

***SPRINTEST***TM CBF-Java Technical Manual



Table of Contents

Table of Contents

[1. General Information 3](#_Toc421632163)

[1.1 Purpose of the document 3](#_Toc421632164)

[1.2 Intended Audience 3](#_Toc421632165)

[1.3 Points of Contact 3](#_Toc421632166)

[2. Framework documentation 4](#_Toc421632167)

[2.1 Class Diagrams 4](#_Toc421632168)

[2.2 Sequence Diagrams 4](#_Toc421632169)

[3. Utility library reference 6](#_Toc421632170)

[3.1 API Style 7](#_Toc421632171)

[4. Script Management 8](#_Toc421632172)

[Pre Scripting Activity 8](#_Toc421632173)

[4.1 Scripting activities 8](#_Toc421632174)

[4.1.1 Javadoc Style 8](#_Toc421632175)

[4.1.2 Creation of module driver file 8](#_Toc421632176)

[4.1.3 Creation of Object Repository 9](#_Toc421632177)

[4.1.4 Creation of reusable components in module driver file 10](#_Toc421632178)

[4.1.5 Creation of testcase in excel file 10](#_Toc421632179)

[5. Standards and Guidelines 16](#_Toc421632180)

[5.1 Coding Standards 16](#_Toc421632181)

[5.1.1 Naming convention for Objects 16](#_Toc421632182)

[5.1.2 Naming convention for Variables 16](#_Toc421632183)

[5.1.3 Guideline for Data Table 16](#_Toc421632184)

[5.1.4 Naming convention of Test Cases 16](#_Toc421632185)

[5.1.5 Naming convention of Reusable components 16](#_Toc421632186)

# General Information

## Purpose of the document

The Test Automation Framework Technical User Manual is designed with a purview of answering the most popular question – *“How does this frame - work?”*

The document highlights on the script management. This document also speaks about the generic utility functions which are defined in utility library reference.

## Intended Audience

Intended audiences for this document are Test engineers, Senior Test Engineers and Automation Experts who will be using the framework for script creation.

## Points of Contact

Please contact following POCs for more information and troubleshooting purpose on the framework

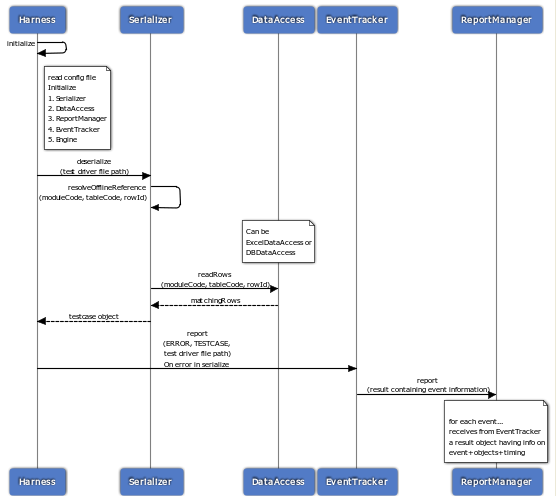
1. Contact Name :
2. Contact Name :

# Framework documentation

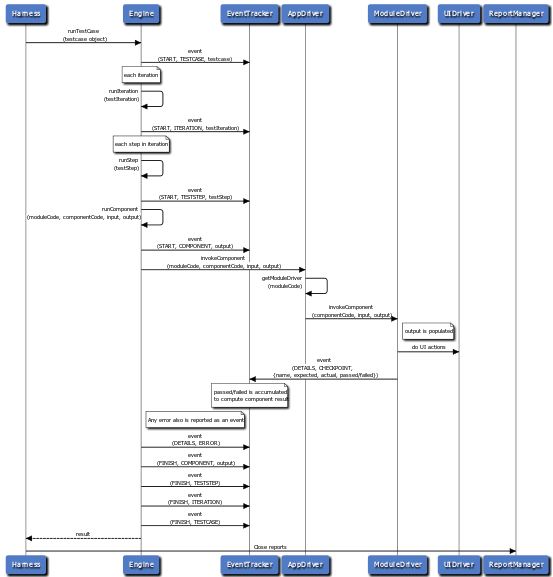
## Class Diagrams

## Sequence Diagrams

Sequence diagrams for framework functional flow for Test case deserialization:



Sequence diagrams for framework functional flow for Test case execution and reporting:



# Utility library reference

This contains the generic utility functions under the cbf.utils. It is a collection of class files to perform the generic functions.

