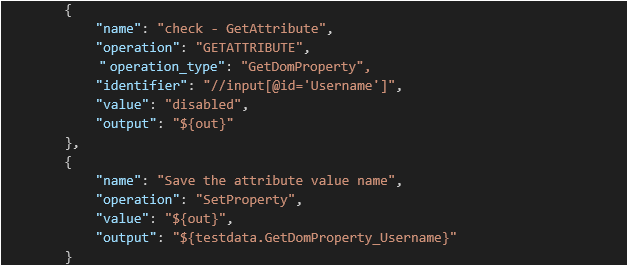
## GETATTRIBUTE

**Mandatory Keywords:** Name, Operation, identifier, value, output

**Optional Keywords:** Desc, identifier\_type, skip, text

**Use:** This method is used to get the attributes of an elements to validate some of the usecases like if the element is disabled or enabled etc.

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | GetAttribute |
| identifier | Locator to where scrolling needed |
| Value | Which type of attribute is needed. |
| Output | To store the attribute value |
| operation\_type | when text is  GET\_DOM\_ATTRIBUTE → Call selenium 4 GetDomAttribute method.  GET\_DOM\_PROPERTY → Call selenium 4 GetDomProperty method.  GET\_ATTRIBUTE → Call selenium 3 GetAttribute method.  JAVASCRIPT → Call JavascriptExecutor getAttribute method.  GET\_CSS\_VALUE -> Call selenium getCssValue method.  Note – Default value is Javascript . If there is no value provided, javascript functionality will be called. |

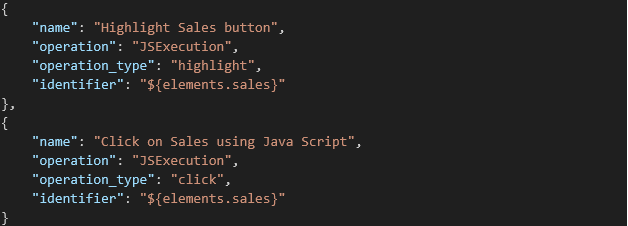


## JAVAScript Execution

**Mandatory Keywords:** Name, Operation, identifier, operation\_type

**Optional Keywords:** Desc, identifier\_type, skip

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | JSExecution |
| identifier | Locator to where scrolling needed |
| Operation\_type | when operation\_type is  Click → Click on the element through Javascript instead of usual selenium click method.  Type → Type on the element through Javascript instead of usual selenium type method.  HIGHLIGHT→ Highlight the element with red border and yellow background. This is used for debugging purposes. There is no clear option for this. |
| Value | Pass the value for “type” operation\_type |



## Handling multiple Windows

### GETWINDOWHANDLES

**Mandatory Keywords:** Name, Operation, output

**Optional Keywords:** Desc, skip

**Use:** Stores all handles in an array

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | GetWindowHandles |
| output | ${arraylist.windows} à to store the handle in an array. |

### GETCURRENTWINDOWHANDLE

**Mandatory Keywords:** Name, Operation, output

**Optional Keywords:** Desc, skip

**Use:** Stores the current window handle in a variable. Can be used for validation purposes.

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | GetCurrentWindowHandle |
| output | ${handle} à To store the handle in a variable. |

### SWITCHTOWINDOWHANDLE

**Mandatory Keywords:** Name, Operation, value

**Optional Keywords:** Desc, skip

**Use:** Switch to a new tab based on the provided value

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | GetCurrentWindowHandle |
| value | ${arraylist.windows#1} à Provide which handle in the array to switch |

A computer screen with text

AI-generated content may be incorrect.

A computer screen with text

AI-generated content may be incorrect.

## Date Functions

### DATEOPERATIONS

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | DateOperations |
| operation\_type | Refer to the below table |
| date\_format |
| value |
| Date\_field |
| output | Variable name |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| date | operation\_type | date\_field | Value | date\_format (Examples) | output |
| When null, current date. Else supplied date. | GET\_CURRENT\_DATE\_TIME | NA | NA | dd-MM-yyyy HH:mm:ss |  |
|  | GET\_DAY | NA | NA | dd-MM-yyyy |  |
|  | GET\_MONTH\_NUMBER | NA | NA | yyyy hh:mm:ss a |  |
|  | GET\_MONTH\_PREFIX | NA | NA | hh:mm:ss a |  |
|  | GET\_MONTH\_NAME | NA | NA | Any of above |  |
|  | GET\_YEAR | NA | NA | Any of above |  |
|  | GET\_SHORT\_YEAR | NA | NA | Any of above |  |
|  | GET\_HOUR | NA | NA | Any of above |  |
|  | GET\_HOUR\_24 | NA | NA | Any of above |  |
|  | GET\_MINUTE | NA | NA | Any of above |  |
|  | GET\_SECOND | NA | NA | Any of above |  |
|  | GET\_AM\_PM | NA | NA | Any of above |  |
|  | GET\_PAST\_OR\_FUTURE\_DATE | when null,  default Days | when null, default 1 | Any of above |  |
|  | GET\_PAST\_OR\_FUTURE\_DATE | YEARS | "+ve or -ve integer" | Any of above |  |
|  |  | MONTHS | "+ve or -ve integer" | Any of above |  |
|  |  | DAYS | "+ve or -ve integer" | Any of above |  |
|  |  | HOURS | "+ve or -ve integer" | Any of above |  |
|  |  | MINUTES | "+ve or -ve integer" | Any of above |  |
|  |  | SECONDS | "+ve or -ve integer" | Any of above |  |

Date formats (date\_format) are for example. While providing the ‘date’, user must use the date\_format.

