**Fully Data Driven Automated Test Development Details**

**Introduction:**

The test automation framework has been designed and developed in such way that all automated tests are fully data driven. There is no need for coding test scripts for each automated test. The test engineers are only expected to prepare the test data files in Excel format for each manual test that is selected for automation based on the existing manual test case details. The manual test case details will serve as the testing and verification requirements for each automated test case.

There is only one test script coded in Java for all fully data driven automated tests that are created as Excel data files. This one and only script is basically the automated Test Engine which essentially reads the test data files, intercepts each line, interprets each row, makes decision as needed, loops through as needed, executes each command in order to perform the actual test and verification steps, produces test result pass/fail report and test detailed log files.

The test data files must be prepared by the test automation engineers according to the predefined guidelines set forth by the test automation framework design and development team. This automated test development documentation is meant for full and proper training and necessary guidance for the test engineers so that they are able to effectively and efficiently prepare the test data files for each automated test. Based on the volume and complexity of the test case, it may take on an average from 1 to 3 working days to complete each of the data driven automated tests and validate the test run.

**Test Case Details Guidelines For Test Automation:**

* The Test Cases That Are Suitable And Worthwhile For Automation Must Be Selected Wisely With Due Diligence
* Front End Application Test Automation Is Most Effective For GUI Based Regression Tests Only
* Test Automation Should Be Used For Application Features That Are Stable
* The Regression Test Cases Must Be Written In Automation Friendly Manner With Specific, Clear And Sufficient Details, No Vague Statements, No Guess Work, No Repetitious Verification
* Each Test Case For Automation Should Be Targeted For Thorough Testing Of One Specific Feature Of The Application Instead Of Cramming In Too Many Of Them In One Test Case
* There Should Be Predefined And Well Maintained Data Sets For The Test Cases For Test Automation
* Manual Review Of The Test Result Report File And The Captured Screen Shots During Runtime Is Mandatory

**The Object Repository Property Files:**

The Object Repository Property files contain the identification strings of various web elements so that the automated test can interact with the GUI objects during run time. They serve as the central repository for all AUT GUI objects that are referenced during automated test run. This approach helps keep the maintenance overhead to its minimum. This way any updating can be made in one central location instead of in multiple places whenever there is any Web element related change in the AUT. Each Application Module has its own Object Repository Property file. There is one common Object Repository Property file which contains the identification strings for the common objects that are present in multiple Application modules. The pertaining Object Repository Property file is automatically loaded into memory by the Test Engine whenever the corresponding Application module becomes active. In case of cross module testing and multiple Object Repository Property files loaded into memory, the test automation framework dynamically locates the referenced AUT GUI object from the appropriate Object Repository Property file.

The Web element labels that are displayed in the AUT are used for naming the corresponding Object Repository objects. The names of the objects in the Object Repository Property files follow a general naming convention. All spaces and non-alphabetic characters are stripped off the label text, the mandatory prefix “Gui” is added to the string and the initial character of each word segment is capitalized (Init-Cap) to form a name for the Object Repository object for each corresponding AUT object, i.e. GuiFirstName, GuiDateOfBirth, GuiMaritalStatus and so on. Sometimes meaningful suffixes are also used for better and necessary clarification, i.e. GuiAddSourceOfFund, GuiAddOccupation, GuiDocumentHeaderTab, GuiNidDocumentTab, GuiMainMenuButton and so on. The AUT GUI Object names must be supplied correctly when pertinent framework methods are called with GUI Object name being one of the method parameters.

**The Global Parameter Property File:**

The Global Parameter Property file contains the predefined global parameters that are used by the test automation framework during run time for miscellaneous purposes. The predefined values of these global parameters can be redefined to new values by the test during run time as deemed necessary. To accomplish that, the global parameters to be redefined must be put in the **“TestParams”** data sheet of the test Excel file with their corresponding new values. The complete list of all global parameters along with their predefined default values is given below.

| **No.** | **Global Parameter Names** | **Default Values** | **Description** |
| --- | --- | --- | --- |
| **1.**  **2.**  **3.**  **4.**  **5.**  **6.**  **7.**  **8.**  **9.**  **10.**  **11.**  **12.**  **13.**  **14.**  **15.**  **16.**  **17.**  **18.**  **19.**  **20.**  **21.**  **22.**  **23.**  **24.**  **25.**  **26.**  **27.**  **28.**  **29.**  **30.**  **31.**  **32.**  **33.**  **34.**  **35.**  **36.**  **37.**  **38.** | **GblActiveModule**  **GblAppLogoutClosure**  **GblAppWebUrl**  **GblBatchReportFolderPath**  **GblBatchTestRunReportFile**  **GblBrowserType**  **GblCurrentCondParameter**  **GblCurrentTestStepTag**  **GblDateFormat**  **GblDateTimeFormat**  **GblDefaultTestLogFile**  **GblDefaultTestReportFile**  **GblDocAttachmentsFolderPath**  **GblFrmErrorMessage**  **GblLastMethodReturnCode**  **GblLoggedInUser**  **GblLoginUserName**  **GblLoginPassWord**  **GblMaxWaitTime**  **GblModObjRepPrpPreLoad**  **GblOverallPassFailStatus**  **GblPrevMethodReturnCode**  **GblScreenShotCount**  **GblScreenShotFileName**  **GblScreenShotImageType**  **GblScreenShotType**  **GblScrnShotFolderPath**  **GblStepPassFailStatus**  **GblSystemSpecificError**  **GblTestLogFile**  **GblTestLogFolderPath**  **GblTestReportFile**  **GblTestReportFolderPath**  **GblTestsFolderPath**  **GblTestStopper**  **GblTestRunListFolderPath** **GblTestsToRunListFile**  **GblTimeFormat** | **CIS**  **TRUE**  **http://192.168.1.140:30080/**  **C:\AbabilNGTest\BatchRun**  **BatchRunReport.txt**  **CHROME**  **NULL**  **NULL**  **dd-MM-yyyy**  **ddMMyyyyHHmmss**  **TestReportDetails.log TestReportSummary.txt C:\AbabilNGTest\Attachments**  **NULL**  **NULL**  **NULL**  **C41911**  **S**  **30**  **CIS**  **PASS**  **NULL**  **0**  **ScreenShot**  **JPG**  **CURRWIN**  **C:\AbabilNGTest\ScrnShot**  **PASS**  **FALSE**  **TestLog.log**  **C:\AbabilNGTest\TestLog**  **TestReport.txt**  **C:\AbabilNGTest\TestReport**  **C:\AbabilNGTest\TestsToRun**  **TRUE**  **C:\AbabilNGTest\TestRunList**  **TestsToRunList**  **HH:mm:ss:SSS** | **The current module of the AUT**  **Indicates to logout and close the AUT window**  **The AUT Web URL address**  **The report folder for the batch test run**  **The batch test run report file name**  **The default browser type for the test run**  **Currently set conditional parameter name**  **Current test step indicator string for reporting**  **Default date format string**  **Default date and time format string**  **Default test log file name**  **Default test report file name**  **Folder path for document attachments**  **System generated last error message**  **Pass/Fail return code of the last method**  **Currently logged in user name**  **Default login user name**  **Default login password**  **Maximum sync. time in seconds to wait up to**  **Preload these Object Repository prop. Files**  **Overall test pass/fail status to begin with**  **Pass/Fail return code of the previous method**  **Current screen shot file count**  **Default screen shot file name**  **Default screen shot image file type**  **Default screen shot capture window type**  **Default screen shot folder path**  **Current test step pass/fail status to begin with**  **System specific error/exception indicator**  **Alternate test log file name, can be reset**  **Default test log folder path**  **Alternate test report file name, can be reset**  **Default test report folder path**  **Default folder path of the Excel test data files**  **Indicator to abort test run due to error**  **Default folder of the batch test run list file**  **Tests to run list file name**  **Default time format string** |

**Test Automation Framework Methods:**

The test automation framework contains a number of miscellaneous test methods that can be called with a set of predefined parameters. Each of these methods performs a specific task and pertinent verification. These methods are essentially called from the automated test data file for performing each test step and verification set forth in the test case chosen for automation. Each test step and corresponding verifications are mapped to one or multiple method calls in the automated test data file. Once the mapping is done, one completed Excel data file will represent one automated test for the chosen test case.

The test automation framework methods are categorized in seven different categories so far. These categories are the following:

1. Application Feature Specific Methods
2. Browser Window And Web Page Specific Methods
3. External File Input Output Specific Methods
4. Math Specific Utility Methods
5. Object Data Set, Get And Verify Specific Methods
6. Non GUI Task Specific Utility Methods
7. Web Table Interaction Specific Methods

The following table contains the category details.

