QTP Open Source Test Automation Framework for Java

Version 1.0

June 2009

TABLE OF CONTENTS

1. Purpose of the Document

The purpose of this document is to describe Open Source Test Automation Framework code in detail.

1.1. Scope

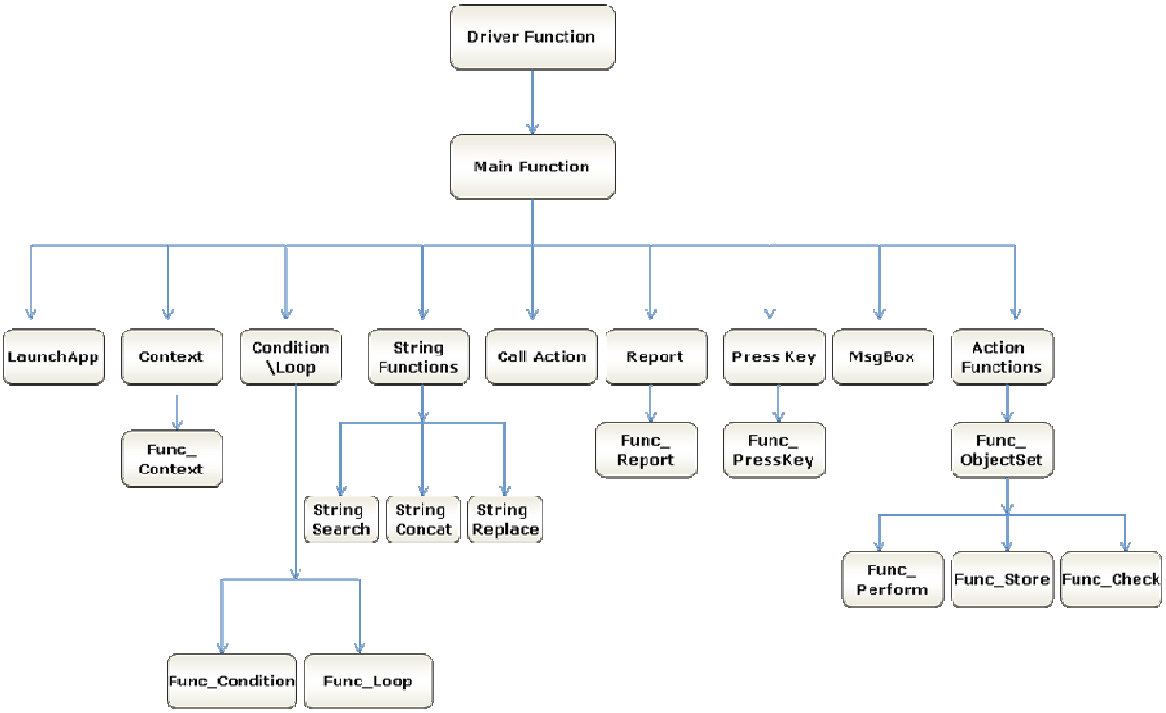
The scope of this document is to provide details about Open Source Test Automation Framework code.

1.2. Overview

This document provides details about:

- Framework Architecture
- Driver Functions
- Action Functions
- Reusable Functions
- Common Functions
- User-defined Functions

2. Framework Code Structure



3. Driver Functions

3.1. Keyword Driver Function

Name of the function: Keyword_Driver ()

Description: This function is used to call the main framework.

Parameters: NA

Assumptions: The automation script is present in the global sheet of HP QuickTest Professional (QTP).

Variables:

a) intRowCount – This variable is used as a counter to loop through all the data table rows.

b) intDataCounter – This variable is used to store the iteration count for looping.

c) intSheet – This variable is used to check whether the keyword script is present in the global sheet.

Functionality:

- The Driver Function reads the values in the first column of the global sheet.
- Whenever the value is 'r', it calls the main function (Keyword_Java).
- When 'r' is not present in any of the cells in the first column, it will skip the row and read the value in the next row.
- If the global sheet is empty then it reports fail, stating "Script is not present in the global sheet".

3.2. Main Function

Name of the function: Keyword_Java ()

Description: This is the Main Function, which interprets the keywords and performs the desired actions. All the keywords used in the data table are processed in this function.

Parameters: NA

Assumptions: The automation script is present in the global sheet of QTP

Variables:

a) initial – This variable is used to store the value in the second column.

b) objName – This variable is used to store the value in the third column.

c) arrObjchk() – This array is used to store the object type, name, and type of match.

d) arrAction() – This array is used to store the object type and name.

e) objPerform – This variable is used to store the value present in the fourth column.

f) arrKeyValue() – This array is used to store two values in the fourth column separated with delimiter ":".

g) arrKeyIndex() – This array is used to store all the values in the fourth column, separated by delimiter ":"

Functionality:

- The Main Function reads the values in the second, third, and fourth columns in the data sheet and stores the values into the variables (For more details please see the variables section.)
- Based on the value in the variable initial (second column), the Main Function calls different functions.
- If the value is other than 'perform', 'storevalue', or 'check', the Main Function calls the respective functions. For example, if the value is 'report', it will call Func_Report ()
- If the value is 'perform', 'storevalue', or 'check', the Main Function calls the function Func_ ObjectSet () to set the object and then it calls the respective functions. For example, if the value is 'perform', the Main Function will call Func_Perform() to perform required actions on the application under test (AUT).
- Refer to the framework code structure diagram for a better understanding of the function calls.

4. Action Functions

4.1. Perform Function

Name of the function: Func_Perform ()

Description: This function performs the set of actions on the required object in the AUT.

Parameters:

a) Object – This is the object on which the specified operation needs to be performed.

b) arrObj – This parameter holds the type of the object and the object name on which the action has to be performed.

c) arrKeyValue – This parameter identifies the operation that needs to be performed on the object.

d) arrKeyIndex – This additional parameter is required to identify the object where the operation needs to be performed. It holds the value of the specific action type and the value to be used, with row and column values for table operations.

e) intRowCount – This parameter holds the count of the current row in the data table.

Assumptions: Context is set on the current object where the action has to be performed.

Variables:

a) strStatus – This variable is used to store the split array of propSplit1(0) when delimiter '_' is used.

b) propSplit1 – This variable is used to store the split array of the fourth column of the data sheet when delimiter '_' is used.

c) VarName –This variable is used to store the value present in strStatus(0).