A computer screen shot of a program

AI-generated content may be incorrect.

### DateOperation2

Additional Date Operation can perform 3 sub operations.

1. COMPARE\_DATE – To compare two dates or time with a compare\_by condition.
2. CONVERT\_DATE\_FORMAT : To convert the date from one format to another format.
3. GET\_DATE\_TIME\_DIFFERENCE : To get the time difference between two date or time.

#### COMPARE\_DATE

This function compares two dates and helps in either asserting a testcase or provides the output for further evaluation. In absence of “output” tag, code will assert if the comparison is false. If output tag is provided, comparison output either “true” or “false” will be stored in the provided variable.

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | DateOperation2 |
| Operation\_type | COMPARE\_DATE |
| operand\_lhs | First date input |
| operand\_rhs | Second date input |
| compare\_by | EQUALS, NOT\_EQUALS, LESS\_THAN, GREATER\_THAN, LESS\_THAN\_EQUALS,  GREATER\_THAN\_EQUALS |
| date\_format | Current date format  How to form valid formats and example of some of the valid date formats available below. |
| output | Variable name |

1. How to form valid formats:

|  |  |  |  |
| --- | --- | --- | --- |
| **Letter** | **Date or Time Component** | **Presentation** | **Examples** |
| G | Era designator | [Text](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html#text) | AD |
| y | Year | [Year](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html#year) | 1996; 96 |
| Y | Week year | [Year](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html#year) | 2009; 09 |
| M | Month in year (context sensitive) | [Month](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html#month) | July; Jul; 07 |
| L | Month in year (standalone form) | [Month](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html#month) | July; Jul; 07 |
| w | Week in year | [Number](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html#number) | 27 |
| W | Week in month | [Number](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html#number) | 2 |
| D | Day in year | [Number](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html#number) | 189 |
| d | Day in month | [Number](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html#number) | 10 |
| F | Day of week in month | [Number](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html#number) | 2 |
| E | Day name in week | [Text](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html#text) | Tuesday; Tue |
| u | Day number of week (1 = Monday, ..., 7 = Sunday) | [Number](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html#number) | 1 |
| a | Am/pm marker | [Text](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html#text) | PM |
| H | Hour in day (0-23) | [Number](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html#number) | 0 |
| k | Hour in day (1-24) | [Number](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html#number) | 24 |
| K | Hour in am/pm (0-11) | [Number](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html#number) | 0 |
| h | Hour in am/pm (1-12) | [Number](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html#number) | 12 |
| m | Minute in hour | [Number](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html#number) | 30 |
| s | Second in minute | [Number](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html#number) | 55 |
| S | Millisecond | [Number](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html#number) | 978 |
| z | Time zone | [General time zone](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html#timezone) | Pacific Standard Time; PST; GMT-08:00 |
| Z | Time zone | [RFC 822 time zone](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html#rfc822timezone) | -0800 |
| X | Time zone | [ISO 8601 time zone](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/text/SimpleDateFormat.html#iso8601timezone) | -08; -0800; -08:00 |

1. **Example of Valid date formats**

|  |  |
| --- | --- |
| **Date and Time Pattern** | **Result** |
| "yyyy.MM.dd G 'at' HH:mm:ss z" | 2001.07.04 AD at 12:08:56 PDT |
| "EEE, MMM d, ''yy" | Wed, Jul 4, '01 |
| "h:mm a" | 12:08 PM |
| "hh 'o''clock' a, zzzz" | 12 o'clock PM, Pacific Daylight Time |
| "K:mm a, z" | 0:08 PM, PDT |
| "yyyyy.MMMMM.dd GGG hh:mm aaa" | 02001.July.04 AD 12:08 PM |
| "EEE, d MMM yyyy HH:mm:ss Z" | Wed, 4 Jul 2001 12:08:56 -0700 |
| "yyMMddHHmmssZ" | 010704120856-0700 |
| "yyyy-MM-dd'T'HH:mm:ss.SSSZ" | 2001-07-04T12:08:56.235-0700 |
| "yyyy-MM-dd'T'HH:mm:ss.SSSXXX" | 2001-07-04T12:08:56.235-07:00 |
| "YYYY-'W'ww-u" | 2001-W27-3 |

1. **Example of Script**

A screenshot of a computer screen

AI-generated content may be incorrect.