| **No.** | **Category Names** | **Method Names** | **Description** |
| --- | --- | --- | --- |
| **1.** | **ApplicationFeatureSpecificMethods** | **AppDocInfoDrivingLicenseDataSet**  **AppDocInfoNidDataSet**  **AppDocInfoPassportDataSet**  **AppDocInfoSmartCardDataSet**  **AppDocInfoTinDataSet**  **AppDocInfoTradeLicenseDataSet**  **AppHomePageModuleSet**  **AppLaunch**  **AppLogin**  **AppLogout**  **AppPageMenuSelect** | **These are application feature specific methods. They perform all kinds of application feature specific tasks and corresponding verifications.** |
| **2.** | **BrowserWindowWebPageMethods** | **BrowserPageViaUrlOpen**  **BrowserTabClose**  **BrowserTabSwitch**  **BrowserWindowClose**  **BrowserWindowMaximize**  **BrowserWindowMinimize**  **BrowserWindowRestore**  **WebPageRefresh** | **These are generic methods. They perform all kinds of Browser and Web page specific tasks and corresponding verifications.** |
| **3.** | **ExternalFileInputOutputMethods** | **ExcelSheetCellValueGet**  **ExcelSheetCellValueSet**  **ExcelSheetDataFetch**  **ExcelSheetDataReload**  **ScreenShotCapture**  **TextOuputLineWrite**  **WindowsDocumentAttach** | **These are generic methods. They perform all kinds of external file input, output and attachment specific tasks and corresponding verifications.** |
| **4.** | **MathSpecificUtilityMethods** | **UtlMathNumbersAdd** | **These are generic methods. They perform all kinds of mathematical computation specific tasks and corresponding verifications.** |
| **5.** | **ObjectDataSetGetVerifyMethods** | **ObjectActionSet**  **ObjectCurrentStateAssert**  **ObjectCurrentStateCheck**  **ObjectGroupActionConditionEvaluate**  **ObjectGroupActionRepeat**  **ObjectGroupActionSet**  **ObjectGroupActionVerify**  **ObjectInvisibilityAssert**  **ObjectPropertyValueGet**  **ObjectVisibilityAssert** | **These are generic methods. They perform all kinds of user interactions with web elements, value setting and value getting specific tasks and corresponding verifications.** |
| **6.** | **NonGUITaskSpecificUtilityMethods** | **UtlConditionalParameterSet**  **UtlDataStringMerge**  **UtlDataValueCompare**  **UtlDateTimeStringFormat**  **UtlOnFailTestAbort**  **UtlOnFailTestContinue**  **UtlParameterValueSet**  **UtlSyncTimeWait** | **These are generic utility methods. They have no interactions with GUI objects of the AUT. They perform miscellaneous utility specific tasks and corresponding verifications.** |
| **7.** | **WebTableInteractionMethods** | **WebTableCellActionSet**  **WebTableCellLocationGet**  **WebTableCellValueGet**  **WebTableColSortOrderVerify**  **WebTableDataValidate**  **WebTableIdentitySet**  **WebTablePaginatorSet** | **These are generic methods. They perform all kinds of Web table specific user interactions, tasks and corresponding verifications.** |

The framework methods are named meaningfully with self-explanatory words so that the names may provide some insight into their respective actions. The initial alphabet of each of the word sections in the method names is capitalized and each name ends with an action depicting verb.

The framework methods are of Boolean type and all of them return only Boolean values to indicate **PASS (TRUE)** or **FAIL (FALSE)** status. Based on this status, the Test Engine determines the Pass/Fail status of each of the corresponding test steps and reports the result accordingly. If one single test step fails, then the overall test pass/fail status is set to **FAIL**. If all test steps pass, only then the overall test pass/fail status is set to **PASS**. The method details are included toward the end of this document in the pertinent section.

**The Excel Test Data File Details:**

As mentioned earlier that one Excel data file represents one automated test. The Excel data file representing the given automated test contains the entire automated test case in different data sheets with clear distinction between the test steps and the test data. The concept behind this is to have the actual automated test steps of the test case and all pertaining data to run the automated test bundled all together in one Excel file. Among the data sheets, the two are mandatory sheets and the rest are auxiliary sheets. The mandatory data sheets are named **“TestMain”** and **“TestParams”** and these names are fixed. The top-most row is the column header row. The **“TestMain”** data sheet contains four columns and these column headers are named **“MethodName”**, **“ParameterList”**, **“TestReportFlag”** and **“MiscComments”**. The **“TestParams”** data sheet contains three columns and these column headers are named **“ParameterName”**, **“ParameterValue”** and **“MiscComments”**. The **“TestMain”** data sheet is the main data sheet as the name suggests and it contains the framework method calls that are mapped to the test steps and verifications in the test case. The **“TestParams”** data sheet contains all the local variable names and their corresponding values used by the automated test during run time in a key-value pair format. The **“TestParams”** data sheet also contains the predefined global variable names and their corresponding new values to be reset during run time as deemed necessary for being overridden for the automated test.

There are five different types of auxiliary data sheets. Depending on the need of the test case, there may be none, one or more of each type of auxiliary data sheets per test case. These auxiliary data sheets are mainly useful for grouping similar and repetitive actions and verifications which can be shared by multiple test steps in different sections of the test case, thus minimizing the number of rows in the **“TestMain”** data sheet. The auxiliary data sheets also support iterative runs of any test step as necessary with different set of parameterized data each time. The names of the auxiliary data sheets are not fixed and should be given meaningful names, i.e. **“TestGroupActionSet”, “TestGroupActionVerify”, “TestGroupActionRepeat”, “TestGroupConditionCheck”, “TestWebTableDataValidate”**. These are the sample meaningful and self-explanatory names of the five different types of auxiliary data sheets. Each of these auxiliary data sheet types is associated with a particular group specific method and serves specific task which is different from each other. For better clarification, these sample auxiliary data sheet types can be further examined below with examples.

The **“TestGroupActionSet”** auxiliary data sheet type contains seven columns and these column headers are named **“RowMarkingFlag”, “ObjectName”, “ActionType”, “ValueToBeSet”, “ExtraActionIndicator”, “TestReportFlag”** and **“MiscComments”**. This data sheet can be used to set values to all the web fields in the current web page and then save the information. One example of this can be setting values to all the mandatory fields in the current web page. The only designated method that uses this particular type of auxiliary data sheet for performing its actions and verifications is **ObjectGroupActionSet**.

The **“TestGroupActionVerify”** auxiliary data sheet type contains eleven columns and these column headers are named **“RowMarkingFlag”, “ObjectName”, “PropertyName”, “ValueToGetParam”, “ExtraPropertyIndicator”, ”ComparisonOperator”, “ValueToCompare”, “ValueType”, “CaseSensitivity”, “TestReportFlag”** and **“MiscComments”**. This data sheet can be used to retrieve values from the database into all the web fields in the current web page and then verify the information via baseline value comparison. One example of this can be retrieving previously saved values for all the mandatory fields in the current web page for verification purpose. The only designated method that uses this particular type of auxiliary data sheet for performing its actions and verifications is **ObjectGroupActionVerify**.

The **“TestGroupActionRepeat”** auxiliary data sheet type contains five columns and these column headers are named **“RowMarkingFlag”, “MethodName”**, **“ParameterList”**, **“TestReportFlag”** and **“MiscComments”**. This data sheet can be used to perform a common set of actions and verifications on a given group of the web fields in the current web page and then report the result accordingly. One example of this can be verifying a set of criteria like red color code of the label text, warning label text, red color code of the warning label text, trailing asterisk in the label as mandatory field indicator and so on and so forth for each of the mandatory fields in the current web page. The only designated method that uses this particular type of auxiliary data sheet for performing its actions and verifications is **ObjectGroupActionRepeat**.

The **“TestGroupConditionCheck”** auxiliary data sheet type contains five columns and these column headers are named **“RowMarkingFlag”, “MethodName”**, **“ParameterList”**, **“TestReportFlag”** and **“MiscComments”**. This data sheet can be used to perform a common set of actions and conditional verifications on a single or a given group of the web fields in the current web page and then report the result accordingly. One example of this can be verifying the existence or disappearance of a dependent web field or header tab based on certain value setting in another web field in the current web page. The only designated method that uses this particular type of auxiliary data sheet for performing its actions and verifications is **ObjectGroupActionConditionEvaluate**.

The **“TestWebTableDataValidate”** auxiliary data sheet type contains three fixed columns besides the web table specific columns and these column headers are named **“RowMarkingFlag”, “[Web Table Column 1]”, “[Web Table Column 2]”, “[Web Table Column 3]”, “[Web Table Column 4]”,… … …, [Web Table Column N], “TestReportFlag”** and **“MiscComments”**. The web table specific column headers must appear enclosed in square brackets in the data sheet. This data sheet can be used to validate the given web table data for the given web table columns in the same order as the data rows appear in the web table in the current web page and then report the result accordingly. One example of this can be reading and validating the web table data against the baseline data in the data sheet for the FATCA Checklist table. The only designated method that uses this particular type of auxiliary data sheet for performing its actions and verifications is **WebTableDataValidate**. Before calling any web table specific method, it is mandatory to call the **WebTableIdentitySet** method to get the identity string in an output parameter for the target web table. This output parameter then can be used as the web table identity by any other subsequent web specific method for any interaction with the table contents. This dynamic recognition approach is necessary because there are too many Web tables preset in various pages of the AUT (Application Under Test) and keeping them all as predefined objects in the Object Repository property files would have been a significant overhead burden for maintenance. The descriptions of these two methods toward the end of this document contain more details on the call usage, dependencies, web table specific column headers and special formatting of the column headers for web cell object identification.

**The Excel Test Data Sheet Column Header Details:**

**RowMarkingFlag:** This column header is common to all five auxiliary data sheets. This column contains the indicative tags, usually in upper case key word form, so the Test Engine can determine how to treat the row for execution. The following are some examples of such tags. The default tag is **ROIFLAG** which means Row Of Interest and indicates normal execution. The **ROWSKIP** tag indicates to skip the row without execution and move on to the next row. It is treated the same as a comment line. The **SCRNSHOT** tag captures the screen shot with default settings after executing the marked row with this tag. The **WEBTABLE** tag indicates to the Test Engine that the row execution is meant for a web table and must utilize web table specific methods for performing the task. Also any meaningful custom tag like **BASICINFO1, BASICINFO2, FINANINFO, …CREDITCARDINFO** and so on can be used to indicate to the Test Engine to selectively execute the named set or sets of rows only.

