

**Framework User Guide**

Version 1.0

Revision History

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version Number | Release  Date | Prepared By | Reviewed By | Approved By | Summary of Changes |
| 1.0 | 14-Nov-2016 | Prasanna K |  |  | Initial |
|  |  |  |  |  |  |

Contents

[1. Purpose 2](#_Toc466398884)

[2. Automation Framework 2](#_Toc466398885)

[2.1. Framework Architecture 3](#_Toc466398888)

[2.1.1. Continuous Integration 4](#_Toc466398889)

[2.1.2. Test Management 4](#_Toc466398890)

[2.1.3. Generic Utilities 4](#_Toc466398891)

[2.1.4. Test Input Data 4](#_Toc466398892)

[3. Framework Folder Structure 6](#_Toc466398893)

[3.1. Driver 7](#_Toc466398897)

[3.2. HTMLLogs 7](#_Toc466398898)

[3.3. Results 8](#_Toc466398899)

[4. Automation Maintenance 2](#_Toc466398901)

[5. Execution 3](#_Toc466398902)

# Purpose

The purpose of this document is to provide an overview of the Gallop Automation Framework.

# Automation Framework

GALLOP Framework is an In-house automation framework created with the vision to support application across various platforms like Web, Mobile and Sauce labs with ease of implementation and limited maintenance effort on the framework. The framework is implemented using TestNG framework. The framework has a feature/capability to support Non UI based automation applications, Parallel execution with a well-structured reporting.

Gallop Framework is a Hybrid framework with multi-tier script architecture. The framework is designed with an aim make it highly robust and scalable. The application business functionality is highly modularized into reusable libraries. The aim is to minimize redundancies in script development. Typically, a single script would handle all the test cases pertaining to the Scenario/Flow under consideration. The TestNG XML (driver) takes care of binding various scripts based on the scenario flow configured. The configuration are centralized in the framework for the scripts to run across the User Roles, Browsers, Environments, scenario, Test Data etc.

The User Roles, Browsers, Test data required for the test scripts are externalized from the scripts and are maintained in workbook. This means that even a non-technical user can drive the test execution, by specifying the corresponding parameters/keywords.

Gallop Framework has a Data Layer, which ensures that appropriate test data is provided to individual test flows. This also helps to parameterize the test scripts easily.

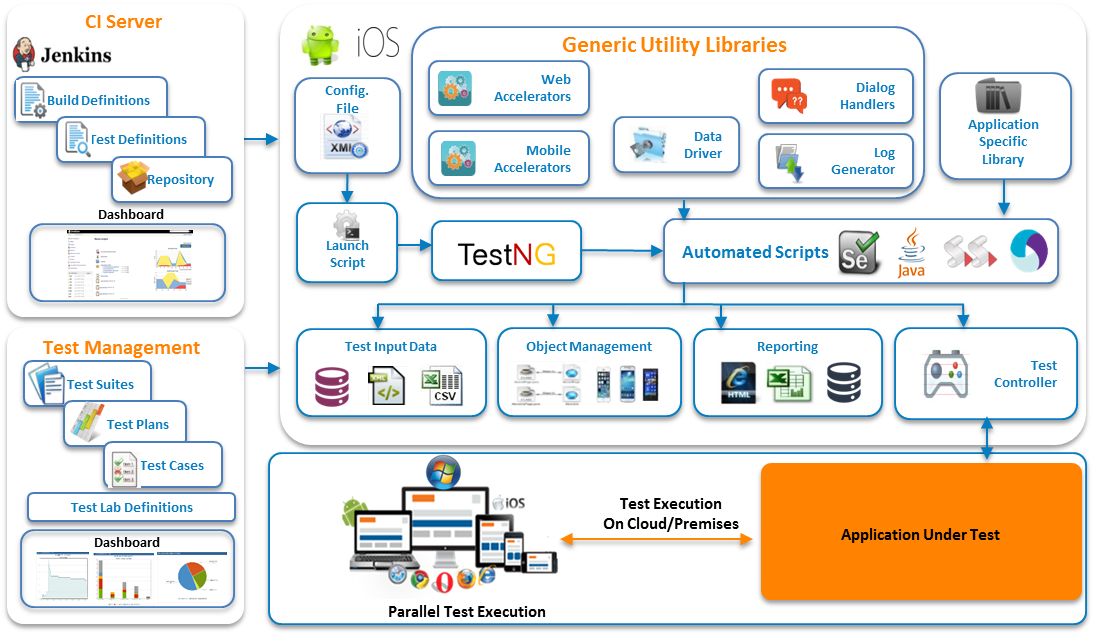
The Gallop Framework support seamless integration with Continuous Integration tool providing ease of CI implementation

Thus, Gallop Framework follows a hybrid-driven approach, combining the best practices of various frameworks.