#### CONVERT\_DTAE\_FORMAT

|  |  |
| --- | --- |
| Keyword | Values |
| Name | Name of the step |
| Operation | DateOperation2 |
| Operation\_type | CONVERT\_DATE\_FORMAT |
| operand\_lhs | First date input |
| operand\_rhs | Required date format to which date needs to be converted |
| date\_format | Current date format |
| output | Variable name where output to be stored. |

A computer screen shot of a code

AI-generated content may be incorrect.

#### GET\_DATE\_TIME\_DIFFERENCE

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | AdvDateOperation |
| Operation\_type | CONVERT\_DATE\_FORMAT |
| operand\_lhs | First date input |
| operand\_rhs | 2nd Date input |
| date\_format | Current date format |
| output | Variable name where output to be stored. |
| date\_field | How the difference in time to be presented. |

A screen shot of a computer code

AI-generated content may be incorrect.

## Browser Operations

### NAVIGATE BACK

Operation to navigate to the back page in browser.

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | BrowserOperation |
| Operation\_Type | Navigate\_Back |

### NAVIGATE FRONT

Operation to navigate to the front page in browser.

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | BrowserOperation |
| Operation\_Type | Navigate\_Front |

### REFRESH

Operation to refresh the page in browser.

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | BrowserOperation |
| Operation\_Type | Refresh |

## Get Element Size

### GET WIDTH

Operation to get the element width.

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | GetElementSize |
| Operation\_Type | Get\_Width |
| Identifier | Identifier of the element |
| Output | Variable in which the value needs to be stored |

### GET HEIGHT

Operation to get the element height.

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | GetElementSize |
| Operation\_Type | Get\_Height |
| Identifier | Identifier of the element |
| Output | Variable in which the value needs to be stored |

# Appendix 1

## Array

This section explains how to use an array for automating testcases. Array is used to store and retrieve data from a group of strings. There is no specific keyword for this except an operation to get the array size and sorting the array.

**Syntax to declare an array:**

**A black screen with orange text

AI-generated content may be incorrect.**

Here PlayerDetails is the array name. The two array elements are the email id and password.

**To get the count of the elements of an array**

Keyword: **GetArraySize**

**A black screen with orange text

AI-generated content may be incorrect.**

**Adding an element to array**

Keyword: **SetProperty**

Adding a new element to array:

A black screen with orange text

AI-generated content may be incorrect.

Updating an existing element:

A black screen with orange text

AI-generated content may be incorrect.

**Getting an element from the array**

Use the syntax - "${array.PlayerDetails#<index number>}",

The index value of an element in an array starts with **0**.

A screen shot of a computer

AI-generated content may be incorrect.

A computer screen shot of a black background

AI-generated content may be incorrect.

**Printing an Array**

Keyword: **SetProperty**

A black background with white text

AI-generated content may be incorrect.

It will store the value in string as ["mukesh.muthupandi@scientificgames.com", "Password123!@#"]

**Sorting an Array Elements [Lexicographically]**

You can sort an array element using the operation ArraySort.

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | ArraySort |
| Operation\_Type | ascending, descending |
| value | Array property needs to sort |
| Output | New Array property where the sorted array needs to be saved |

A black screen with text

AI-generated content may be incorrect.

## Rank

Rank for a testcase can be defined within the testcase. For example:

A black screen with orange text

AI-generated content may be incorrect.

Similarly, rank for a folder can be assigned by creating a module.properties file inside the folder and by mentioning the rank inside the module.properties.

For example:

A screenshot of a computer

AI-generated content may be incorrect.

Let’s say for the FirstTest.json, rank of the testcase is 2 and inside module.property we mention rank=3

So overall rank for the file will be 3.2.

If a folder has no rank, parent folder rank will be considered.

If there is no rank at all either the file or the folder, those files or folders will be considered as highest rank. Execution order for such files could not be guaranteed.

Tips for Testers:

Testcases which need to be run sequentially, should be ranked by folder or testcase level. Where sequence is not mandatory, can be left blank.

Folders rank should be set first and then individual testcase rank if needed. This will make the maintenance easier and inserting scripts in between will be easier.

# APPENDIX 2

## KEY VALUE MAPPING

[KeyValueMappings.xlsx](https://scientificgames.sharepoint.com/:x:/t/AutomationFramework-Flexicode/ER9qG2bchsBMqzGg-CQE4-gBlNiwXf0askFmuJ8_FR72Uw?e=pAcVGn)

## DATABASE

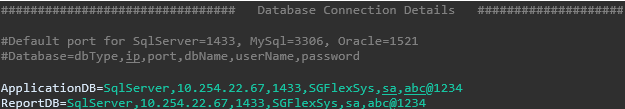
This section explains how to set up database connection for automating testcases.

Database connection String setup:

1. Open the testdata.properties file.
2. Add database-specific details in the following format:



For Example:



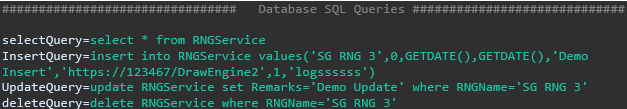
1. You can add multiple database connection strings by repeating the format for each database.
2. Save the testdata.properties file.
3. In case the Instance Name is available and not the port number then get the port number by running the below SQL query after connecting to the DB via SSMS:-

**Query to get Port Number**: select distinct local\_net\_address, local\_tcp\_port from sys.dm\_exec\_connections where local\_net\_address is not null

**Database Query setup:**

Framework support Select, Insert, Update and Delete queries execution in MS SQL Server/ MySQL/ Oracle. Add SQL queries as per the below format in testdata.properties or directly in the testcase JSON files.