The set indicating tag can be further delimited by colon “:” to indicate a subset within a set. For example, the **“ROISET1:SCRNSHOT”** tag captures the screen shot with default settings after executing the marked row with this tag. The **“WEBTABLE:SCRNSHOT”** tag captures the screen shot with default settings after executing the marked row with this **WEBTABLE** tag. The tags like **“WEBTABLE:SET1”**, **“WEBTABLE:SET2”**, **“WEBTABLE:SET3”** and so on can be used to differentiate among various sets of category **WEBTABLE**. In this case, if the calling method parameter value is set to **“WEBTABLE:SET1”**|**“WEBTABLE:SET3”**, it indicates to execute the **SET1** and **SET3** of the **WEBTABLE** tag and skip the **SET2** of the same. The **“WEBTABLE:SET2:SCRNSHOT”** tag captures the screen shot with default settings after executing the marked row with this “**WEBTABLE:SET2”** tag only. The screen shot file name suffix can also be supplied using the colon (“:”) delimiter, i.e. **“WEBTABLE:SET2:SCRNSHOT:FATCAChecklist”, “BASICINFO2:SCRNSHOT:RedColorCodedLabel”** and so on. Among many screen shot files, the meaningful file name suffix can be very useful to know what screen shot the file contains without having to open each file.

This column value is used as one of the parameters (mapped to **RowMarkingTags**) while calling the **ObjectGroupActionSet, ObjectGroupActionVerify, ObjectGroupActionRepeat** and **ObjectGroupActionConditionEvaluate** methods for test step specific action and verification purposes. The descriptions of these methods toward the end of this document contain more details on the usage of this column and the supported key words for the same purpose.

**MethodName:** This column header is common to only **“TestMain”** mandatory data sheet, **“TestGroupActionRepeat”** and **“TestGroupConditionCheck”** auxiliarydata sheets. This column contains the name of the mapped framework methods that are executed in sequence to perform the corresponding test step actions and verifications.

**ParameterList:** This column header is common to only **“TestMain”** mandatory data sheet, **“TestGroupActionRepeat”** and **“TestGroupConditionCheck”** auxiliarydata sheets. This column contains the comma delimited list of the parameters of the corresponding method referred in the **“MethodName”** column of the same row in the data sheet. The literal string value of the parameters must be supplied enclosed in double quotes and these values are treated as is without parsing. The parameter values that are AUT GUI Object names, local variable names, global variable names or upper case key words must not be enclosed in double quotes and these are parsed accordingly by the framework. The output parameters must be named with the “Out” prefix and must be supplied enclosed in square brackets, i.e. [OutPropertyValue]. Method name along with parameter list call samples are given below for reference.

**MethodName** **ParameterList**

**ScreenShotCapture DEFAULT, DEFAULT, "CIS001SnapShot01", NULL, GblScrnShotFolderPath, NULL**

**ObjectPropertyValueGet GuiName, OBJLABEL, [OutPropertyValue], TXTCOLOR**

**ParameterName:** This column header only appears in **“TestParams”** mandatory data sheet. This column contains the local parameter names used by the test and the global parameter names whose predefined values are needed to be redefined for the test. Internally in the framework, this is treated and accessed as the key in the key-value pair data structure format.

**ParameterValue:** This column header only appears in **“TestParams”** mandatory data sheet. This column contains the values to be set to the corresponding parameters referred in the **“ParameterName”** column of the same row in the data sheet. The literal string values must be enclosed in double quotes in this column. The values not enclosed in double quotes in this column will be treated as predefined variables and will be parsed accordingly by the framework. Internally in the framework, this is treated and accessed as the value in the key-value pair data structure format.

**ObjectName:** This column header is common to only **“TestGroupActionSet”** and **“TestGroupActionVerify”** auxiliarydata sheets. This column contains the AUT GUI Object name as defined in the corresponding Object Repository Property file for the application module. The object name is case sensitive and must start with the prefix “Gui” so that it can be differentiated from the other types of parameters. The object name must not be enclosed in double quotes. This column value is used as one of the parameters (mapped to **ObjectName**) while calling the **ObjectActionSet** and **ObjectPropertyValueGet** methods for test step specific action and verification purposes. The descriptions of these two methods toward the end of this document contain more details on the usage of this column and the supported key words for the same purpose.

**ActionType:** This column header only appears in **“TestGroupActionSet”** auxiliarydata sheet. This column contains context sensitive key word in upper case to indicate the kind of action to be performed on the given GUI object, i.e. **DEFAULT, LCLICK, TEXTINPUT, ITEMSELECT, XCLOSE** and so on. A complete descriptive list of such key words is included toward the end of this document in the pertinent section. This column value is used as one of the parameters (mapped to **ActionType and CellActionType** respectively) while calling the **ObjectActionSet** and **WebTableCellActionSet** methods for test step specific action and verification purposes. The descriptions of these two methods toward the end of this document contain more details on the usage of this column and the supported key words for the same purpose.

**PropertyName:** This column header only appears in **“TestGroupActionVerify”** auxiliarydata sheet. This column contains context sensitive key word in upper case to indicate the Web element property name whose value to be obtained for the given GUI object, i.e. **DEFAULT, CURRTEXT, CURRITEM, OBJLABEL, TXTCOLOR** and so on. A complete descriptive list of such key words is included toward the end of this document in the pertinent section. This column value is used as one of the parameters (mapped to **PropertyName**) while calling the **ObjectPropertyValueGet** and **WebTableCellValueGet** methods for test step specific action and verification purposes. The descriptions of these two methods toward the end of this document contain more details on the usage of this column and the supported key words for the same purpose.

**ValueToBeSet:** This column header only appears in **“TestGroupActionSet”** auxiliarydata sheet. This column contains the actual value to be set in the given GUI object. The value can be a literal string for input boxes and must be enclosed in double quotes. The value can also be a context sensitive key word in upper case like **NULL**, **BLANK** to indicate no value or blank string or to indicate object state toggle, i.e. **ON** or **OFF**. This column value is used as one of the parameters (mapped to **ValueToBeSet**) while calling the **ObjectActionSet** and **WebTableCellActionSet** methods for test step specific action and verification purposes. The descriptions of these two methods toward the end of this document contain more details on the usage of this column and the supported key words for the same purpose.

**ValueToGetParam:** This column header only appears in **“TestGroupActionVerify”** auxiliarydata sheet. This column contains the name of the output parameter enclosed in square brackets, i.e. [OutPropertyValue]. This output parameter is dynamically set by the framework to the current value obtained from the given AUT GUI object for the given property name. This value in turn is used in the verification via comparison with the corresponding baseline value given in the **ValueToCompare** column of the same row in the data sheet. This **ValueToGetParam** column value is used as one of the parameters (mapped to **ValueParam1**) while calling the **UtlDataValueCompare** method and treated as the primary entity in the comparison. The **UtlDataValueCompare** method description toward the end of this document contains more details on the usage of this column and the supported key words for the same purpose.

**ExtraActionIndicator:** This column header only appears in **“TestGroupActionSet”** auxiliarydata sheet. This column contains specially formatted instruction to further qualify the main action indicated in the **ActionType** column of the same row in the data sheet. For example, in order to get the tooltip text, the mouse pointer may need to be moved over to a specific “x,y” point on the given object using the key word **MOUSEOVER** in the **ActionType** column of the same row. In this case, this column can be used to supply the Cartesian coordinates of the location for the mouse pointer to be moved in the format **“x,y”** relative to the given object. Another example would be, for an object contained within a web table cell, this column can be used to supply formatted instruction to identify the contained object and then to perform action on the contained object. The formatted instruction may look like **"RI=3|CN=No/Yes|OT=ONOFFSWITCH"** to identify the contained object within the web cell. Here the row index (**RI**) of the web cell is **3**, the column name (**CN**) is **“No/Yes”** and the contained object type (**OT**) is **“ONOFFSWITCH”**. A complete descriptive list of context sensitive key words in upper case to indicate the Web element object types is included toward the end of this document in the pertinent section. Putting **NULL** key word in this column indicates to ignore this column. This column value is interpreted and accordingly mapped and delegated as respective parameters to the **ObjectActionSet** and **WebTableCellActionSet** methods for test step specific action and verification purposes. The **ObjectGroupActionSet** method description toward the end of this document contains more details on the usage of this column and special instruction formats.

**ExtraPropertyIndicator:** This column header only appears in **“TestGroupActionVerify”** auxiliarydata sheet. This column contains context sensitive key word in upper case or specially formatted instruction to further qualify the main property indicated in the **PropertyName** column of the same row in the data sheet. For example, there may be a need to get the color code of the label text and to do that the sub property key word **TXTCOLOR** can be put in this column whereas the main property key word can be set to **OBJLABEL** in the **PropertyName** column. In the same token, the sub property key words **CUSTOMERNAME** or **CUSTOMERID** can be put in this column to get the customer name or id from the URL which can be indicated as the main property by the key word **URLSTRING** in the **PropertyName** column. Another example would be, for an object contained within a web table cell, this column can be used to supply formatted instruction to identify the contained object and then to obtain the given property value for the contained object. The formatted instruction may look like **"RI=4|CN=No/Yes|OT=ONOFFSWITCH"** to identify the contained object within the web cell. Here the row index (**RI**) of the web cell is **4**, the column name (**CN**) is **“No/Yes”** and the contained object type (**OT**) is **“ONOFFSWITCH”**. A complete descriptive list of context sensitive key words in upper case to indicate the Web element object types is included toward the end of this document in the pertinent section. Putting **NULL** key word in this column indicates to ignore this column. This column value is interpreted and accordingly mapped and delegated as respective parameters to the **ObjectPropertyValueGet** and **WebTableCellValueGet** methods for test step specific action and verification purposes. The **ObjectGroupActionVerify** method description toward the end of this document contains more details on the usage of this column and special instruction formats.

**ComparisonOperator:** This column header only appears in **“TestGroupActionVerify”** auxiliarydata sheet. This column contains the two letter key words in upper case indicating the operator types for the comparison, i.e. **EQ** (Equal), **GT** (Greater Than), **LT** (Lesser Than) and so on. This column value is used as one of the parameters (mapped to **CompOperator**) while calling the **UtlDataValueCompare** method for performing the comparison. The **UtlDataValueCompare** method description toward the end of this document contains more details on the usage of this column and the supported key words for the same purpose.