Functionality:

Based on the values in arrObj(0), Func_Perform() performs different actions on objects. If the value is:

1. Window

   Based on the values in arrKeyValue(0) the function performs different actions on objects. If the value in the variable arrKeyValue(0) is:

   a. close: Performs close operation on the parent object

   b. activate: Activates the parent object

   c. maximize: Maximizes the parent object

   d. minimize: Minimizes the parent object

   e. restore: Restores the parent object to its previous size

   f. textclick: Calls the function Func_SelectText () and clicks on the text specified in arrKeyValue(1)

2. dialog

Based on the values in arrKeyValue(0), the function performs different actions on objects. If the value in the variable arrKeyValue (0) is:

a. close: Performs close operation on the parent object

b. restore: Restores the parent object to its previous size

c. textclick: Calls the function Func_SelectText () and clicks on the text specified in arrKeyValue(1)

3. object

Based on the values in arrKeyValue(0) the function performs different actions on objects. If the value in the variable arrKeyValue (0) is:

a. type: Types the value present in arrKeyValue(1) in the current object

b. click: Performs click operation on the current object

4. button

Performs click operation on the current object

5. toolbar

Based on the values in arrKeyValue(0), the function performs different actions on objects. If the value in the variable arrKeyValue (0) is:

a. press: Performs a 'Press' operation on the toolbar item specified in the variable arrKeyValue(1)

6. spinner

Based on the values in arrKeyValue(0), the function performs different actions on objects. If the value in the variable arrKeyValue (0) is:

a. next: Sets the spin object to its next value using Next method

b. previous: Sets the spin object to its previous value using the Prev method

c. set: Sets the spin object to the value specified in arrKeyValue(1) using the Set method

d. click: Performs click operation on the current object

7. treeview

Based on the values in arrKeyValue(0), the function performs different actions on objects. If the value in the variable arrKeyValue (0) is:

a. expand: Expands the node specified in arrKeyValue(1) using the Expand method

b. expandall: Expands all the child items of the node specified in arrKeyValue(1) using the ExpandAll method

c. collapse: Collapses the node specified in arrKeyValue(1) using the Collapse method

d. select: Selects the node specified in arrKeyValue(1) using the Select method

e. check: Sets the treeview item to checked state

f. uncheck: Sets the treeview item to uncheck state

g. click: Sets the treeview item state to click

8. listview

   Based on the values in arrKeyIndex(0) the function performs different actions on objects. If the value in the variable arrKeyIndex(0) is:

   a. select: Selects the value in the variable arrKeyIndex(1) from the list

   b. extendselect: Selects an additional item specified in arrKeyIndex(1) from the list

   c. deselect: Deselects the value in the specified variable arrKeyIndex(1) from the list

   d. selectindex: Selects the value from the list using the index value specified in the variable arrKeyIndex(1)

   e. extendselectindex: Selects an additional value from the list using the index value specified in the variable arrKeyIndex(1)

   f. deselectindex: Deselects the value from the list using the index value specified in the variable arrKeyIndex(1)

   g. selectrange: Selects the range of items present between the value specified in arrKeyIndex(1) and the value specified in arrKeyIndex(2) from the list

   h. selectrangeindex: Selects the range of items present between the index value specified in arrKeyIndex(1) and the index value specified in arrKeyIndex(2) from the list

   i. check: Sets the listview item to a checked state

   j. uncheck: Sets the listview item to an unchecked state

   k. itemclick: Sets the listview item to a clicked state

   l. doubleclick: Sets the listview item to a double-clicked state

   m. activate: Activates the current object

   n. click: Performs click operation on the current object

9. menu

   Selects the menu item set in the context

10. combobox

    Based on the values in arrKeyValue(0) the function performs different actions on objects. If the value in the variable arrKeyValue(0) is:

    a. type: Types the value specified in variable arrKeyValue(1) in the current object

    b. select: Selects the item specified in arrKeyValue(1) from the current object using the Select method

    c. selectindex: Selects the value from the current object using the index value specified in the variable arrKeyValue(1)

11. listbox

    Based on the values in arrKeyIndex(0) the function performs different actions on objects. If the value in the variable arrKeyIndex(0) is:

    a. select: Selects the item specified in arrKeyIndex(1) from the current object using the Select method

b. selectindex: Selects the value from the current object using the index value specified in the variable arrKeyIndex (1)

c. selectrange: Selects the range of values from the current object using the index value specified in the variable arrKeyIndex (1) and arrKeyIndex (2)

d. selectrangeindex: Selects the range of items present between the index value specified in arrKeyIndex(1) and the index value specified in arrKeyIndex(2) from the current object

e. deselect: Deselects the value specified in the variable arrKeyIndex(1) from the current object

f. deselectindex: Deselects the value specified in the variable arrKeyIndex(1) from the current object

g. extendselect: Selects an additional item specified in arrKeyIndex(1) from the current object

h. extendselectindex: Selects an additional value from the list using the index value specified in the variable arrKeyIndex(1)

i. click: Performs click operation on the current object

j. activate: Activates the current selected item

12. winlistbox

Based on the values in arrKeyIndex(0) the function performs different actions on objects. If the value in the variable arrKeyIndex(0) is:

a. select: Selects the item specified in arrKeyIndex(1) from the current object using the Select method

b. selectindex: Selects the value from the current object using the index value specified in the variable arrKeyIndex (1)

c. selectrange: Selects the range of values from the current object using the index value specified in the variable arrKeyIndex (1) and arrKeyIndex (2)

d. selectrangeindex: Selects the range of items present between the index value specified in arrKeyIndex(1) and the index value specified in arrKeyIndex(2) from the current object

e. deselect: Deselects the value specified in the variable arrKeyIndex(1) from the current object

f. deselectindex: Deselects the value specified in the variable arrKeyIndex(1) from the current object

g. extendselect: Selects an additional item specified in arrKeyIndex(1) from the current object

h. extendselectindex: Selects an additional value from the list using the index value specified in the variable arrKeyIndex(1)

i. click: Performs click operation on the current object

13. weblistbox

Based on the values in arrKeyIndex(0) the function performs different actions on objects. If the value in the variable arrKeyIndex(0) is:

a. select: Selects the item specified in arrKeyIndex(1) from the current object using the Select method.

b. deselect: Deselects the value specified in the variable arrKeyIndex(1) from the current object

c. extendselect: Selects an additional item specified in arrKeyIndex(1) from the current object

d. click: Performs click operation on the current object

14. checkbox

Based on the values in arrKeyValue(0), the function performs different actions on objects. If the value in the variable arrKeyValue(0) is:

a. check: Performs check operation on the current object

b. uncheck: Performs uncheck operation on the current object.

15. radiobutton

Based on the values in arrKeyValue(0), the function performs different actions on objects. If the value in the variable arrKeyValue(0) is:

a. set: Performs set operation on the current object

b. click: Performs click operation on the current object

16. tab

Based on the values in arrKeyIndex(0), the function performs different actions on objects. If the value in the variable arrKeyIndex(0) is:

a. select: Selects the tab item specified in arrKeyIndex(1) from the current object using the Select method

b. click: Performs click operation on the current object

c. selectindex: Selects the value from the current object using the index value specified in the variable arrKeyIndex (1).

d. closetab: Performs close operation on the tab specified in the variable arrKeyIndex (1).

17. scrollbar

Based on the values in arrKeyValue(0), the function performs different actions on objects. If the value in the variable arrKeyValue(0) is:

a. nextline: Scrolls the current object to the right or down arrKeyValue(1) number of lines. If no value is specified in arrKeyValue(1) then the scroll operation is performed once.

b. nextpage: Scrolls the current object to the right or down arrKeyValue(1) number of pages. If no value is specified in arrKeyValue(1) then the scroll operation is performed once.

c. prevpage: Scrolls the current object to the left or up arrKeyValue(1) number of pages. If no value is specified in arrKeyValue(1) then the scroll operation is performed once.

d. prevline: Scrolls the current object to the left or up arrKeyValue(1) number of lines. If no value is specified in arrKeyValue(1) then the scroll operation is performed once.

e. set: Sets the current object to the position specified in arrKeyValue(1).