For Example:



Before incorporating SQL queries into your automation framework, ensure the following points:

* **Verify in Database IDE:**
  + Execute each SQL Execute each SQL query in the respective Database IDE (MS SQL Server, MySQL, Oracle) to ensure it performs as expected.
  + Confirm that the queries produce the desired results and do not cause unintended side effects.
* **Use Proven Queries in Automation:**
  + Only use queries in your automation scripts that have been thoroughly tested and verified in the Database IDE.
  + Avoid using untested queries directly in automation to prevent unexpected behavior during script execution.

This practice ensures the reliability and accuracy of your database interactions within the automation framework, reducing the risk of errors and enhancing the overall stability of your test scripts

* **Executing Queries and Obtaining Output:**

Once the connection string and the query statement are available, use the below methods to execute the query and get desired output.

Keyword: **RunDBQuery**

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | **RunDBQuery** |
| db\_connection\_name | Provide the connection string against which the query must be run. |
| value | Provide the actual query. |
| Output | Store the output of the query. Use mapList to store multiple rows and columns of result. |

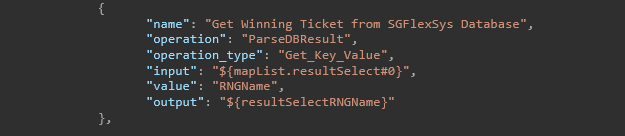
A screen shot of a computer code

AI-generated content may be incorrect.

* **Retrieving Result for a Specific Column and Row:**

Keyword: **ParseDBResult**

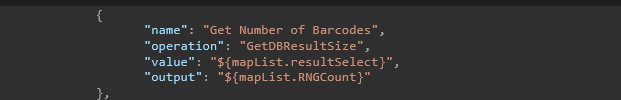
|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | **ParseDBResult** |
| Operation\_type | **Get\_Key\_Value** |
| input | * Input should be the variable where result is stored with an index.   Ensure that the Index in the variable corresponds to the desired row number.  For example - "${mapList.resultSelect#0}"  In the variable "${mapList.resultSelect#0}", the index value '0' refers to the record in a **mapList**. It's important to note that the index values for **mapList** start with 0. This means that the first record in the **mapList** is accessed using index 0, the second record with index 1, and so on. |
| Value | Column name for which data has to be extracted. |
| Output | Final output can be stored in a variable. |



* **Retrieving the Result Size**

Keyword: **GetDBResultSize**

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Name | Name of the step |
| Operation | **GetDBResultSize** |
| Value | should be the variable where result is stored |
| Output | Final output can be stored in a variable. |



## REST API

This section explains how to set up Rest API for automating testcases.

* **API URL SETUP:**

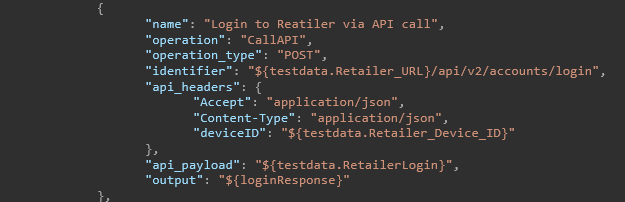
1. Open the testdata.properties file.
2. Add REST API Details as shown in the format below
3. Here RetailerLogin and CreateTicket are two separate Rest API requests.

A screenshot of a computer program

AI-generated content may be incorrect.

* **Executing POST method in Rest API request:**

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| name | Name of the step |
| operation | **CallAPI** |
| operation\_type | **POST,** **GET, PUT**  Below example is for POST operation. |
| identifier | Rest API URI |
| api\_header | Should include the headers required for the API call. |
| api\_payload | Rest API Request Body. This can be read from the testdata. |
| output | Variable can be used to store the response. |

****

* **Retrieving response status code:**

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| name | Name of the step |
| operation | **ParseAPIResponse** |
| operation\_type | **Get\_Status\_Code** à To get the status code from the response |
| input | Rest API Response  For example - "${loginResponse}" from above API call |
| Output | Output variable where result to be stored. |

A screen shot of a computer screen

AI-generated content may be incorrect.

* **Retrieving a key value from the response:**

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Operation | **ParseAPIResponse** |
| operation\_type | **Get\_Key\_Value** à To get any value from a particular key from the response. |
| input | Rest API Response  For example, "${loginResponse}" from above API call |
| value | Key whose value needs to be read.  e.g. "token" |
| output | Output variable where result to be stored. |

A screen shot of a computer code

AI-generated content may be incorrect.