## Framework Architecture

The following diagram presents a detailed graphical representation of the framework



**Features**:

* + Unified Framework for Web application testing & Mobile
  + Supports UI and Non UI based test automation
  + Centralized, easy to configure settings for the scripts to run across the environments
  + Supports Selenium and Appium Test Automation tool
  + TestNG is used for better organization and controlling of test automation suite
  + Execution enabled using Jenkins to support CI implementation
  + Test Execution on multiple devices, browsers and OS using Spacelabs
  + Customized test execution reports generated in CSS/HTML formats

## Continuous Integration

The CI Layer provides a mechanism to run automated scripts using CI servers like Jenkins and extends support to CI implementation

## Test Management

The Test Management Layer provides seamless integrations of test execution results with industry leading test management tools MS TFS and Jira+Zephyr

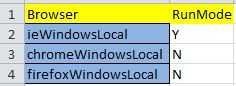
## Generic Utilities

The Generic utilities layer provides access to pre build/defined functionality which manages Dialog Handling, Reporting, Test Data, Test Flow, Web and Mobile accelerators and Execution on Sauce Labs

## Test Input Data

The Input Data for the framework is categorized into two types considering their behavior. The “Test Data” workbook is located in “”Test Data” folder within Framework.

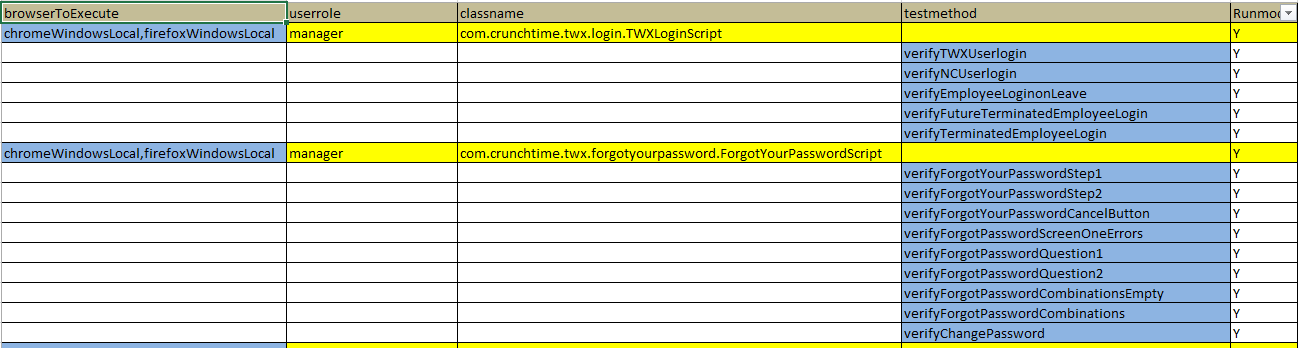
1. **Scenario Driver Data**: The configurations w.r.t scenarios can be defined in various sheet as described below
   * **Browsers** sheet: Defines the browser on which the execution should be triggered along with configurable items like version, platform and environment. The “RunMode” column in the “Browsers” sheet can be set to “Y” for executing on that browser. Set it to “N” for excluding it from execution



* + **browserConfiguration** sheet: The additional Browser configuration can be defined in the “browserConfiguration” sheet. Below is the pictorial view of the sheet.



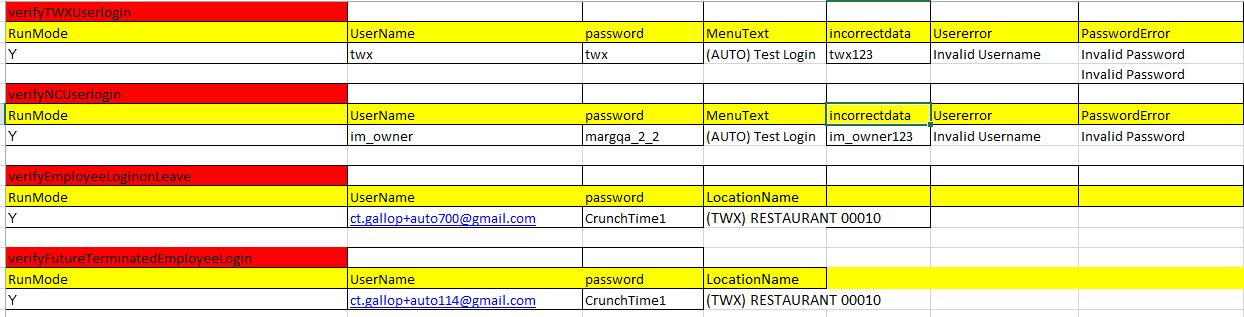
* + **allTests** sheet: For each of the selected browser combination, the test to be executed can be defined by setting “Y” in the “RunMode” column for each of the tests



1. **Test Data for test cases**: This is the placeholder for the data that is used by the test cases while performing action on the application under test. This requires frequent updates for each of the run. Below pictorial diagram highlights data required for some of the scenario of Accounts module. The test data in the cell that should be unique for each run/execution is highlighted in “Light red”

The test data in the cell that is Pre-requisite data for each run/execution is highlighted in “Yellow”.

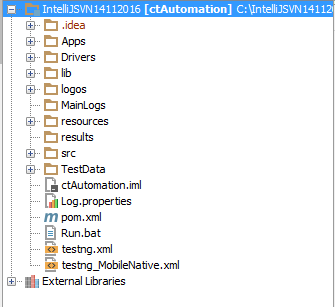
For each of the module a separate sheet is created in the “Test Data” workbook and the same configuration is implement. The various sheet are “CreateAccount”, “LaborSchedule”, and “StaffRequests”.



1. **Configuration File**: The Base URL and Saucelabs credentials are stored in the gallopReporter.properties file in the “resources” folder. The variables for base url is “baseUrl” and the variable for Saucelabs credentials are “userName” and “accessKey”

# Framework Folder Structure

The framework folder structure is depicted below



Below highlighted are the key placeholders/folders within the framework

.



## Driver

The folder contains all the drivers that are used in the project like Chrome, IE and Java

## HTMLLogs

## 

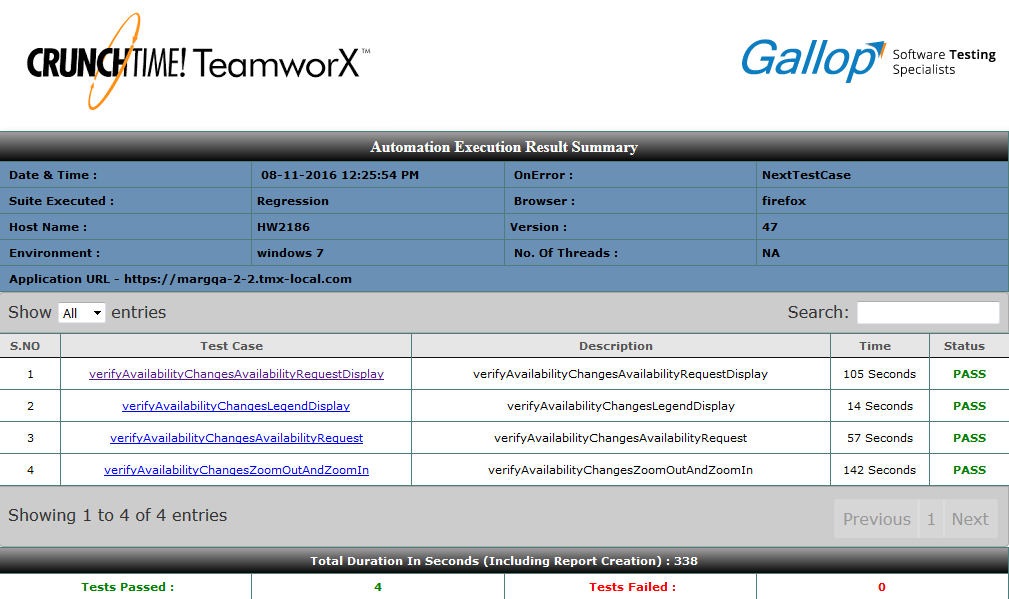
The folder contains the logs captured during each round of execution

## Results

## 

For each round of execution, based on the browser & platform selection, a sub folder is created and the results are stored. For each execution a summary and detailed reports (with test script names) are created

1. Summary Report: A summary report provides a detailed summary of configurable items along with navigation facility to individual test scripts. Below is a graphical representation of summary report.



1. Detailed Report: A detailed report provides step by step actions performed on the AUT along with results and execution time. Below is a graphical representation of detailed report.

# 

# Automation Maintenance

Once the automated scripts are developed, maintenance is needed to keep the scripts updated and running against the changes in the AUT. Defining a process to maintain the suite is imperative for effective script execution. As a part of the Maintenance approach, the following approach can be followed.

**Assess**: This stage will comprise of assessing the changes. The changes can be caused by one or more attributes like

* Change in business process and requirements or
* GUI property changes, etc.,

This will be directed towards understanding the new/modification of enhancements/changes in the existing requirements. Automation engineer will understand the GUI change (Addition of a button/link or modification of text box to drop down), change in the object properties (change in ID, addition of frames, tables etc.,), functional changes. The object map, reusable functions/methods, new functions/methods, test data and change in the environment will be analyzed and modified/created.

**Create/Modify**: All inputs for modification of the scripts are reviewed and documented. It is important to locate exact lines in scripts without affecting others.

Based on the input from the previous stage, either new scripts needs to be created for new added functionalities to the application or existing scripts needs to be updated for the enhancements.

* Create new scripts: This is carried out when new functionalities are added to application.
* Update the existing scripts: Modifications made to existing scripts based on the change requests and impacted test scripts. The base lined test scripts are modified and appropriate versioning of the scripts is performed.

**Execution**: Dry run is carried out to validate the newly created and updated automation scripts. Execution summary report will be generated at the end of every run.

# Execution

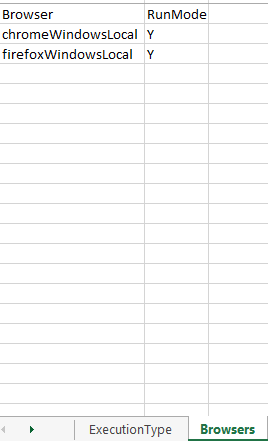
The automation scripts can be executed in the following ways

## Execution from Local:

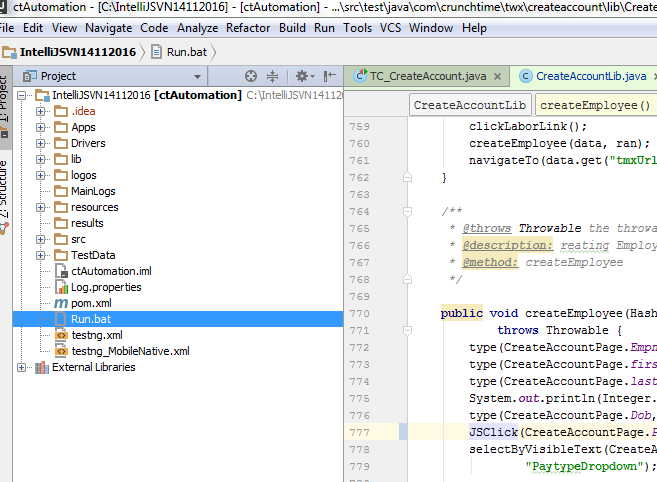
Update the following data in ExecutionType sheet.



Update the following data in Browsers sheet.



After entering the required details in ExecutionType and Browsers sheets, please run Run.bat file from command prompt (cmd)

****

**Note:** By Default, Scripts available in “allTests” sheet will be considered while generating the Dynamic TestNG.xml file.

## Execution from Jenkins:

Select the fields in build parameters section and click on build button.