**ValueToCompare:** This column header only appears in **“TestGroupActionVerify”** auxiliarydata sheet. This column contains the baseline value that is expected. This value in turn is used in the verification via comparison with the corresponding actual value obtained from the given AUT GUI object for the given property name and stored in an output parameter during run time. This output parameter is supplied in the **ValueToGetParam** column of the same row in the data sheet. This **ValueToCompare** column value is used as one of the parameters while calling the **UtlDataValueCompare** method and treated as the secondary entity (mapped to **ValueParam2**) in the comparison. The **UtlDataValueCompare** method description toward the end of this document contains more details on the usage of this column and the supported key words for the same purpose.

**ValueType:** This column header only appears in **“TestGroupActionVerify”** auxiliarydata sheet. This column contains the three letter key words in upper case indicating the value types for the comparison, i.e. **NUM** (Numeric), **STR** (String). This column value is used as one of the parameters (mapped to **CompValueType**) while calling the **UtlDataValueCompare** method for performing the comparison. Based on the value of this column, the method determines the appropriate type of data and the type of comparison algorithm to use, either numeric or string. The **UtlDataValueCompare** method description toward the end of this document contains more details on the usage of this column and the supported key words for the same purpose.

**CaseSensitivity:** This column header only appears in **“TestGroupActionVerify”** auxiliarydata sheet. This column contains the Boolean key words **TRUE/YES** or **FALSE/NO** to indicate case sensitivity mode in string type comparison. This indicator is ignored in numeric type comparison. This column value is used as one of the parameters (mapped to **CaseSensitivity**) while calling the **UtlDataValueCompare** method for performing the comparison. Based on the value of this column, the method determines whether to consider or ignore the case sensitivity factor while making the string comparison. The **UtlDataValueCompare** method description toward the end of this document contains more details on the usage of this column and the supported key words for the same purpose.

**TestReportFlag:** This column header is common to all mandatory and auxiliary data sheets. This column contains the Boolean key words **TRUE/YES** or **FALSE/NO** to indicate whether or not to report the preformatted text string to the test report file in case of the **PASS** return code by the method. If this column is set to **TRUE** and the corresponding **MiscComments** column of the same row in the data sheet contains a custom message that starts with the tag “RMI:” (Report Message Indicator), then the tag is stripped off and the message is reported to the test report file if the method return code is **PASS**. Otherwise the default formatted message is reported by the Test Engine to the test report file if the method return code is **PASS**. Any kind of failure, system or verification, is always reported to the test report and log files regardless of the setting put in this column.

**MiscComments:** This column header is common to all mandatory and auxiliary data sheets. This column contains miscellaneous comments that are considered pertinent and useful for the particular method call row. This is optional and the column can be left blank if so desired. This column can also be used to report custom messages to the test report file after the successful (**PASS** state) execution of the method. In this case, the message must be prefixed with the “RMI:” (Report Message Indicator) tag to distinguish it from normal comments and also to indicate that this message is for reporting purpose. If any predefined parameter name is used in the text message prefixed with the “RMI:” tag, then the parameter must be enclosed in back quotes, i.e. **`**OutPropertyValue**`**, so that its value can be parsed, replaced and put in the report message accordingly. This column can also be used optionally for supplying meaningful string as the file name suffix for screen shot file names when the row is marked with the **SCRNSHOT** tag. In this case, the string must be prefixed with the “FSI:” (Filename Suffix Indicator) tag, i.e. “FSI:RedColorCodedLabels”. Among many screen shot files, the meaningful file name suffix can be very useful to know what screen shot the file contains without having to open each file. This column provides an additional option to supply the file name suffix for that purpose.

**The Batch Run Excel Data File Details:**

This Excel data file contains the list of the automated tests that are to be run in batch mode. The list may consist of one or more automated tests. This file is mandatory for running any automated test, either in single run or in batch run mode, from the command line. The name of this Excel file may be predefined in the **GblTestsToRunListFile** global parameter. The fully qualified folder path including this file name may be supplied in the command line as an argument or else the folder path and the test list file name will be taken from the predefined global parameters **GblTestRunListFolderPath** and **GblTestsToRunListFile** respectively.

The file contains two mandatory data sheets. The mandatory data sheets are named **“BatchTestRunList”** and **“BatchTestRunParams”** and these names are fixed. The **“BatchTestRunList”** data sheet contains four columns and these column headers are named **"RowMarkingFlag", "TestDataFileName", "TestDataFilePath", "TestReportFolderPath", "TestLogFolderPath", "TestScreenShotFolderPath", "TestTitleName"** and **"TesterName"**. The **“BatchTestRunParams”** data sheet contains three columns and these column headers are named **“ParameterName”**, **“ParameterValue”** and **“MiscComments”**.

The **“BatchTestRunList”** data sheet is the main data sheet and it contains the list of the automated test names to be run and other pertinent information of these tests as hinted in the corresponding column header names above. The **“BatchTestRunParams”** data sheet contains all the local variable names and their corresponding values used by the batch test during run time in a key-value pair format for generating header information in the batch run report file.

**The Batch Run Data Sheet Column Header Details:**

**RowMarkingFlag:** This column header only appears in **“BatchTestRunList”** mandatorydata sheet. The details are same as described in the above section.

**TestDataFileName:** This column header only appears in **“BatchTestRunList”** mandatorydata sheet. This column contains the individual names of the Excel test data files that are to be run as automated tests in batch run mode one after the other.

**TestDataFilePath:** This column header only appears in **“BatchTestRunList”** mandatorydata sheet. This column contains the individual folder path names for each corresponding test. The tests can reside in the same or different folder paths and this column must contain these folder paths correctly. Otherwise the Test Engine cannot locate the tests and will report error.

**TestReportFolderPath:** This column header only appears in **“BatchTestRunList”** mandatorydata sheet. This column contains the individual folder path names for the test report file for each test. This is the location where the Test Engine will create and save the test report file for the corresponding test.

**TestLogFolderPath:** This column header only appears in **“BatchTestRunList”** mandatorydata sheet. This column contains the individual folder path names for the test log details file for each test. This is the location where the Test Engine will create and save the test log details file for the corresponding test.

**TestScreenShotFolderPath:** This column header only appears in **“BatchTestRunList”** mandatorydata sheet. This column contains the individual folder path names for the screen shot files taken during run time for each test for record keeping purpose. This is the location where the Test Engine will create and save the screen shot files for the corresponding test.

**TestTitleName:** This column header only appears in **“BatchTestRunList”** mandatorydata sheet. This column contains the individual titles for each test which is in turn used in the test header part of the test report and the log files of the corresponding test.

**TesterName:** This column header only appears in **“BatchTestRunList”** mandatorydata sheet. This column contains the individual tester name for each test run which is in turn used in the test header part of the test report and the log files of the corresponding test.

**ParameterName:** This column header only appears in **“BatchTestRunParams”** mandatorydata sheet. The details are same as described in the above section.

**ParameterValue:** This column header only appears in **“BatchTestRunParams”** mandatorydata sheet. The details are same as described in the above section.

**MiscComments:** This column header only appears in **“BatchTestRunParams”** mandatorydata sheet. The details are same as described in the above section.

**The Command Line Test Run Details:**

The entire test framework has been compiled into a JAR file for making it compatible for command line execution. The compatible version of JVM (Java Virtual Machine) must be installed on each system where the framework JAR file is intended to be launched from the command line. Typing “java –version” at the command console prompt will confirm if JVM is installed or not by showing the Java installed version.

A batch file in Excel format must be created and saved in a known folder location which will contain the list of the automated tests to be run. The test automation framework JAR file can be executed with or without any command line argument. If executed without any command line, then the name and location of the batch file will be obtained from the predefined global parameters **GblTestsToRunListFile** and **GblTestRunListFolderPath** respectively by the framework.

Also the other way of executing the test automation framework JAR file is with a command line argument. In this case the fully qualified path including the name of the batch file in the path can be supplied as the command line argument to the JAR file command line execution. The following are syntax examples of running the test automation framework JAR file with or without command line argument.

**Without Command Line Argument:**

C:\TestAutomationFramework> java -jar TestAutomationFramework.jar

**With Command Line Argument:**

C:\TestAutomationFramework> java -jar TestAutomationFramework.jar C:\TestFolder\BatchRun\BatchTestList.xlsx

There are also other types of command line arguments that can be used with the test automation framework JAR file. These command line arguments serve different and useful purposes. Among these command line arguments, one can be used to print a list of the global parameters on to the console window. While the others can be used to print the list of the GUI object names and their corresponding on screen filed label names from each module specific Object Repository Property file for reference purposes.

The usage of the key word **GBLPARAMS** command line argument will dynamically print to the command console window, the formatted contents of the most up-to-date global parameter property file which is packaged within the JAR file. The output can also be redirected to an external text file for reference purposes during the development or review of the automated tests and data files in Excel format. The following are syntax examples of running the test automation framework JAR file with the command line argument.

**Printed On The Command Console Window:**

C:\TestAutomationFramework> java -jar TestAutomationFramework.jar GBLPARAMS

**Output Redirected To The Text File:**

C:\TestAutomationFramework> java -jar TestAutomationFramework.jar GBLPARAMS > GlobalParamsList.txt

The usage of the following command line arguments will dynamically print to the command console window, the formatted contents of the most up-to-date Object Repository Property files which are packaged within the JAR file. The output can also be redirected to an external text file for reference purposes during the development or review of the automated tests and data files in Excel format. The following are syntax examples of running the test automation framework JAR file with the command line arguments.

**Printed On The Command Console Window:**

C:\TestAutomationFramework> java -jar TestAutomationFramework.jar CIS

C:\TestAutomationFramework> java -jar TestAutomationFramework.jar DEPOSIT

C:\TestAutomationFramework> java -jar TestAutomationFramework.jar SANCTIONLIMIT

**Output Redirected To The Text File:**

C:\TestAutomationFramework> java -jar TestAutomationFramework.jar CIS > CisObjList.txt

C:\TestAutomationFramework> java -jar TestAutomationFramework.jar DEPOSIT > DepositObjList.txt

C:\TestAutomationFramework> java -jar TestAutomationFramework.jar SANCTIONLIMIT > SanctionLimitObjList.txt

**The Test Steps And Method Calls Mapping Sample:**

The following are some sample snippets of the actual test steps along with verifications and the corresponding method calls that are mapped to perform the same actions and verifications stated in these test steps in the same order. The comments under each method call are given to explain the purpose of the method calls in order to have a better understanding of the implementation of the sequential mapping of the test steps to the method calls.

**Test Step No. 1 And Verifications:**

S1: Launch, Logon And Navigate To Ababil-->CIS Module:

Logon to Ababil application. Once the Ababil home page appears, click on the icon of the CIS module, then dropdown the CIS main menu, click on CIS menu item and then click on the Customers menu item.

V1: Verify that the Customer page appears as per expectation. The "Search a customer" and “+ New Customer” buttons are present on the page. Take a screen shot of the page and save to the designated storage for record keeping. [Screen Shot 01]

**Corresponding Method Calls In TestMain Data Sheet:**

MethodName ParameterList

@@Test Step 1

[Comments: Test Step Identifier Tag for readability. This line is not executed by the Test Engine.]

OnFailTestAbort NULL

[Comments: Abort test when any subsequent test step fails. This is the default setting.]

AppLaunch GblBrowserType, TRUE, NULL

[Comments: Launch the application in the given Browser type.]

ObjectVisibilityAssert GuiLoginPage, GblMaxWaitTime, NULL

[Comments: Wait for the application login page to appear up to the predefined maximum wait time in seconds for synchronization and verification purposes.]

AppLogin TestUserName, TestPassWord, YES, NULL

[Comments: Login to the application with the given username and password.]

ObjectVisibilityAssert GuiAppHomePage, GblMaxWaitTime, NULL

[Comments: Wait for the application module home page to appear up to the predefined wait time in seconds for synchronization and verification purposes.]

AppHomePageModuleSet GuiCisModule, YES, NULL

[Comments: Click on the CIS module button to initiate the module in the application home page.]

ObjectVisibilityAssert GuiMainMenuButton, GblMaxWaitTime, NULL

[Comments: Wait for the application module main menu button on the top bar to appear up to the predefined wait time in seconds for synchronization and verification purposes.]

AppPageMenuSelect MAINMENU, CisMenuPath, NULL

[Comments: Select the Customers menu item from the given menu path in the main menu of the module page.

ObjectVisibilityAssert GuiActivePageTitleBar, GblMaxWaitTime, NULL

[Comments: Wait for the destination page title bar to appear up to the predefined wait time in seconds for synchronization and verification purposes.]

ObjectPropertyValueGet GuiActivePageTitleBar, TITLETEXT, [OutPropertyValue], NULL

[Comments: Get the title text of the current page and store it in the given output parameter within the square brackets.]

UtlDataValueCompare OutPropertyValue, EQ, "Customer", STR, TRUE, NULL

[Comments: Compare and verify the title text against the expected value.]

ObjectVisibilityAssert GuiSearchACustomer, NULL, NULL

[Comments: Confirm the visibility of the given push button as part of the verification.]

ObjectVisibilityAssert GuiNewCustomer, NULL, NULL

[Comments: Confirm the visibility of the given push button as part of the verification.]

ScreenShotCapture DEFAULT, DEFAULT, "CIS001SnapShot01", NULL, GblScrnShotFolderPath, NULL

[Comments: Capture and save the screen shot of the current page for record keeping purpose.]

TextOutputLineWrite "Test Step S1: Launch, Logon And Navigate To Ababil-->CIS Module -- Verification", GblStepPassFailStatus, GblTestReportFile, GblTestResultFolderPath, NULL

[Comments: Print the test step verification pass/fail result to the given test report file.]

**Test Step No. 2 And Verifications:**

S2: Open Customer Information File Dialog:

Click on the “+ New Customer“ button.

V2: Verify that the Customer Information File dialog page appears as per expectation. An asterisk mark (\*) is present beside each mandatory field. Take a screen shot of the page and save to the designated storage for record keeping. [Screen Shot 02]

**Corresponding Method Calls In TestMain Data Sheet:**

MethodName ParameterList

@@Test Step 2

[Comments: Test Step Identifier Tag for readability. This line is not executed by the Test Engine.]

ObjectActionSet GuiNewCustomer, LCLICK, NULL, NULL

[Comments: Left mouse click on the given push button labelled “New Customer”.]

ObjectVisibilityAssert GuiCustomerInformationFileDialog, NULL, NULL

[Comments: Wait for the dialog page to appear up to the predefined wait time in seconds for synchronization and verification purposes.]

OnFailTestContinue NULL

[Comments: Continue test even when any subsequent test step fails. This is optional setting.]

ObjectPropertyValueGet GuiName, OBJLABEL, [OutPropertyValue], NULL

[Comments: Get the label text of the given web element and store it in the given output parameter within the square brackets.]

UtlDataValueCompare OutPropertyValue, EQ, "Name\*", STR, TRUE, NULL

[Comments: Compare and verify the label text including the trailing asterisk against the expected value for mandatory field indicator verification.]

ObjectPropertyValueGet GuiCustomerType, OBJLABEL, [OutPropertyValue], NULL

[Comments: Get the label text of the given web element and store it in the given output parameter within the square brackets.]

UtlDataValueCompare OutPropertyValue, EQ, "Customer Type\*", STR, TRUE, NULL

[Comments: Compare and verify the label text including the trailing asterisk against the expected value for mandatory field indicator verification.]

ScreenShotCapture DEFAULT, DEFAULT, "CIS001SnapShot02", NULL, GblScrnShotFolderPath, NULL

[Comments: Capture and save the screen shot of the current page for record keeping purpose.]

TextOutputLineWrite "Test Step S2: Open Customer Information File -- Verification", GblStepPassFailStatus, GblTestReportFile, GblTestResultFolderPath, NULL

[Comments: Print the test step verification pass/fail result to the given test report file.]

**Test Step No. 3 And Verifications:**

S3: Blank Customer Information File Submit:

Click on the Submit button without inserting any data.

V3: Verify that the submission operation is unsuccessful and the mandatory fields are marked in red color indicating that these are required data fields. Also verify that the appropriate warning texts are displayed underneath each field in red color. Take a screen shot of the page and save to the designated storage for record keeping. [Screen Shot 03]

**Corresponding Method Calls In TestMain Data Sheet:**

MethodName ParameterList

@@Test Step 3

[Comments: Test Step Identifier Tag for readability. This line is not executed by the Test Engine.]

ObjectActionSet GuiSubmit, LCLICK, NULL, NULL

[Comments: Left mouse click on the given push button.]

ObjectPropertyValueGet GuiName, OBJLABEL, [OutPropertyValue], TXTCOLOR

[Comments: Get the color code of the label text of the given web element and store it in the given output parameter within the square brackets.]

UtlDataValueCompare OutPropertyValue, EQ, TextRedColorCode, STR, TRUE, NULL

[Comments: Compare and verify the label text red color code against the expected value.]

ObjectPropertyValueGet GuiName, WRNLABEL, [OutPropertyValue], TXTCOLOR

[Comments: Get the color code of the warning label text of the given web element and store it in the given output parameter within the square brackets.]

UtlDataValueCompare OutPropertyValue, EQ, TextRedColorCode, STR, TRUE, NULL

[Comments: Compare and verify the warning label text red color code against the expected value.]

ObjectPropertyValueGet GuiName, WRNLABEL, [OutPropertyValue], NULL

[Comments: Get the warning label text of the given web element and store it in the given output parameter within the square brackets.]

UtlDataValueCompare OutPropertyValue, EQ, "Customer name is required.", STR, TRUE, NULL

[Comments: Compare and verify the warning label text against the expected value.]

ObjectPropertyValueGet GuiCustomerType, WRNLABEL, [OutPropertyValue], TXTCOLOR

[Comments: Get the color code of the warning label text of the given web element and store it in the given output parameter within the square brackets.]

UtlDataValueCompare OutPropertyValue, EQ, TextRedColorCode, STR, TRUE, NULL

[Comments: Compare and verify the warning label text red color code against the expected value.]

ObjectPropertyValueGet GuiCustomerType, WRNLABEL, [OutPropertyValue], NULL

[Comments: Get the warning label text of the given web element and store it in the given output parameter within the square brackets.]

UtlDataValueCompare OutPropertyValue, EQ, "Customer type is required.", STR, TRUE, NULL

[Comments: Compare and verify the warning label text against the expected value.]

ScreenShotCapture DEFAULT, DEFAULT, "CIS001SnapShot03", NULL, GblScrnShotFolderPath, NULL

[Comments: Capture and save the screen shot of the current page for record keeping purpose.]

TextOutputLineWrite "Test Step S3: Blank Customer Information File Submit -- Verification", GblStepPassFailStatus, GblTestReportFile, GblTestResultFolderPath, NULL

[Comments: Print the test step verification pass/fail result to the given test report file.]

**Test Step No. 4 And Verifications:**

S4: Valid Name And Valid Customer Type:

Type in a name in the Name field, select a customer type, i.e. SINGLE, and then click on the submit button.

V4: Verify that the submit operation is successful and a new page is displayed as per expectation with the newly created customer name and customer ID appearing on the top of the page. Verify that the Edit Profile button exists on the page as well. Take a screen shot of the page and save to the designated storage for record keeping. [Screen Shot 03]

**Corresponding Method Calls In TestMain Data Sheet:**

MethodName ParameterList

@@Test Step 4

[Comments: Test Step Identifier Tag for readability. This line is not executed by the Test Engine.]

OnFailTestAbort NULL

[Comments: Abort test when any subsequent test step fails. This is the default setting.]

ObjectActionSet GuiName, DEFAULT, "Simon Saint", NULL

[Comments: Set the text input field labelled “Name” with the given value.]

ObjectActionSet GuiCustomerType, DEFAULT, "SINGLE", NULL

[Comments: Set the dropdown list field labelled “Customer Type” with the given value.]

ObjectActionSet GuiSubmit, DEFAULT, NULL, NULL

[Comments: Left mouse click on the given push button labelled “Submit”.]

ObjectVisibilityAssert GuiActivePageTitleBar, GblMaxWaitTime, NULL

[Comments: Wait for the destination page title bar to appear up to the predefined wait time in seconds for synchronization and verification purposes.]

ObjectPropertyValueGet GuiActivePageTitleBar, TITLETEXT, [OutPropertyValue], NULL

[Comments: Get the title text of the current page and store it in the given output parameter within the square brackets.]

ObjectPropertyValueGet ACTIVEPAGE, URLSTRING, [OutSingleCustomerId], CUSTOMERID

[Comments: Get and extract the customer id from the URL of the current page and store it in the given output parameter within the square brackets.]

UtlDataStringMerge "Single Customer: Simon Saint |Customer ID: ", AE, OutSingleCustomerId, [OutActivePageTitle], NULL

[Comments: Construct the expected title text with the customer id obtained from the URL string for comparison with the actual title text of the page.]

UtlDataValueCompare OutActivePageTitle, EQ, OutPropertyValue, STR, TRUE, NULL

[Comments: Compare and verify the actual title text against the expected value just been constructed.]

TextOutputLineWrite "Test Step S4: Single Customer Page Title Showing Customer Name And ID -- ", OutActivePageTitle, GblTestReportFile, GblTestResultFolderPath, NULL

[Comments: Print the verified title text containing the customer name and id to the given test report file.]

ObjectVisibilityAssert GuiEditProfile, NULL, NULL

[Comments: Confirm the visibility of the given push button as part of the verification.]

ScreenShotCapture DEFAULT, DEFAULT, "CIS001SnapShot04", NULL, GblScrnShotFolderPath, NULL

[Comments: Capture and save the screen shot of the current page for record keeping purpose.]

TextOutputLineWrite "Test Step S4: Valid Name And Valid Customer Type -- Verification", GblStepPassFailStatus, GblTestReportFile, GblTestResultFolderPath, NULL

[Comments: Print the test step verification pass/fail result to the given test report file.]

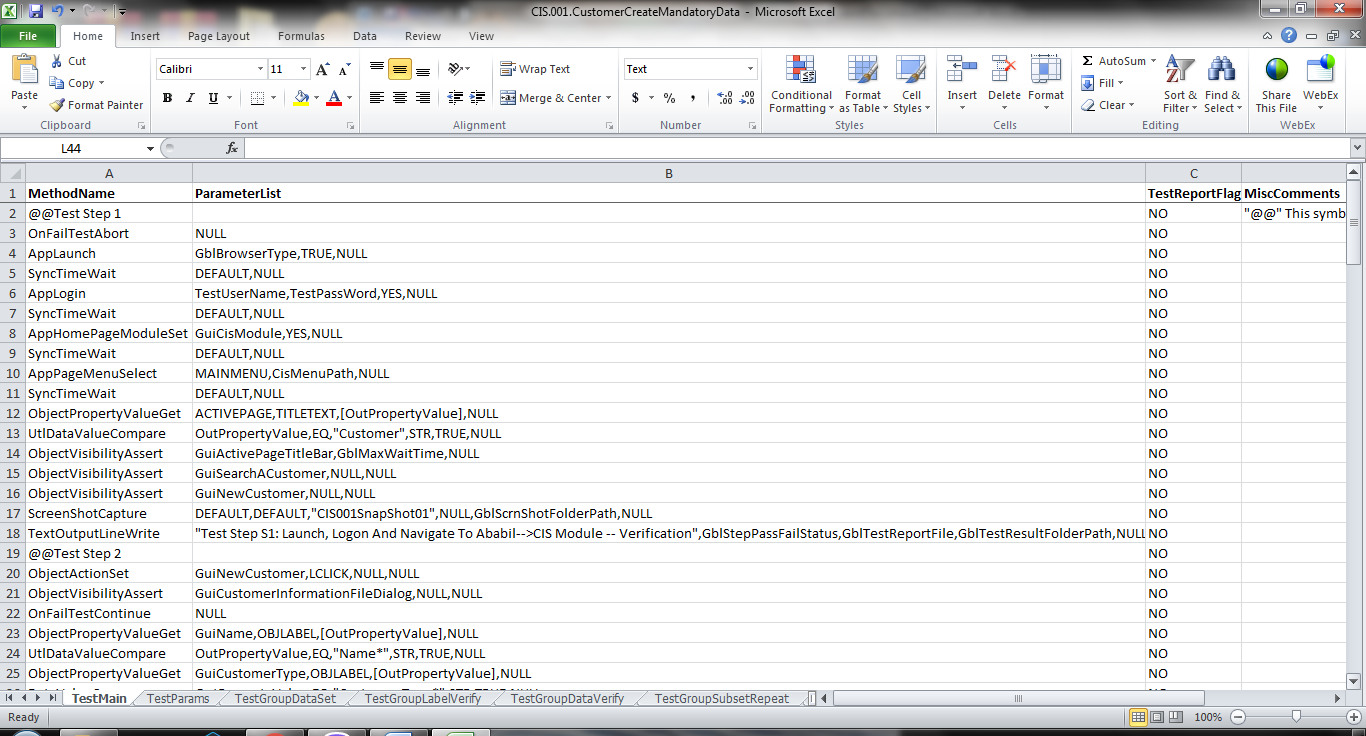
**The Test Automation Framework Methods And Parameters Details:**

The test automation framework methods, parameters, relevant key words in uppercase and their usage and all other pertinent details are included in another document called **“TestAutomationFrameworkMethodAndParameterDetails.doc”** for reference.

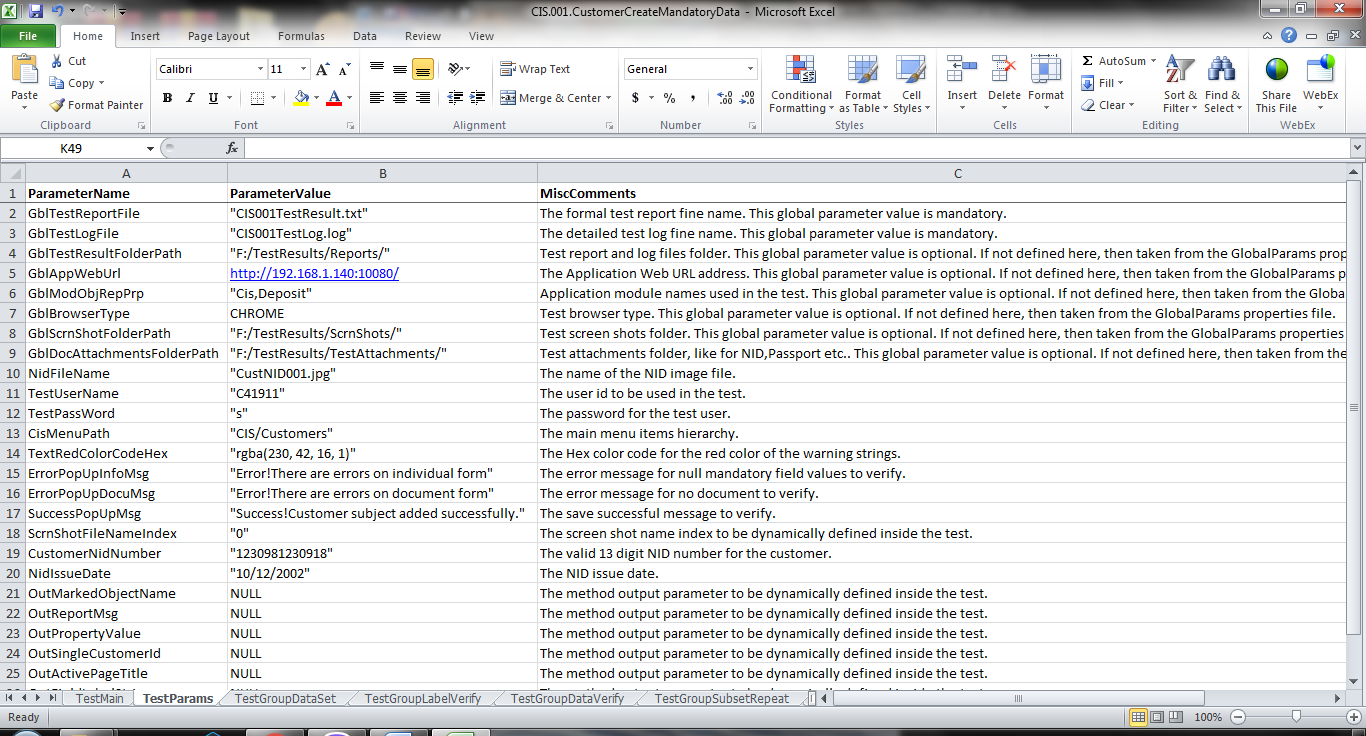
**Test Data Sheet Screen Shots For Reference:**

The following are sample screen shots representing each of the data sheets of the test data file in Excel format.