* **Executing** **GET method in Rest API request:**

|  |  |
| --- | --- |
| **Keyword** | **Values** |
| Operation | **CallAPI** |
| operation\_type | **POST** or **GET**  Below example is for GET operation. |
| input |  |
| identifier | Rest API URI |
| api\_params | Add all params as key and value maps |
| api\_headers |  |
| api\_payload |  |
| output |  |

**A screen shot of a computer program

AI-generated content may be incorrect.**

## Working with QTEST

This section outlines the procedures for setting up an automation project and integrating it with qTest to seamlessly upload the results to the designated qTest project.

### Configuring the Project Property to Connect with qTest

**Updating “project. property” file**

**A black screen with orange text

AI-generated content may be incorrect.**

**link\_qtest**: Set this to true if you want the results to be updated in qTest.

**project\_id**: This component is essential for the qTest integration. Upon logging in to qTest, navigate to your project. The project ID is identified by the number following "P/" in the URL. For instance, in the provided link, the project ID is 53.

<https://sgl.qtestnet.com/p/53/portal/project>

**end\_point:** End point remains same across all SG projects. https://sgl.qtestnet.com

**authorization:**

Here are the steps to locate the Bearer token for Authorization:

* Log in to your qTest account.
* Navigate to the top right corner menu bar and click on the down arrow button.

A white and blue screen with a question mark and a gear

AI-generated content may be incorrect.

* From the listed items, select the "API & SDK" section.

A screenshot of a computer

Description automatically generated

Locate and copy the Bearer Token provided in this section.

**test\_cycle\_id:**

Within the Test Execution module, choose a specific test cycle. You can identify the Test Cycle ID from the URL. For instance: in the below URL, test\_cycle\_id is 3175

https://sgl.qtestnet.com/p/53/portal/project#tab=testexecution&object=4&id=3175

If the test\_cycle\_id is not provided, the automation process will proceed as usual, but the results won't be uploaded to the qTest project.

**test\_suite\_id**

If automation tester wants to rerun the testcases and update to same suite which they did earlier, they can provide a test\_suite\_id. If not provided, a new test\_suite will be created. Also, if the provided test\_suite\_id is not inside the test\_cycle\_id, a new test\_suite will be created.

To find the test\_suite\_id, select the test suite in qTest and find the id from the URL. The URL will be similar to:

<https://sgl.qtestnet.com/p/77/portal/project#id=14536&object=2&tab=testexecution>

Here the test\_suite\_id is 14536.

### Configuring the testcase

**Inside Testcase Json File:**

Supply the qTest testcase IDs to the automation script. If the script encompasses automation for multiple test cases, list the testcase IDs as comma-separated values.

For clearer readability and organization, it's recommended to assign the same testcase name to the automation script as it appears in qTest.

To obtain testCaseIds, follow these steps:

* Navigate to Test Design.
* Select a specific testcase.
* The testcase ID will be visible in the URL.
* For example, in the below URL, testcase id is 34433

<https://sgl.qtestnet.com/p/53/portal/project#tab=testdesign&object=1&id=34433>

A black background with white text

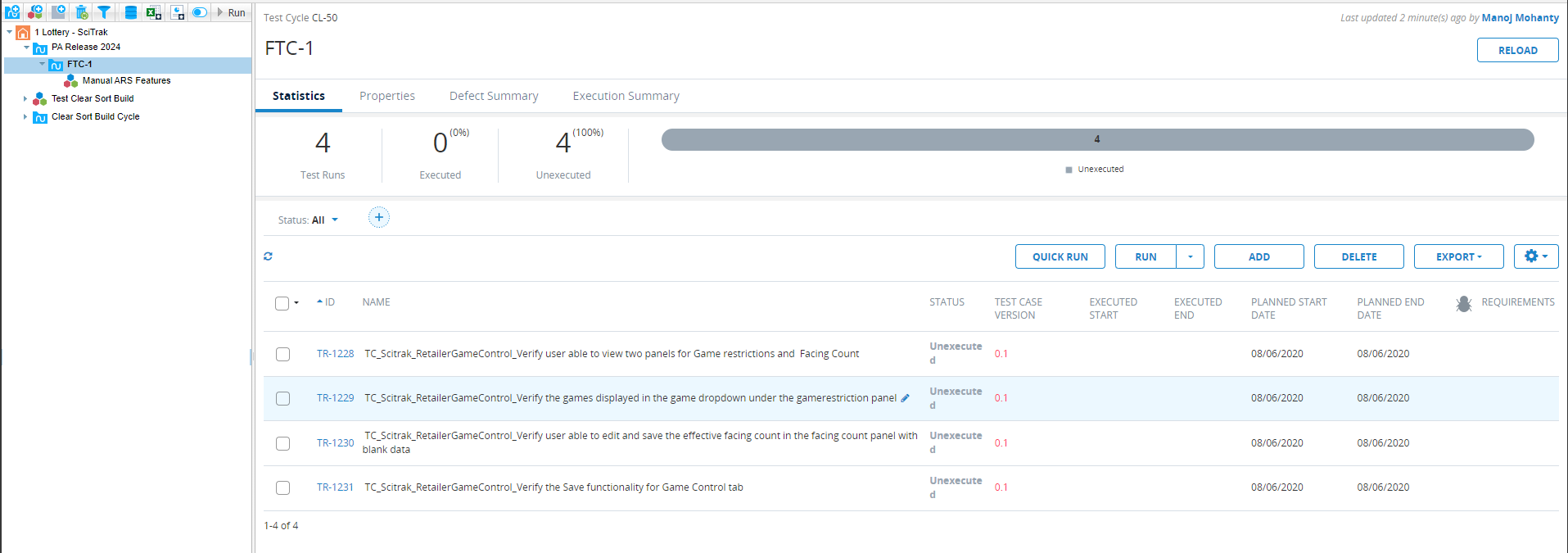
AI-generated content may be incorrect.