| **Class Name** | **Description** |
| --- | --- |
| **Configuration** | Represents a common XML-based configuration. Contains methods to load from files and access configuration items |
| **DataRow** | Wrapper class on DataRow |
| **DBUtils** | Utility to handle connection to Microsoft Access Database |
| **DTAccess** | Implements functionalities like readrows to read all rows from component sheet, read selected rows etc. |
| **ExcelAccess** | Handles all the function related to Excel Data sheet like reading sheet,reading row,etc.. |
| **FileUtils** | Utility for performing various file activities. E.g. CreateFile, CreateFolder, etc |
| **FrameworkExceptions** | Utility for dealing with framework exceptions. |
| **HtmlUtils** | Handles all the HTML utilities E.g. Converting object to HTML, Map to table,etc |
| **JsonUtils** | Utility to handle all the JSON related things |
| **LogUtils** | Utility provides functionalities related to logging and error handling Levels of logging: debug|trace|details|warning|error |
| **MongoDBUtils** | Utility to handle MongoDB connection |
| **PersistentMap** | Extends HashMap and maintains a global map |
| **RegExUtils** | Utility handles string matching and replacement functionalities. |
| **SleepUtils** | Utility to handle sleep with different time slabs |
| **StringUtils** | Utility to convert various data types to string |
| **SubstitutorUtils** | Evaluates the expression which provided as parameters for test step. |
| **UniqueUtils** | Utility to retrieve unique string |
| **Utils** | Generic utility functions. |

## API Style

* API refers to Application Programming Interface. There are 3 types of [Java Programming Language](http://en.wikipedia.org/wiki/Java_(programming_language)) [Application Programming Interfaces (APIs)](http://en.wikipedia.org/wiki/Application_programming_interface).
* The official core Java API, contained in the [JDK](http://en.wikipedia.org/wiki/Java_Development_Kit) or [JRE](http://en.wikipedia.org/wiki/Java_Runtime_Environment), of one of the editions of the [Java Platform](http://en.wikipedia.org/wiki/Java_Platform).
* The three editions of the Java Platform are [Java ME (Micro edition)](http://en.wikipedia.org/wiki/Java_Platform,_Micro_Edition), [Java SE (Standard edition)](http://en.wikipedia.org/wiki/Java_Platform,_Standard_Edition), and [Java EE (Enterprise edition)](http://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition).

# Script Management

This section provides details on how to create module drivers, module-wise data files and test case excel files.

## Pre Scripting Activity

* Understand the test cases and identify reusable components.
* Application functionalities are broken down to components.
* The granularity of the components are adjusted in such a way that tests can be built as an assembled sequence of such components.
* Creation of Test inputs:
* Test case file (Excel): Create the test case file for each test case separately
* Test Data (Excel) : Gather the test data required for Test execution
* Object repository (Excel) : Store the HTML property of web-elements

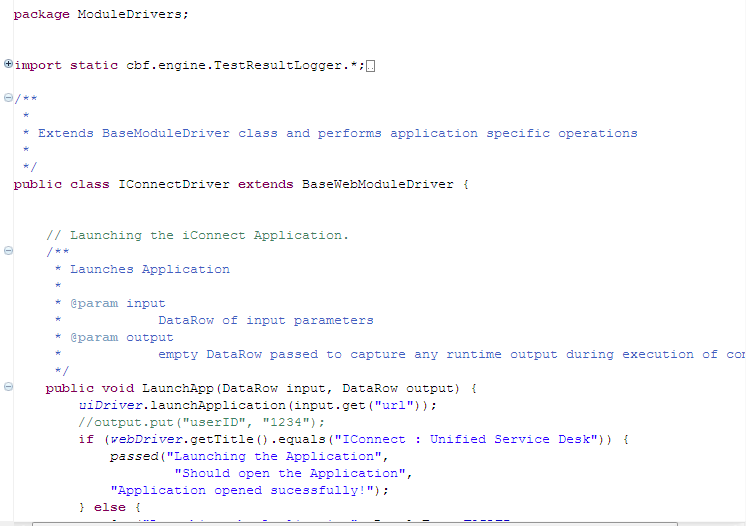
## Scripting activities

Scripting activities are mainly involved in creation of module driver and creation of test cases in excel file.

## Javadoc Style

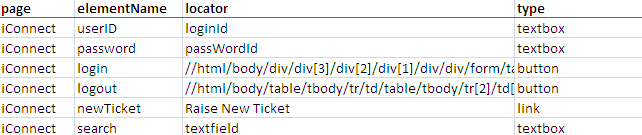
## Creation of module driver file

* Business processes in an application are broken down into components.
* The **Module Driver** is a java file which holds together a comprehensive collection of such components.
* The components designed in the module driver are used to design the flow of the design script (or Test Case Data sheet).
* It is called as Module driver because a single application can be broken down into multiple modules based on the size of the application.
* For a simple application there might be only one module, but for complex applications there might be multiple.
* Number of modules for an application is purely based on the designer’s perspective of manageability of the application.
* Every module driver has minimum following entities.
* Class name is “<java file name>Driver”
* Methods for implementing components.
* **Example:**
* To Create ManagerUser module driver file.
* ManageUser.java file has following minimum entities.
* Refer screenshot below.
* ManageUserDriver Class
* Methods for components with signature(DataRow input, DataRow output)



## Creation of Object Repository

* + **Create Shared Object Repository**
* Maintain shared object repository with object identification details in uiMap.xls.
* This is shared across all test cases for a given application.
* Following is a sample uiMap file:



## Creation of reusable components in module driver file

Every action identified for the application is defined in the Module Driver as a function. These components contain scripts to invoke actions on controls and also track the validation in the application.

**Example:**

Following is an action for Login to application. Login component requires user name and password values. Every reusable component accepts two parameters. Parameters are “input” and “output”. Input parameter is dictionary object which contains testdata of the component.



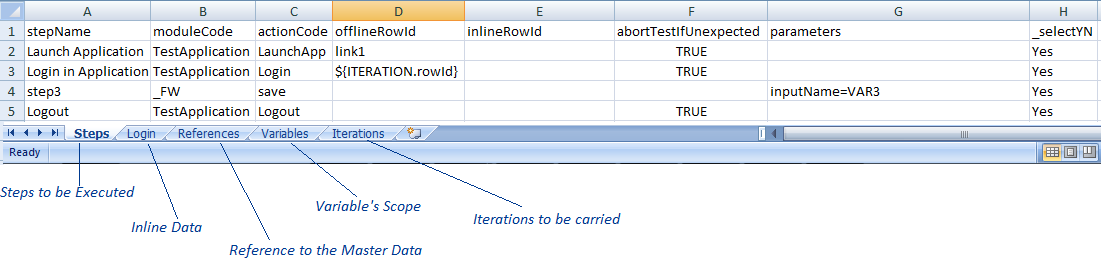
**webActions = Object of the Web Application Driver Class**

* **Step 1:** Sets value in userID field
* **Step 2:** Sets the value in password field
* **Step 3 :** Click on “login” button
* **Step 4 :** Verification and reporting results

All such business components will be designed and added to the module driver. These modules together will comprise of a scenario (test case scenario). Hence, allowing us to perform all business processes by breaking them down into components and reusing them in different scenarios.

## Creation of testcase in excel file

* CBF is a data driven testing automation framework where test input values are read from data files.
* The different data files in CBF are Master Data and Centralized data Table(CDT) and Test Case specific data sheet.
* The data is then loaded into variables in action code in the coded scripts.
* Test Case file contains the Reference, Variables, Iterations, Steps sheets



**Conventions:**

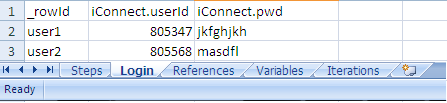
The steps containing sheet must be named as “Steps”.

1st row must be same as shown in above fig.

**‘Steps’ sheet: (TC file)**

**Column Names:**

* **stepName**: User friendly name
* **moduleCode**: module name
* **actionCode**: action name
* **offlineRowId**: rowId of a table from CDT
* **inlineRowId**: inline rowid
* **parameters**: to access or to store parameters
* \_**selectYN** : test step to be executed or not
* **‘Login’ sheet:** Inline data (This sheet name will vary as per the action code)



**Conventions:**

1st column name must be \_rowId.

If you give inline row reference then it over-rides the offline row reference.

**‘References’ Sheet: (TC file)**

Refers to the Master table

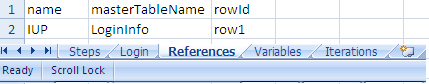
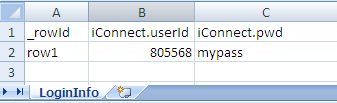


Fig 4.1.5 (c) Master Data



**Conventions:**

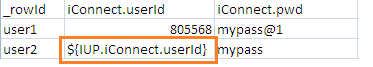
Sheet Name must be “References”.

Column-Names must be same as shown in diagram.

**Column Names:**

* **name**: reference name
* **masterTableName**: table from a master data excel
* **rowId**: rowid of a table in master data excel.

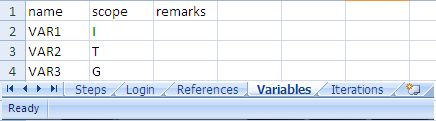
This is how you can refer:



**Syntax:** *${ReferenceName.column\_name}*

**‘Variable’ Sheet: (TC file)**

Stores values in variables



**Convention:**

Sheet Name must be “Variables”.

Column-Names must be same as shown in diagram.

**Column Names:**

* **name** : variable name
* **scope**: I-Iteration , T- Test Case , G- Global (Across Test Cases)

**How to store value in**

D:\Siddhartha\Documentations\Images for iTAF\Howtostore.png

This is how you have to write a step in your test case.

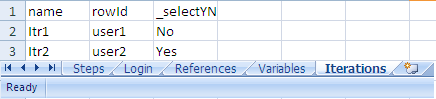
VAR3 is just a variable name declared in Variables sheet with the scope.

**How to use that variable:**D:\Siddhartha\Documentations\Images for iTAF\HowtouseVar.png

**Syntax:** *column\_name= ${VAR3.columnName}*

**‘Iterations’ sheet: (TC file)**

Iterations with multiple inputs:



**Convention:**

Sheet Name must be “Iterations”.

Column-Names must be same as shown in diagram.

**Column Names:**

* **name**: iteration variable name
* **rowId**: rowId
* \_**selectYN**: Decision whether to execute the iteration.

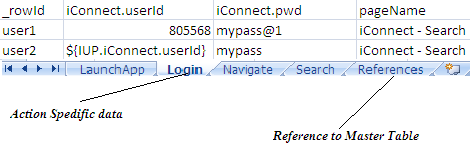
**Usage:**

D:\Siddhartha\Documentations\Images for iTAF\Howtoiter.png

**Syntax:** *${ITERATION.rowId}* (this is a FINAL string)(use as it is)

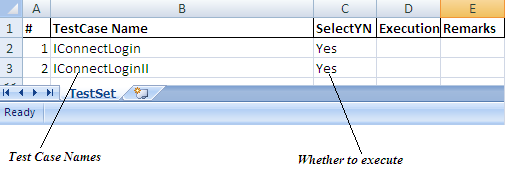
**Test Data:**

**CDT:**



**Test Set:**

List of test cases to be executed



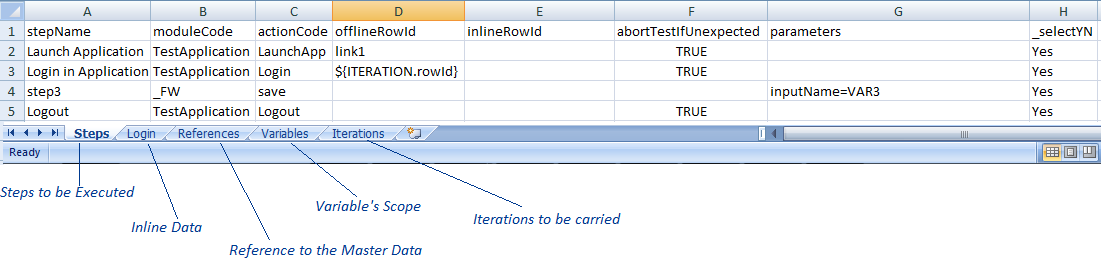
**Conventions:**

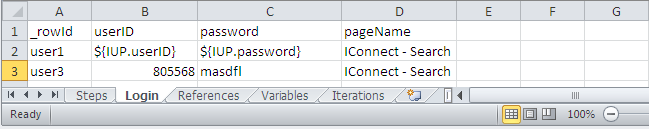
Sheet name must be “TestSet”

Column-Names must be same.

**Composing a Test Case:**

**TC:**



**CDT**



**Test Script**

* **As you can see, the action code from TC file should be same as Sheet Name from a Data File and the same string should be present as a Method name in Test Script.**
* **ModuleCode should be same as TestData excel file name but without sub-string “Data”.**

**Eg. If Excel file name is AppData then module code will be App.**

# Standards and Guidelines

## Coding Standards

## Naming convention for Objects

Specify logical names for objects

## Naming convention for Variables

## Guideline for Data Table

* CDT File (Centralized Data Table file) is named same as module driver file
  + Sheets of CDT file represents data for components of module driver file. Sheet name is same as action name and action sheet has testdata for its action.
  + First column in each sheet is “\_rowId”. It is applicable to CDT file, Master data file and Test.

## Naming convention of Test Cases

* Test Case name is starting with “Auto” e.g. Auto\_TestCaseInsurance.xls

## Naming convention of Reusable components

* Specify logical name as Action name which is related to application area.
* Specify only Alphanumeric and Underscore characters in Action name
* Space is not allowed anywhere in action name.
* Avoid underscore in beginning of the action name.
* Specify first character in Action name
* Example for Action name is : Login