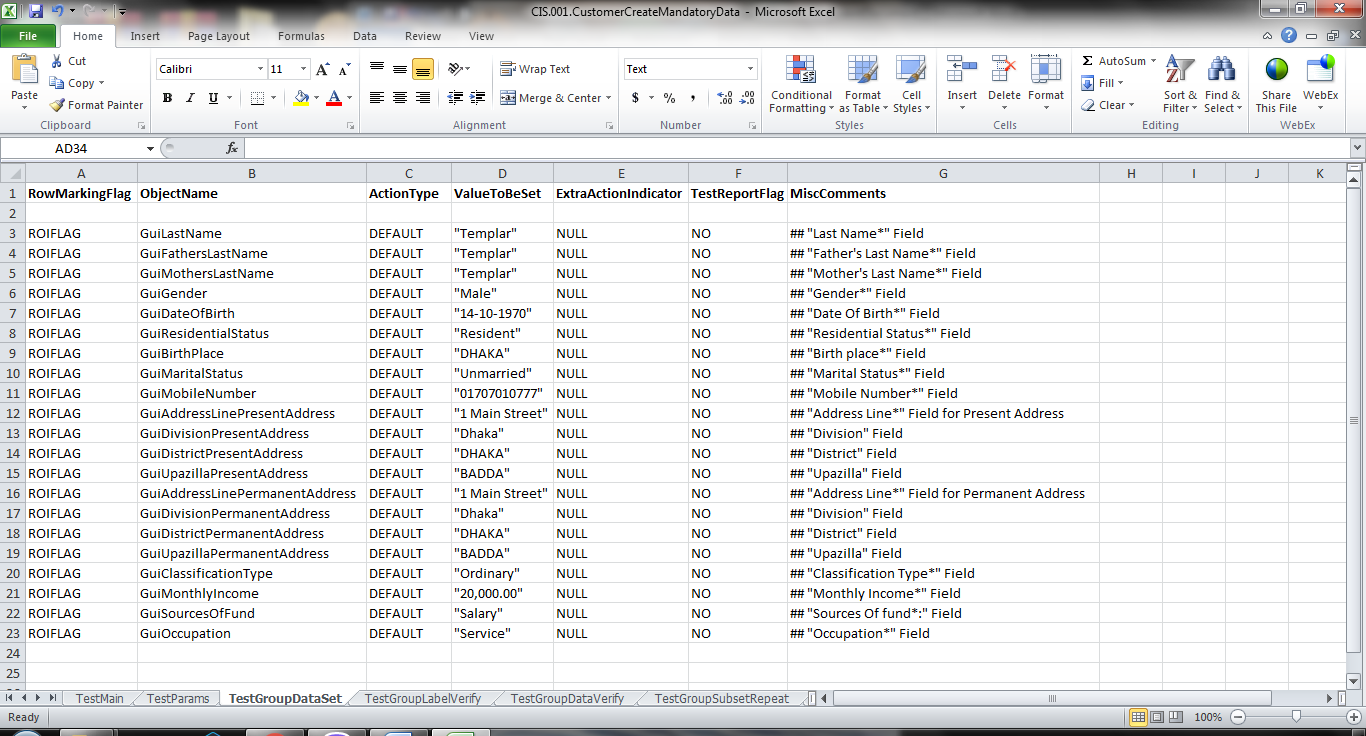
**TestMain Data Sheet:**



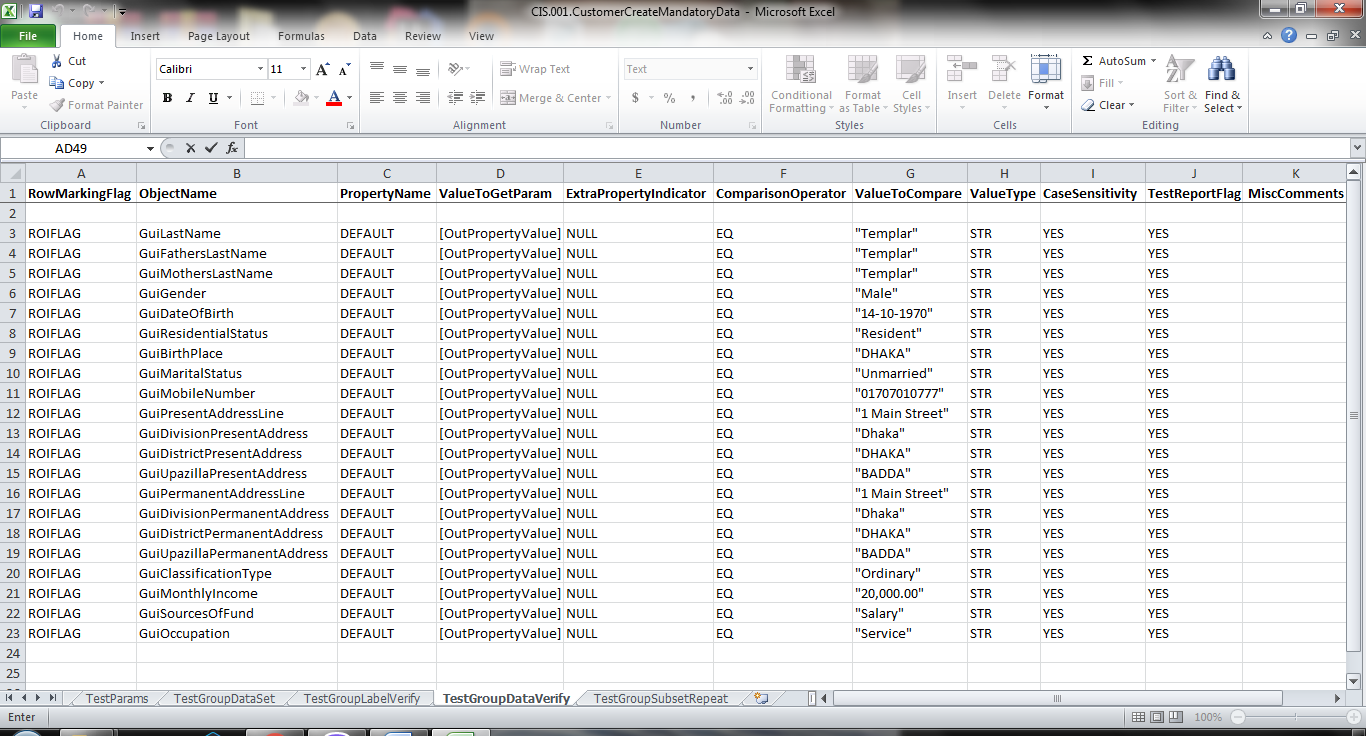
**TestParams Data Sheet:**



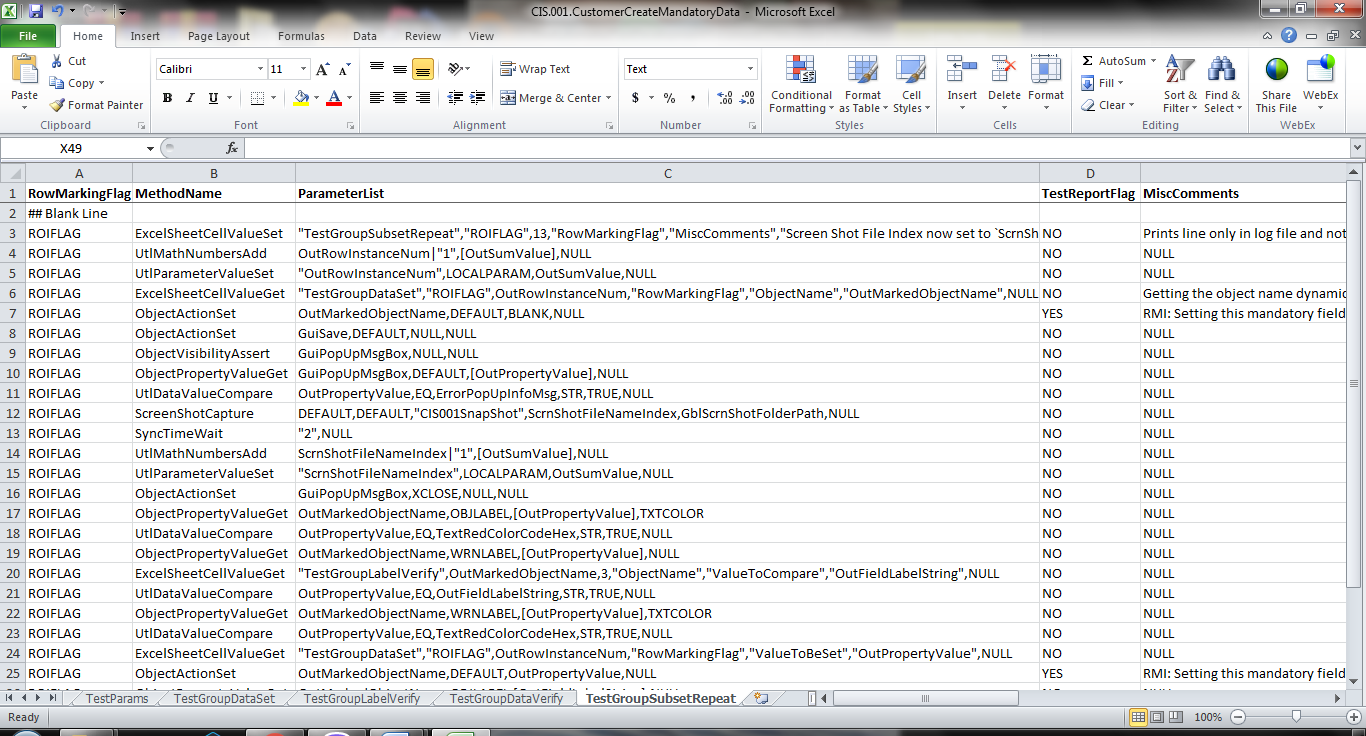
**TestGroupActionSet Data Sheet:**



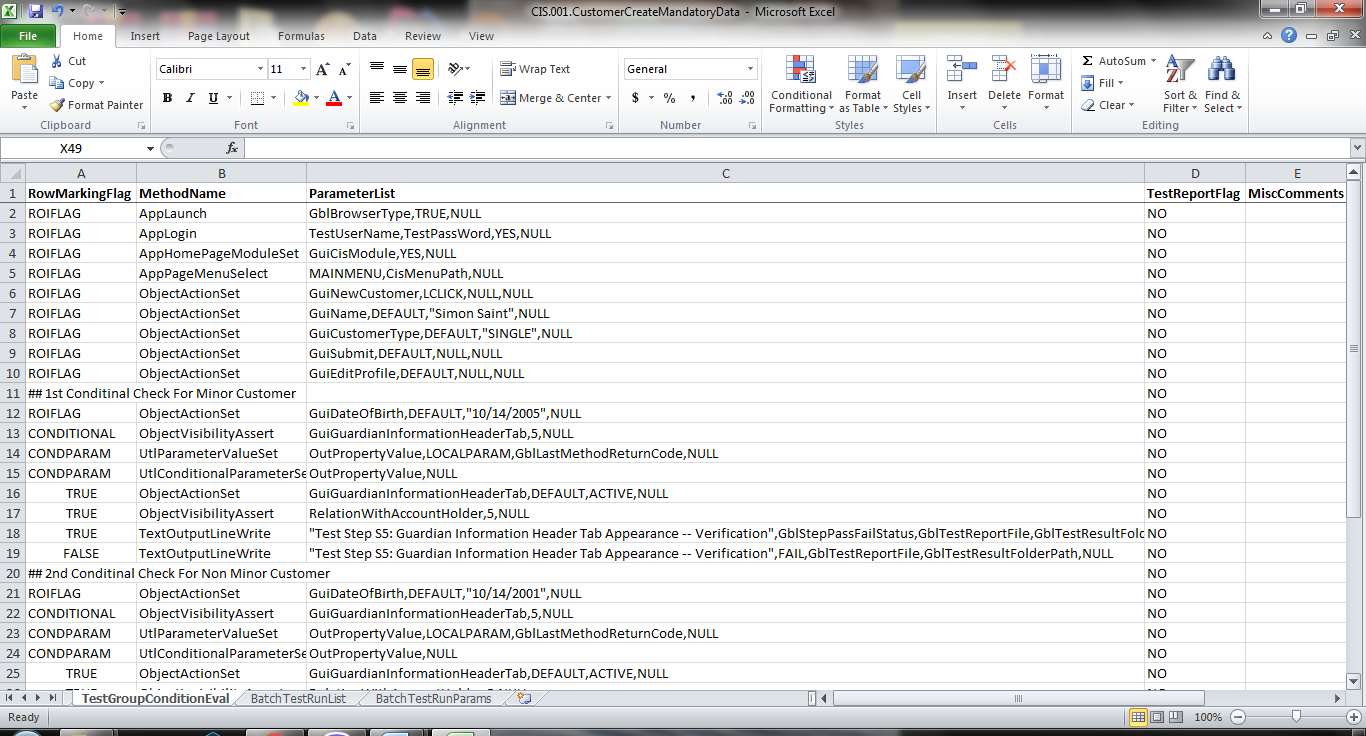
**TestGroupActionVerify Data Sheet:**



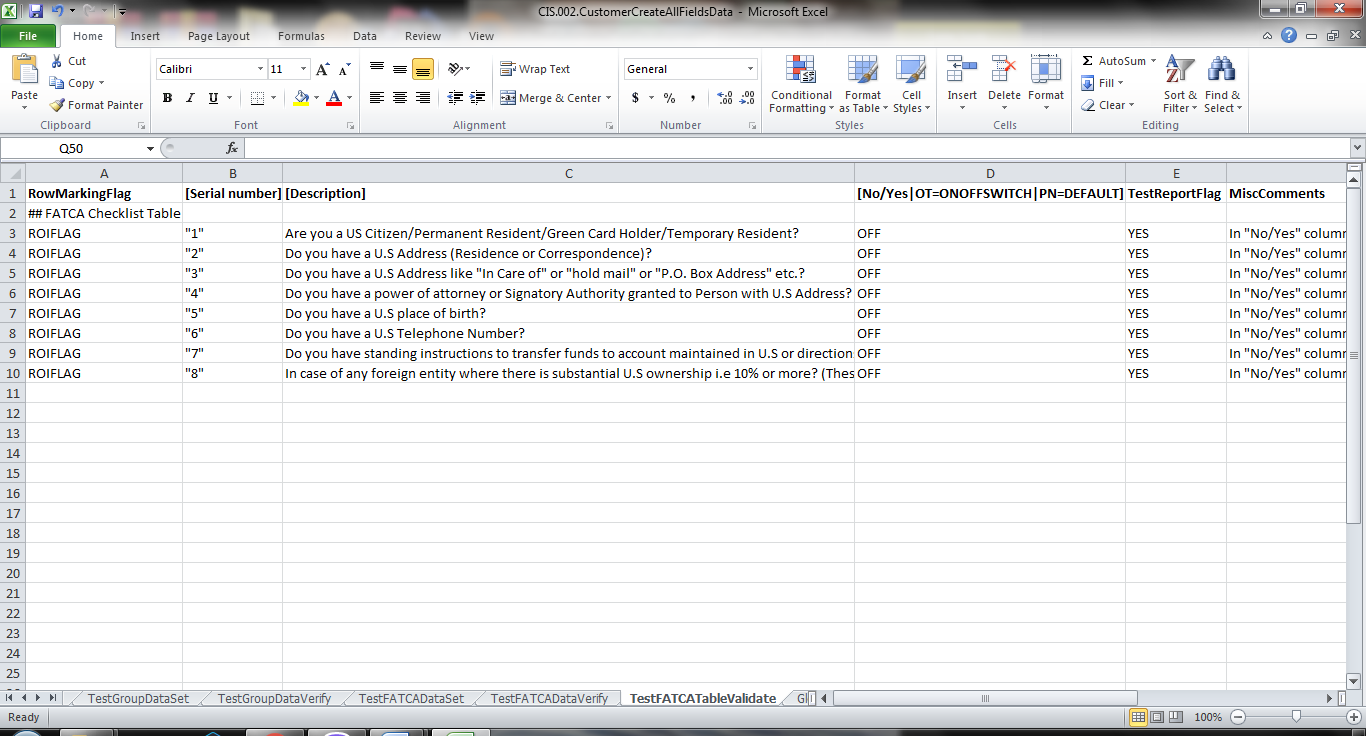
**TestGroupActionRepeat Data Sheet:**



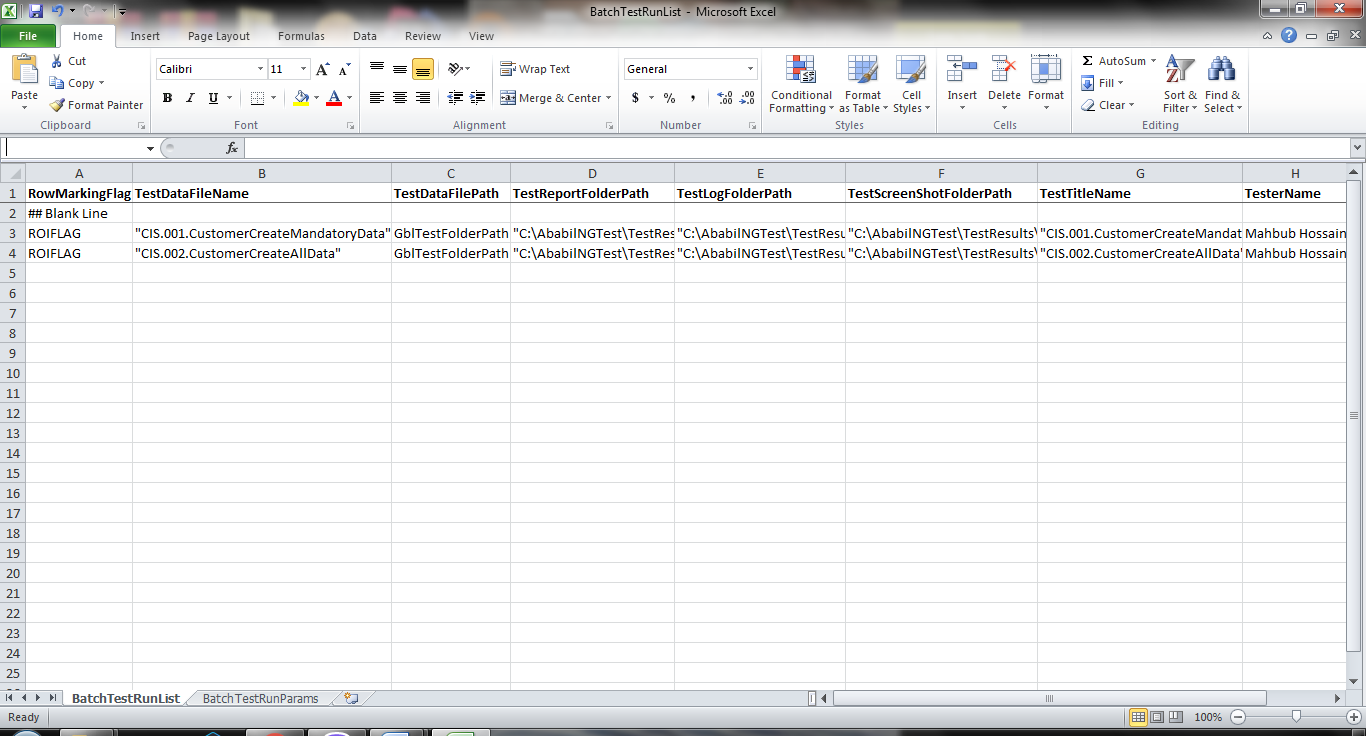
**TestGroupConditionCheck Data Sheet:**



**TestWebTableDataValidate Data Sheet:**



**BatchTestRunList Data Sheet:**



**BatchTestRunParams Data Sheet:**