Please Note:

* Scripts not provided with testCaseIds will still be executed but the result will not be updated to qTest.
* Testcases not approved in qTest will not be added to test execution.

### Creating Test plan in qTest Project

Testers should continue to create the test plans in qTest project using the traditional method employed during the manual testing. All the testcases automated should be removed from the Test plan initially.



**Figure: Manual Test Plan**

Depending upon project configuration, Type or Automated? field in qTest project for the testcase can be used to segregate the testcases to either manual or automated. Contact your qTest project admin in order to decide which filed can be used to segregate the automated testcases.

A screenshot of a computer

Description automatically generated

**Figure: Fields for Marking Test Case Status**

A screenshot of a computer

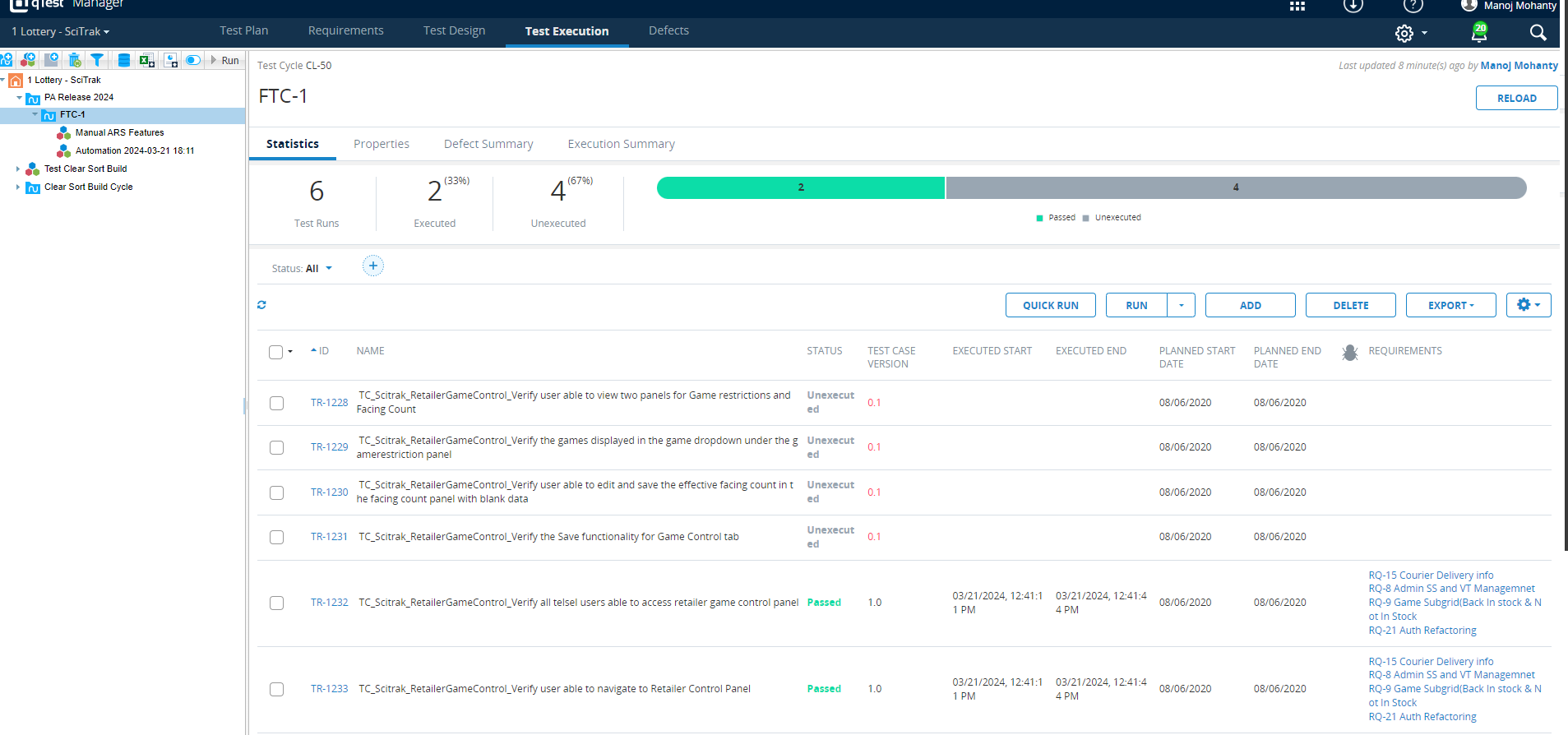
Description automatically generated

**Figure: Fields for Marking Test Case Status**

While automation runs in parallel, testers should continue their manual testcases as usual.