18. editor

Based on the values in arrKeyIndex(0), the function performs different actions on objects. If the value in the variable arrKeyIndex(0) is:

a. type: Types the value in variable arrKeyIndex(1) in the current object

b. setselection: Selects the text present in the specified location from the current object

c. setcaretpos: Places the cursor in the current object to the position specified in arrKeyIndex(1) and arrKeyIndex(2)

19. slider

Based on the values in arrKeyValue(0), the function performs different actions on objects. If the value in the variable arrKeyValue(0) is:

a. drag: Drags the current object to the value in the variable arrKeyValue(1)

20. link

Based on the values in arrKeyValue(0), the function performs different actions on objects. If the value in the variable arrKeyValue(0) is:

a. click: Clicks the current object in the co-ordinates mentioned in the variables arrKeyIndex(1), arrKeyIndex(2)

b. clicklink: Clicks the current object with the value in the variable arrKeyValue(1)

21. static

Based on the values in arrKeyValue(0), the function performs different actions on objects. If the value in the variable arrKeyValue(0) is:

a. click: Clicks the current object in the co-ordinates mentioned in the variables arrKeyIndex(1), arrKeyIndex(2)

b. doubleclick: Double clicks the current object with the value in the variables arrKeyIndex(1), arrKeyIndex(2)

22. textbox

Based on the values in arrKeyValue(0), the function performs different actions on objects. If the value in the variable arrKeyValue(0) is:

a. type: Types the value specified in arrKeyValue(1)

b. set: Sets the value specified in arrKeyValue(1) in the current object

Also it checks if the first two characters of arrKeyValue(1) is 'd_'. If true, then it stores the values of the variable arrObj(1) after splitting with the delimiter "_" into the variable propSplit1. It splits the value in propsplit(1) using delimiter ';' and stores it in variable 'strStatus',depending upon the value in strStatus(0), of current time, current date, specified date, specified month, or specified year typed in the current object.

If the first two characters of arrKeyValue(1) is not 'd_', then it types the value specified in arrKeyValue(1) in the current object.

c. click: Performs click operation on the current object

d. setcaretpos: Sets the cursor in the position mentioned in the variables arrKeyIndex(1) and arrKeyIndex(2)

e. setselection: Selects the text present in the specified location from the current object

f. doubleclick: Performs the doubleclick operation on the current object.

23. Wait

Makes the script wait for the specified value in arrKeyValue(1)

24. calendar

Based on the values in arrKeyValue(0) it performs different actions on objects. If the value in the variable arrKeyValue(0) is:

a. setdate: Sets the date in the current object to the value specified in the variable arrKeyValue(1)

b. settime: Sets the time in the current object to the value specified in the variable arrKeyValue(1)

c. click: Performs a click operation on the current object

- If the value of arrObj(0) is not among the above listed then the function will check for arrObj, which holds the value in the third column.

If the value of arrObj(0) is:

i) sqlexecute:

- Creates DB object dbConn
- Executes the query using the execute method of the DB object by passing the strSQL variable as an argument
- Closes the DB object

ii) sqlvaluecapture:

- Calls the function Func_gfQuery() by passing the argument arrObj(1)
- Stores the value returned by the function in the variable in the fourth column of the data table

iii) sqlcheckpoint:

- Sets the current row in the action sheet to 1
- Sets the TO property of the connection string
- Changes the DB Object's source(SQL) statement
- Executes the DB checkpoint

iv) sqlmultiplecapture:

- Sets the current row in the action sheet to 1
- Sets the TO property of the connection string
- Changes the DB Object's source(SQL) statement
- Executes the DB output checkpoint
- Returns the value from the function stored in the variables specified in the output checkpoint

4.2. Store Function

Name of the function: Func_Store ()

Description: This function is used to store any property of a particular object into a variable.

Parameters:

a) Object – This is the object on which the specified operation needs to be performed.

b) arrObj – This parameter holds the type of the object from which the property has to be stored and the object name.

Assumptions: Context is set on the current object where the action has to be performed.

Variables:

a) strPropName – This variable is used to store the name of the property.

b) arrPropSplit – This array holds the property name and variable name.

c) intGRowNum – This variable stores the row number.

d) intGColNum – This variable stores the column number.

Functionality:

- Based on the values in arrPropSplit(0) the function performs different actions. If the value is:

  i) itemscount: "items count" RO property of the object is stored into the variable in the fourth column in the data table (arrPropSplit(1)).

  ii) enabled: "disabled" RO property of the object is captured and converted to the Boolean data type. Then negation of the value retrieved is stored in the variable in the fourth column in the data table (arrPropSplit(1)).

  iii) columncount: "Columncount" property of the object is stored in the variable in the fourth column in the data table (arrPropSplit(1)).

  iv) rowcount: "Rowcount" property of the object is stored in the variable in the fourth column in the data table (arrPropSplit(1)).

  v) filename: "file name" RO property of the object is stored in the variable in the fourth column in the data table (arrPropSplit(1)).

  vi) imagetype: "image type" RO property of the object is stored in the variable in the fourth column in the data table (arrPropSplit(1)).

  vii) defaultvalue: "default value" RO property of the object is stored in the variable in the fourth column in the data table (arrPropSplit(1)).

  viii) exist: "exist" property of the object is stored in the variable in the fourth column in the data table (arrPropSplit(1)).

- If the value of arrPropSplit(0) is not among the above listed, then arrPropSplit(0) ROProperty value of the object is stored in the variable arrPropSplit(1).

4.3. Check Function

Name of the function: Func_Check()

Description: This function is used for all the checking operations to be performed on the AUT.

Parameters:

a) Object – This parameter is the object on which the specified operation needs to be performed.

b) arrObj – This parameter holds the type of object and the object name on which the check operation has to be performed.

c) arrKeyValue – This parameter holds the check or checks that need to be performed on the object.

d) arrKeyIndex – This parameter includes details of the fourth column such as the property that needs to be checked and the input value provided.

e) intRowCount – This parameter holds the count of the current row in the data table.

Assumptions: Context is set on the current object where the action has to be performed.

Variables:

a) strChkParameter – This variable is used to store the mode of checking. It can have either “exactchk” or “regexpchk” as the value. By default it is set to “exactchk”.

b) ActualValue –Stores the property value retrieved from the AUT

c) ExpectedValue – Stores the expected value to be returned from the AUT

d) strStatus – Internal variable used for reporting

e) iStatus – Stores the final status of the check (pass, fail, done, etc.)

f) reportStep – Stores the expected results as part of the report

g) reportStepPass – Stores the actual results as part of the report when the check passes

h) reportStepFail – Stores the actual results as part of the report when the check fails

i) result – Stores the return value (‘true’ or ‘false’) from some of the check operations

Functionality:

- Based on the values in arrKeyValue(0), the function performs different checks. If the value is:

  i) enabled: The value of the enabled property of the object is stored in ActualValue, which is then evaluated to be equal with ExpectedValue and an appropriate report is generated. ExpectedValue is the boolean conversion of the ‘true’ or ‘false’ value of arrKeyValue(1).

  ii) focused: The value of the focused property of the object is stored in ActualValue, which is then evaluated to be equal with ExpectedValue and an appropriate report is generated.

ExpectedValue is the boolean conversion of the 'true' or 'false' value of arrKeyValue(1).

iii) visible: The value of the visible property of the object is stored in ActualValue, which is then evaluated to be equal with ExpectedValue and an appropriate report is generated. ExpectedValue is the boolean conversion of the 'true' or 'false' value of arrKeyValue(1).

iv) itemscount: The value of the items count property of the object is stored in ActualValue, which is then evaluated to be equal with ExpectedValue and an appropriate report is generated. ExpectedValue is the integer conversion of arrKeyValue(1).

v) columncount: The value of the columncount property of the object is stored in ActualValue, which is then evaluated to be equal with ExpectedValue and an appropriate report is generated. ExpectedValue is the integer conversion of arrKeyValue(1).

vi) rowcount: The value of the RowCount property of the object is stored in ActualValue, which is then evaluated to be equal with ExpectedValue and an appropriate report is generated. ExpectedValue is the integer conversion of arrKeyValue(1).

vii) text: The value of the text property of the object is stored in ActualValue, which is then evaluated to be equal with ExpectedValue and an appropriate report is generated. ExpectedValue is the value of arrKeyValue(1).

viii) selection: The value of the selection property of the object is stored in ActualValue, which is then evaluated to be equal with ExpectedValue and an appropriate report is generated. ExpectedValue is the value of arrKeyValue(1).

ix) exist: The value of the exist property of the object is stored in ActualValue, which is then evaluated to be equal with ExpectedValue and a appropriate report is generated. ExpectedValue is the boolean conversion of the 'true' or 'false' value of arrKeyValue(1).

x) Checked: The value of the checked property of the object is stored in ActualValue, which is then evaluated to be equal with ExpectedValue and an appropriate report is generated. ExpectedValue is the UpperCase value of arrKeyValue(1).

xi) tabexist: The value specified in arrKeyValue(1) is searched in the entire item range of the Tab object and an appropriate report is generated.

xii) itemexist: The value specified in arrKeyValue(1) is searched in the entire item range of the object and an appropriate report is generated.

xiii) windowtext: The specified text is searched for in the currently displayed text on a window and an appropriate report is generated.

5. Functions for setting object

5.1. Function for setting the context:

Name of the function: Func_Context ()

Description: This function is used to set the full hierarchical path for the object on which some action is to be performed.

Parameters:

a) arrObj – This parameter holds the class of the object and the object name on which the context has to be set.

b) intRowCount – This parameter holds the count of the current row in the data table.

Assumptions: The AUT is already up and running.

Variables:

a) strReportData – This variable is used to store the contents of the fourth column of the current row in the global sheet.

b) arrChildCell – This variable is used to store the elements when strReportData is split with delimiter '::'.

c) arrFramed – This variable is used to store the elements according to the object type and name.

d) contextData – This variable is used to store the value present in the fourth column.

e) arrChild – This variable is used to store the child objects of the main window.

Functionality:

- Based on the values on arrObj(0), this function will set the context on different objects. If the value is:

  i) window: Sets the curParent object with “JavaWindow” Class with the name as the value of variable arrObj(1)

  ii) dialog: Sets the curParent object with “JavaDialog” Class with the name as the value of variable arrObj(1)

  iii) browser: Sets the curParent object with “Browser” Class with the name as the value of variable arrObj(1)

  iv) winwindow: Sets the curParent object with “Window” Class with the name as the value of variable arrObj(1)

  v) windialog: Sets the curParent object with “Dialog” Class with the name as the value of variable arrObj(1)

- arrChildCell is obtained by splitting contextData with delimiter “::” and arrChild is obtained by splitting the elements of arrChildCell with delimiter “;”. Based on the value of arrChild(0) the full context is set. If arrChild(0) is:

  i) window: Sets the curParent object (as received from above) as curParent.javawindow(arrChild(1)).

  ii) dialog: Sets the curParent object (as received from above) as curParent.javadialog(arrChild(1)).

iii) winwindow: Sets the curParent object (as received from above) as curParent.window(arrChild(1)).

iv) windialog: Sets the curParent object (as received from above) as curParent.dialog(arrChild(1)).

v) applet: Sets the curParent object (as received from above) as curParent.Javaapplet(arrChild(1))

vi) page: Sets the curParent object (as received from above) as curParent.Page(arrChild(1))

vii) frame: Sets the curParent object (as received from above) as curParent.Frame(arrChild(1))

viii) table: Sets the curParent object (as received from above) as curParent.JavaTable(arrChild(1))

ix) menu: Sets the curParent object (as received from above) as curParent.JavaMenu(arrChild(1))

x) activex: Sets the curParent object (as received from above) as curParent.ActiveX(arrChild(1))

- After the above two Select Case statements, the value of curParent that is obtained is stored in the parent and can be used in the next function 'func_ObjectSet()'.

5.2. Function for setting the object:

Name of the function: Func_ObjectSet ()

Description: This function is used to set the child object on which some action is to be performed.

Parameters:

a) arrObjChk – This parameter holds the class of the object and the object name on which the context has to be set.

b) intRowCount – This parameter holds the count of the current row in the data table.

Assumptions: The AUT is already up and running.

Variables:

a) curObjClassName – This variable is used to store the exact object name.

Functionality:

- The function will check for the value in the variable arrObjChk(0).
- If the value of arrObjChk(0) is:

i) window: Sets the object as the parent object obtained from the 'func_Context()' function

ii) dialog: Sets the object as the parent object obtained from the 'func_Context()' function

iii) browser: Sets the object as the parent object obtained from the 'func_Context()' function

iv) page: Sets the object as the parent object obtained from the 'func_Context()' function

v) frame: Sets the object as the parent object obtained from the 'func_Context()' function

vi) applet: Sets the object as the parent object obtained from the 'func_Context()' function

vii) activex: Sets the object as the parent object obtained from the 'func_Context()' function

viii) button: Sets the object with class JavaButton and name curObjClassName under the parent object obtained from the 'func_Context()' function

ix) calendar: Sets the object with class JavaCalendar and name curObjClassName under the parent object obtained from the 'func_Context()' function

x) checkbox: Sets the object with class JavaCheckbox and name curObjClassName under the parent object obtained from the 'func_Context()' function

xi) expandbar: Sets the object with class JavaExpandBar and name curObjClassName under the parent object obtained from the 'func_Context()' function

xii) link: Sets the object with class JavaLink and name curObjClassName under the parent object obtained from the 'func_Context()' function

xiii) listbox: Sets the object with class JavaList and name curObjClassName under the parent object obtained from the 'func_Context()' function

xiv) menu: Sets the object with class JavaMenu and name curObjClassName under the parent object obtained from the 'func_Context()' function

xv) object: Sets the object with class JavaObject and name curObjClassName under the parent object obtained from the 'func_Context()' function

xvi) radiobutton: Sets the object with class JavaRadiobutton and name curObjClassName under the parent object obtained from the 'func_Context()' function

xvii) slider: Sets the object with class JavaSlider and name curObjClassName under the parent object obtained from the 'func_Context()' function

xviii) spinner: Sets the object with class JavaSpin and name curObjClassName under the parent object obtained from the 'func_Context()' function

xix) static: Sets the object with class JavaStatic and name curObjClassName under the parent object obtained from the 'func_Context()' function

xx) tab: Sets the object with class JavaTab and name curObjClassName under the parent object obtained from the 'func_Context()' function

xxi) table: Sets the object with class JavaTable and name curObjClassName under the parent object obtained from the 'func_Context()' function

xxii) textbox: Sets the object with class JavaEdit and name curObjClassName under the parent object obtained from the 'func_Context()' function

xxiii) toolbar: Sets the object with class JavaToolbar and name curObjClassName under the parent object obtained from the 'func_Context()' function

xxiv) treeview: Sets the object with class JavaTreeView and name curObjClassName under the parent object obtained from the 'func_Context()' function

xxv) statusbar: Sets the object with class WinStatusBar and name curObjClassName under the parent object obtained from the 'func_Context()' function

xxvi) scrollbar: Sets the object with class WinScrollbar and name curObjClassName under the parent object obtained from the 'func_Context()' function

xxvii) editor: Sets the object with class WinEditor and name curObjClassName under the parent object obtained from the 'func_Context()' function

xxviii) winspinner: Sets the object with class WinSpin and name curObjClassName under the parent object obtained from the 'func_Context()' function

xxix) listview: Sets the object with class WinListView and name curObjClassName under the parent object obtained from the 'func_Context()' function

xxx) editor: Sets the object with class WinEditor and name curObjClassName under the parent object obtained from the 'func_Context()' function

xxxi) combobox: Sets the object with class WinComboBox and name curObjClassName under the parent object obtained from the 'func_Context()' function

xxxii) weblistbox: Sets the object with class Weblist and name curObjClassName under the parent object obtained from the 'func_Context()' function

xxxiii) winlistbox: Sets the object with class Winlist and name curObjClassName under the parent object obtained from the 'func_Context()' function

xxxiv) winstatic: Sets the object with class Static and name curObjClassName under the parent object obtained from the 'func_Context()' function

6. Reporting and Error Handling Functions

6.1. Reporting Function:

Name of the function: Func_Report ()

Description: This function is used for generating the customized report with specified user inputs through the keyword.

Parameters: None

Assumptions: NA

Variables:

a) reportobj – This variable is used to store the contents of the third column of the current row in the global sheet.

b) reportcon – This variable is used to store the status of the report (Pass/Fail).

c) reportcon1 – This variable is used to store the actual message of the report.

d) reporter0 – This variable is used to store the expected message of the report.

e) expmess– This variable is used to store the concatenated expected message.

f) actmess – This variable is used to store the concatenated actual message.

Functionality:

- 'reportobj' is split with delimiter ";" and is stored in the array 'reportcon'.
- 'reportcon(0)' holds the status of the report.
- 'reportcon(1)' is split with delimiter "::" and is stored in the array 'reportcon1'.
- 'reportcon1(0)' is split with the delimiter ":" and is stored in the 'reporter0'.
- 'reporter0' holds the expected message.
- 'reportcon1(1)' is split with the delimiter ":" and is stored in the 'reporter1'.
- 'reporter1' holds the actual message.
- Based on the values in the 'reportcon(0)' the function will generate different reports. If the value is:

xxxv) Pass:

  - Generates a report with status as Pass, expected message as 'reporter0', and actual message as 'reporter1'

xxxvi) Fail:

- Generates a report with status as Fail, expected message as 'reporter0', and actual message as 'reporter1'

xxxvii) Done:

- Generates a report with status as Done, expected message as 'reporter0', and actual message as 'reporter1'

xxxviii) Warning:

- Generates a report with status as Warning, expected message as 'reporter0', and actual message as 'reporter1'

6.2. Error-handling Function:

Name of the function: Func_Error ()

Description: This function is used to capture errors generated at runtime.

Parameters: NA

Assumptions: NA

Variables:

a) strError – This variable is used to store the value present in the fifth column of the current row in the data sheet.

Functionality:

This function checks for the Err.Number after processing each keyword line. Whenever the error number is not equal to '0', the function will generate a fail report. It also checks for the strError variable, which holds the value of the fifth column of the current row in the data table. Whenever the value is 'onfailureexit' and the value of the variable keyword is '1', the function will exit the test.

7. String and Regular Expression Functions

7.1. Function for string operations:

Name of the function: Func_StringOperations ()

Description: This function is used for all string operations.

Parameters:

a) strCriteria - This variable holds the value of the second column of the data table.

Assumptions: None

Variables:

a) arrSplit - This array is used to store the elements from the third column of the data table after splitting with ";" delimiter.

b) strMainString - This variable is used to store the main string (arrSplit(0)).

c) strSubString - This variable is used to store the sub string(arrSplit(1)).

d) intLen - This variable is used to store the length of the array "arrSplit".

e) ReturnVal - This variable is used to store the return value.

Functionality:

- Based on the values in strCriteria, the function performs different actions. If the value is:

  i) strsearch:

     1. Searches for the sub string (strSubString) in the main string (strMainString)
     2. Stores the position of the substring in the return value variable (ReturnVal)

  ii) strconcat:

     1. Concatenates the main string (strMainString) and the sub string (strSubString)
     2. Stores the concatenated string in the return value variable (ReturnVal)

  iii) strreplace:

     1. Searches for the sub string (strSubString) in the main string (strMainString) and replaces it with strString (arrSplit(2))
     2. Stores the replaced main string in the return value variable (ReturnVal)

- After the ReturnVal variable is updated, the value in the ReturnVal variable is stored in the variable specified in the fourth column of the data table.

7.2. Function for Regular Expression Test:

Name of the function: Func_gfRegExpTest ()

Description: This function conducts a regular expression test.

Parameters:

a) strPattern – This variable holds the pattern string to be searched for in the main string.

b) strString – This variable holds the main string.

Return Value: True/False

Assumptions: None

Variables:

a) objRegEx – This variable hold a regular expression object.

Functionality:

- A new regular expression object is created.
- The Pattern property of the above object is set with the value in the strPattern variable.
- The IgnoreCase property of the above object is set to 'False'.
- The test method of the object is invoked by passing the strString argument. This will execute a regular expression test, which returns 'True' or 'False'.

7.3. Function for Regular Expression Match:

Name of the function: Func_RegExpMatch ()

Description: This function executes a regular expression search against a specified string.

Parameters:

a) strPattern – This variable holds the pattern string.

b) strString – This variable holds the main string.

Return Value: This returns a Match collection when a regular expression search is performed. Reference parameters are used to return the start position (aIndex) and value (aValue).

Assumptions: None

Variables:

a) regEx – This variable holds a regular expression object.

b) Match – This variable holds the counter that can loop through the matches in 'Matches' variable.

c) Matches – This variable holds the collection of matches found in the main string.

Functionality:

- A new regular expression object is created.
- The Pattern property of the above object is set with the value in the strPattern variable.

- The IgnoreCase property of the above object is set to 'False'.
- The Global property of the above object is set to 'True'.
- The Matches object is set.
- aValue and aIndex variables hold the value and position, respectively.

7.4. Function for Regular Expression Replace:

Name of the function: Func_gfRegExpReplace ()

Description: This function replaces text found in a regular expression search.

Parameters:

a) strPattern – This variable holds the pattern string to be searched for and replaced.

b) strFind – This variable holds the main string in which the pattern string needs to be replaced.

c) strReplace – This variable holds the string that replaces the pattern string found in the main string.

Return Value: Returns replaced text

Assumptions: None

Variables:

a) regEx – This variable holds a regular expression object.

Functionality:

- A new regular expression object is created.
- The Pattern property of the above object is set with the value in the strPattern variable.
- The IgnoreCase property of the above object is set to 'False'.
- The Replace method of the object is invoked by passing strFind and strReplace arguments. This will execute a regular expression Find and Replace and return the replaced string.

8. Common Functions

8.1. Function for Retrieving Variables:

Name of the function: GetValue ()

Description: This function is used to retrieve the value from any variable.

Parameters:

a) strCellData – This variable holds the type of the object and the object name on which the action should be performed or checked.

Assumptions: NA

Variables:

a) arrSplitCheckData – This variable is used to store the row number in the table where the text is present.

b) strParamName – This variable is used to store the flag to check if text is found or not.

Functionality:

- This function searches for '#' in the variable strCellData. If '#' is present then the environment value in the variable is returned by the function.
- If '#' is not present then the variable strCellData is split with the delimiter "_" and is stored in the array arrSplitCheckData.
- Based on the values in the variable arrSplitCheckData(0), the function performs different actions. If the value is:

i) p: Returns the parameter value of the variable in arrSplitCheckData(0) returned by the function

ii) env: Retrieves the environment value in the variable arrSplitCheckData(0) returned by the function

iii) dt: Retrieves the value from the cell in the action 1 sheet with the column name, as in the variable arrSplitCheckData(0) and row, or as in the variable intDataCounter

8.2. Function for Press Key Operations:

Name of the function: Func_presskey ()

Description: This function is used to retrieve the value from any variable.

Parameters:

a) arrObj – This variable hold the value of the third column in the data table.

b) WshShell – This is the object created for shell scripting.

Assumptions: NA

Variables:

a) WshShell – This variable is used to store the row number in the table where the text is present.

b) sPresskey – This variable is used to store the flag to check if text is found or not.

Functionality:

- The WshShell object is created.
- Based on the values in arrObj(0), different values are passed as arguments to the SendKeys method of the created shell scripting object.
- Shell object is set to 'nothing'.

8.3. Function for Arithmetic Operations:

Name of the function: Func_arith ()

Description: This function is used to perform the addition (+) and subtraction (-) arithmetic operations.

Parameters:

a) strX – This variable is used to store the input values specified in the keyword script.

b) strY – This variable is used to store the output value of the function in a variable specified in the keyword script.

Assumptions: NA

Variables:

a) arrSplit1 – This variable is used to store the arithmetic equation.

b) intz – This variable is used to store the flag return value.

c) cellData – This variable is used to store arithmetic equations from the environment variable.

Functionality:

- This function will search for '#' in the variable strX. If '#' is not present then it will call the eval function, passing the variable strX as an argument. The return value is assigned to the environment variable in the fourth column of the data sheet.
- If '#' is present, then the function will search for '+', '*', '/', or '–' in the variable strX.
- If the Value in strX is :

i) '+': strSplit array is stored with the values in strX after splitting with the delimiter '+'. Then it will retrieve the environment values if they start with '#'. Then the strX variable is replaced with values in the variables strSplit(0) and strSplit(1). Then it will call the eval function by passing the variable strX as an argument. The return value is assigned to the environment variable in the fourth column of the data sheet

ii) '*': strSplit array is stored with the values in strX after splitting with the delimiter '*'. Then the function will retrieve the environment values if they start with '#'. Then the strX variable is replaced with values in the variables strSplit(0) and strSplit(1). Then the function will call the

eval function by passing the variable strX as an argument. The return value is assigned to the environment variable in the fourth column of the data sheet.

iii) '/': strSplit array is stored with the values in strX after splitting with the delimiter '/'. Then the function will retrieve the environment values if they start with '#'. Then the strX variable is replaced with values in the variables strSplit(0) and strSplit(1). Then it will call the eval function by passing the variable strX as an argument. The return value is assigned to the environment variable in the fourth column of the data sheet.

iv) '-': strSplit array is stored with the values in strX after splitting with the delimiter '-'. Then the function will retrieve the environment values if they start with '#'. Then the strX variable is replaced with values in the variables strSplit(0) and strSplit(1). Then it will call the eval function by passing the variable strX as an argument. The return value is assigned to the environment variable in the fourth column of the data sheet.

v) '^': strSplit array is stored with the values in strX after splitting with the delimiter '^'. Then the function will retrieve the environment values if they start with '#'. Then the strX variable is replaced with values in the variables strSplit(0) and strSplit(1). Then it will call the eval function by passing the variable strX as an argument. The return value is assigned to the environment variable in the fourth column of the data sheet.

8.4. Function for Querying Database:

Name of the function: Func_gfQuery ()

Description: This function is used to query the database.

Parameters:

a) strSQL – This holds the query that needs to be executed.

Assumptions: The connection string is specified and a connection is established with the database.

Variables:

a) dbConn – This variable is used to store the database connection object.

b) dbRS – This variable holds the results of the database operation performed.

c) connectionString – This variable stores the connection string for the database.

d) dbUID – This variable is used to store the user name to connect to the database.

e) dbPWD – This variable is used to store the password to connect to the database.

f) dbServer – This variable is used to store the database server name.

g) dbHost – This variable is used to store the database host name.

h) dbDRIVER – This variable is used to store the database driver name.

Functionality:

- Creates DB object dbConn
- Calls the method open for the dbConn object, passing the environment value of the variable connectionString as argument
- Calls the execute method by passing the variable strSQL as an argument
- Returns the items retrieved from the database after querying using the execute method
- Closes the object and sets dbConn to 'Nothing'

8.5. Function for Converting Data Types:

Name of the function: Func_Convert ()

Description: This function is used to query the database.

Parameters:

a) variable – This holds the variable that has to be converted.

b) strconverttype – This holds the data type into which the variable is to be converted.

Assumptions: NA

Variables:

a) strObject– This variable is used to store the variable to be converted.

b) arrConvert – This variable is used to store elements such as the conversion type and variable it is to be stored in.

c) strObject1 – This variable is used to store the converted value.

Functionality:

- This function searches for '#' in the variable strObject.
- If '#' is present, the strObject1 variable will be assigned with the environment value of strObject; Otherwise the strObject1 variable will be assigned with the value of the variable strObject.
- It will store the values in the variable strconverttype after splitting the delimiter ':' into the array arrConvert.
- Based on the values in arrConvert(0), the function will perform different actions. If the value is:

  i) 'date': It will convert the value in the strObject1 variable based on the format available in arrConvert(2).

  ii) 'roundto': It will round the value in the variable strObject1 and store it in the environment value of arrConvert(1).

  iii) 'lcase': It will convert the value in strObject1 into lowercase and store it in the environment value of arrConvert(1).

  iv) 'ucase': It will convert the value in strObject1 into uppercase and store it in the environment value of arrConvert(1).

v) 'cstr': It will convert the datatype of the value in strObject1 into the string datatype and store the converted value in arrConvert(1).

vi) 'ascii': It will convert the value in strObject1 into ASCII value and store the converted value in arrConvert(1).

vii) 'trim': It will remove the extra spaces in the value in the variable strObject1.

viii) 'len': It will return the length of the value in the variable strObject1 and store the returned length in the variable arrConvert(1).

ix) 'bln': It will convert the datatype of the value in strObject1 into the Boolean datatype and store the converted value in arrConvert(1).

8.6. Function for Importing Keyword Script from Excel Sheet

Name of the function: Func_ImportData ()

Description: This function is used to import test data at runtime.

Parameters:

a) strTestCase – This holds the file name and the path where it is stored.

Assumptions: The required file is present and it is an Excel sheet.

Variables:

a) strPath – This variable is used to store the path with .xls.

b) strDataPath – This variable is used to locate and store the full path.

c) strSheetName – This variable stores the sheet name to be imported.

Functionality:

- This function will store the value of the fourth column in the variable 'strSheetName'.
- Using the ImportSheet method, the Excel sheet with the name in the variable 'strSheetName' in the path specified by the variable 'strDataPath' is imported to the Action1 sheet.

8.7. Function for FSO Operations

Name of the function:

Func_CommonFunctions(strType,strDetails,intRowCount)

Description: This function is used to perform a set of operations using the file system objects.

Parameters:

a) strType –This holds the type of object being used for FSO such as File or Folder.

b) strDetails – This holds the details to be used while using FSO.

c) intRowCount – This holds the current row count in the data table.

Assumptions: NA

Variables:

a) strFuncType – This variable is used to store the type of object (Ex:File,Folder) to be used.

b) strFuncDetails – This variable is used to store the details of the object (Ex:Folder name,Folder Path).

Functionality:

- This function assigns the value in the variable strType to the variable strFuncType.
- This function assigns the value in the variable strDetails to the variable strFuncDetails.
- Based on the values in the variable strFuncType this function performs different actions. If the value is:

i) folder: It will call the function Func_Folder while passing the variable strFuncDetails as an argument.

ii) file: It will call the function Func_File while passing the variable strFuncDetails as an argument.

iii) exportxml: It will call the function Func_ExportXML while passing the variable strFuncDetails and the fifth column value in the data sheet as an argument.

iv) deletexml: It will call the function Func_DeleteXML while passing the variable strFuncDetails as an argument.

8.8. Function for Folder Operations

Name of the function: Func_Folder

Description: This function is used to work on folders using FSO.

Parameters:

a) pCellData – This holds the details to be used while using FSO.

Assumptions: NA

Variables:

a) arrFolderpath – This variable is used to store the elements of the folder path separated by delimiter "\".

b) intFolderlo – This variable is used to store the element number of the folder name.

c) DestFolder – This variable is used to store the destination folder.

d) Foldername – This variable is used to store the folder name.

e) oFSO – This variable is used to store the created object.

f) arrCellData – This variable is used to store the details of the operation to be performed

g) oFolder – This variable is used to store the details of the object created.

Functionality:

- The value in the variable pCellData is split with the delimiter “;” and is stored in the array arrCellData.

- Based on the values in the arrCellData[0] the function will perform different actions. If the value is:

i) create:

  - If the folder with the name arrCelldata[1] is present then a report is generated stating that the folder already exists.
  - If the folder is not present then a new folder with the name in arrCellData[1] is created.

ii) delete:

  - If the folder with the name arrCelldata[1] is not present then a report is generated stating that the folder is not present.

iii) If the folder is present then the folder is deleted.

iv) copy:

  - If the folder with the name arrCelldata[1] is not present then a report is generated stating that the folder does not exist.
  - If the folder is present it is copied to the location present in the variable arrCelldata[2].

v) move:

  - If the folder with the name arrCelldata[1] is not present then a report is generated stating that the folder does not exist.
  - If the folder is present it is moved to the location present in the variable arrCelldata[2].

8.9. Function for File Operations

Name of the function: Func_File

Description: This function is used for working with files by using FSO.

Parameters:

a) pCellData – This holds the details to be used while using FSO.

Assumptions: NA

Variables:

a) arrFilepath – This variable is used to store the file path.

b) DestFile – This variable is used to store the destination file name.

c) strFilename – This variable is used to store the file name to be used.

d) intFileLoc – This variable is used to store the element number of the file name.

e) iFSO – This variable is used to store the created object.

f) oFile – This variable is used to store the details of the created object.

g) arrCellData1 – This variable is used to store the details of the operation to be performed.

h) intf – This variable is used for looping.

i) stress – This variable is used to store the string, which has to be written into a file.

Functionality:

- The value in the variable pCellData is split with the delimiter ";" and is stored in the array arrCellData1.
- Based on the values in the arrCellData1[0] the function will perform different actions. If the value is:

i) create:

- If the file with the name arrCelldata1[1] is present then a report is generated stating that the file already exists.
- If the file is not present then a new file with the name in arrCellData1[1] is created.

ii) delete:

- If the file with the name arrCelldata1[1] is not present then a report is generated stating that the file is not present.
- If the file is present then the file is deleted.

iii) copy:

- If the file with the name arrCelldata1[1] is not present then a report is generated stating that the file does not exist.
- If the file is present it is copied to the location present in the variable arrCelldata1[2].

iv) move:

- If the file with the name arrCelldata1[1] is not present then a report is generated stating that the file does not exist.
- If the file is present it is moved to the location present in the variable arrCelldata1[2].

v) write:

- If the file with the name arrCelldata1[1] is not present then a report is generated stating that the file does not exist.
- If the file is present then required text is written in the file.

vi) read:

- If the file with the name arrCelldata1[1] is not present then a report is generated stating that the file does not exist.
- If the file is present then the required line is read from the file.

vii) append:

- If the file with the name arrCelldata1[1] is not present then a report is generated stating that the file does not exist.

- If the file is present then it opens the file in append mode (i.e., write = true).

8.10. Function for Exporting XMLs

Name of the function: Func_ExportXML

Description: This function is used to export data and store it in XML format.

Parameters:

a) strDetails1 – This holds the details to be used while exporting in XML format.

b) sPath – This holds the path where the XML file has to be stored.

Assumptions: NA

Variables:

a) into – This variable is used for looping.

b) oRoot – This variable is used to store the root element for the object.

c) arrDocSplit – This variable is used to store the elements to be exported to XML and the document name.

d) strDocName – This variable is used to store the document name.

e) arrElementSplit – This variable is used to store the variables and the tag names to be exported to XML.

f) arrElementName – This variable is used to store the current variable and its tag names to be exported to XML.

g) oDoc – This variable is used to store the XML object.

Functionality:

- The value in the variable strDetails1 is split with the delimiter ";" and is stored into the array arrDocSplit.
- XML file Object oDoc is created.
- The XML file with the name in the variable 'docname' is created by using the method CreateDocument.
- All the data present in arrDocSplit[1] is exported into the above created file by using the AddChildElementByName method.
- The XML file is saved in the path available in the variable 'sPath'.
- oDoc object is set to Nothing.

8.11. Function for Deleting XMLs

Name of the function: Func_DeleteXML

Description: This function is used to delete the XML files.

Parameters:

a) sPath – This holds the path in which the XML file is present.

Assumptions: NA

Variables:

a) dFileObj – This variable is used to store the XML file to be used.

b) dFSO – This variable is used to store the XML object.

Functionality:

- The File System Object 'dFSO' object is created.
- The XML file in the path specified in the variable 'sPath' is accessed using the GetFile method.
- Using the delete method, the above file is deleted. The dFSO object is set to Nothing.

9. Reusable Functions

9.1. Function for Calling Reusable Actions

Name of the function: Func_CallAction ()

Description: This function is used to call a reusable action.

Parameters:

a) strData – This holds the name of the reusable action.

b) strInfo – This holds the parameters for the reusable action.

Assumptions: NA

Variables:

a) arrParam – This variable is used to store the parameters to be passed to the reusable action.

b) strActionName – This variable is used to store the reusable action name.

Functionality:

- This function checks for the value in the variable strInfo.
- If the value is Null then the function will call the RunAction method by passing the variable 'actionName' and value 'oneIteration' as arguments (without passing parameters).
- If the value is not Null then the function will split the value in the variable strInfo with the delimiter ':' and store it in array 'paramSplit'.
- Based on the number of items in the array paramSplit, the function performs different actions.
- For example, if the number of items in paramSplit is 4, then 4 parameters are passed while calling the RunAction method.

10. Condition and Looping Functions

10.1. Condition function

Name of the function: Func_Condition ()

Description: This function is used to evaluate the expression according to the inputs given in the keyword script.

Parameters:

a) intRowCount – This holds the value of the current row number of the data table.

Assumptions: NA

Variables:

a) iFlag– This variable is used to set the flag.

b) cndSplit – This variable is used to store the value of the fourth column of the global sheet.

c) startRow – This variable is used to store the start row for the condition.

d) endRow – This variable is used to store the end row for the condition.

e) cstrCellData – This variable is used to store the condition to be checked.

f) var1 – This variable is used to store the first element to be evaluated.

g) var2 – This variable is used to store the second element to be evaluated.

Functionality:

- cndSplit array is stored with values in the fourth column of the data table after splitting with the delimiter “;”.
- The variable startRow is assigned with the value of the first item in the array cndSplit.
- The variable endRow is assigned with the value of the second item in the array cndSplit.
- The cstrCellData array is stored with values in the third column of the data table after splitting with the delimiter “;”.
- The variable var1 is assigned with the value of the first item in the array cstrCellData.
- The variable var2 is assigned with the value of the third item in the array cstrCellData.
- Then the condition in the cstrCellData(1) is evaluated and the intRowCount is assigned with the value in the startRow if the condition is true. Otherwise intRowCount is assigned with the value in the endRow if the condition is false.

10.2. Loop Function

Name of the function: Func_loop ()

Description: This function is used to repeat a set of statements for a specified number of times.

Parameters:

a) variable – This holds the query variable that has to be converted.

b) strconverttype – This holds the data type into which the variable is to be converted.

Assumptions: If the number of times to be looped is not specified, by default this number is taken as the number of active rows in the Action1 sheet of the data table.

Variables:

a) arrloopData – This variable is used to store the start row and end row values.

b) intcntr – This variable is used to store the loop count.

c) Counter – This variable is used to store the count value.

d) endRow1 – This variable stores the end row for looping.

e) loopRowCount – This variable stores the current loop count.

f) intDataCounter – This variable stores the current data counter.

Functionality:

- arrloopData is stored with the values after splitting the value in the third column of the data table with “;” as the delimiter.
- The variable intcntr is assigned with the value in the fourth column in the data table.
- The value in the variable intcntr is converted into an integer and stored in the variable counter.
- The value in the variable intcntr is stored into the variable counter.
- This function recursively calls the Keyword_Web function (Main function) ‘n’ times. Here ‘n’ is the value present in the variable counter.

11. Debug Functions

11.1. Function for Debugging:

Name of the function: DebugFunc ()

Description: This function is used while debugging the keyword script.

Parameters:

a) StartLineNumber – This holds the start line of the execution.

b) EndLinenumber – This holds the end line of the execution.

c) PrintOption – This holds the option to provide the QTP print dialog with the environment variables.

d) LogFile – This holds the option to log all the environment variables in a test file in the desktop.

Assumptions: The environment variables are not stored in a XML file and then loaded when debugging because the QTP script might have an existing XML file associated with it.

Variables:

a) strText– This variable is used to store the contents of the log file.

b) ostrFile – This variable is used to store the file object.

c) oFSO – This variable is used to store the file system object.

d) strDesktop – This variable stores the path of the desktop.

e) oQtApp – This variable stores the QTP Automation Object Model (AOM).

f) oWshShell – This variable stores the windows shell object.

g) strContents – This variable stores the contents of the log file.

h) arrSplit2 – This variable stores the array.

i) VarName – This array stores the split contents of each line in the log file.

j) strVarFullName – This array stores the value of each item in the array VarName.

k) strVariableName – This variable stores the environment variable name.

l) strVariableVal – This array stores the split contents of all the environment variable values in the log file.

m) strVariableValue – This variable stores the environment variable values.

n) strFileName – This variable stores the file name.

o) EnvSplit – This array stores the value of the environment variable mentioned in the test script data sheet.

p) VariableName – This variable stores the variable name.

Functionality:

- This function will check if the log file is required for the debugging, and then load all the environment variables to the QTP.
- This function creates a shell object to access the desktop.
- This function stores the value of the desktop path.
- This function creates a file system object.
- This function creates a QTP application object to access the test name.
- This function stores the file name in a variable.
- If the file exists, then it will load the variables in QTP into the file created above.
- If the user requires the QTP print option, then select “to print” for the new environment variables.
- If the log file is required for debugging, then it will store all the environment variables to the log file.
- This function opens the file for appending.
- This function writes the environment variables.
- This function clears all the object variables.