### Creating Test Suite in qTest for Automation Results

On completion of automation, script will create a test suite with current date and time stamp and update the results to the provided test cycle. Test suite name will be of the format “Automation YYYY-MM-DD HH:MM”. For example – “Automation 2024-03-21 16:43”



**Figure: Final Test Plan after Automation Completion**

Automation test completion report will also display the Test Cycle and Test Suite name for easier reference.

A screenshot of a computer

Description automatically generated

**Figure: Automation Report**

## RUNNING AUTOMATION JAR WITH EXTRA PARAMETERS

### Rerun of failure or skipped Testcases

On completion of first round of automation, if testers want to rerun the only failed testcases, they can use the command below.

Java -jar SGAutoPro<version>.jar --rerun failed

Similarly On completion of first round of automation, if testers want to rerun the only skipped testcases, they can use the below command.

Java -jar SGAutoPro<version>.jar --rerun skipped

**Please Note:** Framework reads the failed or skipped testcases from the file testng-results.xml created by TestNG while running the automation. If testers want to rerun the same set of testcases multiple times for their debugging purpose, they can save and replace the file before each run.

testng-results.xml file will be created under the folder “test-output” inside root folder.

### Running automation using extra parameters

From sprint 16 onwards, automation can be run from using extra parameters from runtime. Below are some of the commands supported.

**Driver**

|  |  |  |
| --- | --- | --- |
| **Usage** | **Command** | **Permitted values** |
| Driver | --driver | As per the values mentioned in [Project configuration](#_Project_Configurations) -> driver\_choice |
| Tags to run | --TagsToRun | As per the tags created in testscripts |
| UDID Value | --UDID | UDID value of the mobile |
| Appium Server Value | --AppiumServer | Appium server value |
| Appium port Value | --AppiumPort | Appium port value |
| Rerun of failed or skipped Testcases | --ReRun | Failed or skipped |

For example, if you want to run your testcases on chrome browser use the command below

Java -jar SGAutoPro<version>.jar --driver chrome\_browser

### Preparing data for iLottery CICD

This section is currently applicable for the Jenkins server configured for iLottery project automation in India.

After completion of testing, to prepare the data for Jenkins, use the command.

Java -jar SGAutoPro<version>.jar –CIDataProcess true

This command must be configured in Jenkins, and this will copy the data available under “config\environment.properties” and the test data from “test-output\testng-results.xml” to prepare the “config\ jenkin.properties” file.

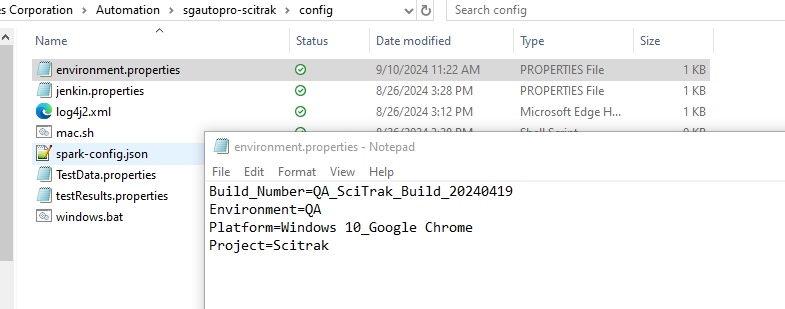
For Jenkin to report the data, these 4 keys must be there in environment.properties.

Project=

Platform=

Build\_Number=

environment=



**Please note:**

environment.properties & jenkin.properties should also be available in git with all the required parameters. jenkin.properties should have default values and will be updated during runtime. environment.properties must have above 4 keys with values as required in the Jenkins report.

This command will also copy the test result files from test server to the web server, if appropriate windows.bat or mac.sh file is available in the config folder with required commands to copy the files.

The IIS server link will be set as per the iisServer config available in project.properties file.

## CHECK DUPLICATE KEYS

This section is for automation script developers. While the developers are developing scripts for the project, they can check if there is any duplicate key within a property file or outside of property file.

### Checking duplicate key within a property file:

This is comparatively simple by adding external plug ins to your IDE. For example,

VS Code has the plugin which can detect if there is a duplicate key within a property file.

Install the plugin from [here](https://marketplace.visualstudio.com/items?itemName=TirthPatel.identify-duplicate-properties#:~:text=%F0%9F%9A%80%20Identify%20Duplicate%20Properties&text=%F0%9F%91%89%20Open%20any%20.,highlighted%20with%20red%20wavy%20underlines.).

A screenshot of a computer error

Description automatically generated

### Checking duplicate keys within multiple properties files

The elements folder can hold multiple properties file in terms of multiple page objects. Automation to work properly, the keys within these page objects should be unique. To test these developers can run a simple command while passing the arguments like:

Java -jar <SGAutoPro.jar> --CheckDuplicateKey true

This will check the uniqueness of the keys within the elements folder as set by your project.properties and print the output to console. Below is an example:

2024-Nov-14 17:05:42.364 \_[32mINFO \_[m - Duplicate key: "reg\_state\_confirm" found in files: [SC\_Android\_Reg.properties, DE\_Android\_Reg.properties]

2024-Nov-14 17:05:42.365 \_[32mINFO \_[m - Duplicate key: "login\_windowHeader" found in files: [DE\_Android\_Login.properties, SC\_Android\_Login.properties]

2024-Nov-14 17:05:42.365 \_[32mINFO \_[m - Duplicate key: "reg\_last\_name" found in files: [SC\_Android\_Reg.properties, DE\_Android\_Reg.properties]

# APPENDIX 3

## Setting browser capabilities from external file

Currently for Chrome, Firefox and Edge Browser capabilities can be set from the external json file available under the folder **config\capabilities**.

Files available are the following:

ChromeDriverManager.json

EdgeDriverManager.json

FireFoxDriverManager.json

Do not change the file names to any other name. Each file will have 2 sections. Arguments and Capabilities. Here is an example of the format.

A black rectangle with white text

AI-generated content may be incorrect.

**addArguments**:

Arguments can be added like

"start-maximized",

"headless",

A black rectangle with white text

AI-generated content may be incorrect.

**setCapability**

Capabilities can be added like

"browserName": "chrome",

"platformName": "WINDOWS"

A black screen with white text

AI-generated content may be incorrect.

For more information see the article:

<https://developer.chrome.com/docs/chromedriver/capabilities#recognized_capabilities>

<https://peter.sh/experiments/chromium-command-line-switches/>

## Running Appium with image plugin

Appium can recognize images and can click, or type based at the center of the image. SGAutoPro has been developed based on this capability of Appium. To use these functionalities, appium server needs to be run with image plugin.

Here is the command for installing image plugins to Appium server. It should be performed one time.

appium plugin install images

Once plugin is installed, Appium server can be run with image plugin using below command.

appium --use-plugins=images

### KEYS USED IN MOBILE.PROPERTIES

Below four the keys can be used in mobile.properties file to control the image recognition process by Appium. When not supplied, default value will be considered.

* IMAGE\_MATCH\_THRESHOLD
* FIX\_IMAGE\_TEMPLATE\_SCALE
* FIX\_IMAGE\_TEMPLATE\_SIZE
* DEFAULT\_IMAGE\_TEMPLATE\_SCALE

More details about the keys can be found at <https://github.com/appium/appium/blob/master/packages/images-plugin/docs/find-by-image.md>

**Baseline images folder:**

This key is used in the mobile.properties folder to notify automation framework about the folder or directory where baseline images are stored. For example, if you are running automation for a Samsung device and baseline images are stored in the folder Samsung\_images, respective folders can be provided in the key.

baseline\_images=samsung\_device

Post that in the automation script, steps can be created without mentioning the folder name. This will be helpful to seamlessly change the device during automation run.

### METHODS SUPPORTED

Two methods are supported for use of image plugin in appium.

* Click
* Type

Along with regular operation parameters, additional *operation\_type* to be included to access this feature along with the location to the image to which comparison is needed. Methods are updated in respective sections.

"operation\_type": "ON\_MOBILE\_SCREEN"

For best result, images should be snipped on the same screen where automation is going to run and crops to get the desired base line image.

## Parameterizing Testcases Based on external Inputs

### PREPARING THE DATA

Parameterization of testcase can happen based on the inputs from an excel spreadsheet. A header row is mandatory. Excel spreadsheet may have a column called “skip”. If available, and value under the column is true, that row won’t be considered for execution. If “skip” column is not available, all the rows will be considered for execution.

For example,

A screenshot of a computer

Description automatically generated

### EXECUTION STEPS

#### Iterating the testcase multiple times

Execution can happen in two ways. At testcase level or internally inside a loop. When a dataset tag is provided inside a testcase body, SGAutoPro will consider the eligible rows available for execution and trigger the same testcase for multiple rounds.

Dataset will have two parameters. First parameter denotes the location of the test data spreadsheet. Second parameter denotes the sheet name from which data is extracted. A dataset tag has the format:

"dataset":["location of the excel spreadsheet", "Sheet name"],

To use the data inside the step, simple use with the format: "${dataset.headername}”

For example:

A screenshot of a computer program

AI-generated content may be incorrect.

Report generated will have testcase name with iteration count.

A screenshot of a computer

AI-generated content may be incorrect.

#### Iterating inside a loop

While iterating inside a loop, dataset can be provided inside the loop operation. And value can be passed as

"${dataset.headername#${loopIndex}}

**Please note that operand\_lhs should be rowCount and operand\_rhs should be loopIndex.**

**A computer screen shot of a program

AI-generated content may be incorrect.**

Report generated in this case will have only one testcase with steps repeated multiple times.

Sample Testcase:

