QTP Open Source Test Automation Framework for .NET

Version 1.0

May 2009

TABLE OF CONTENTS

1. Purpose of the Document

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

1.1. Scope

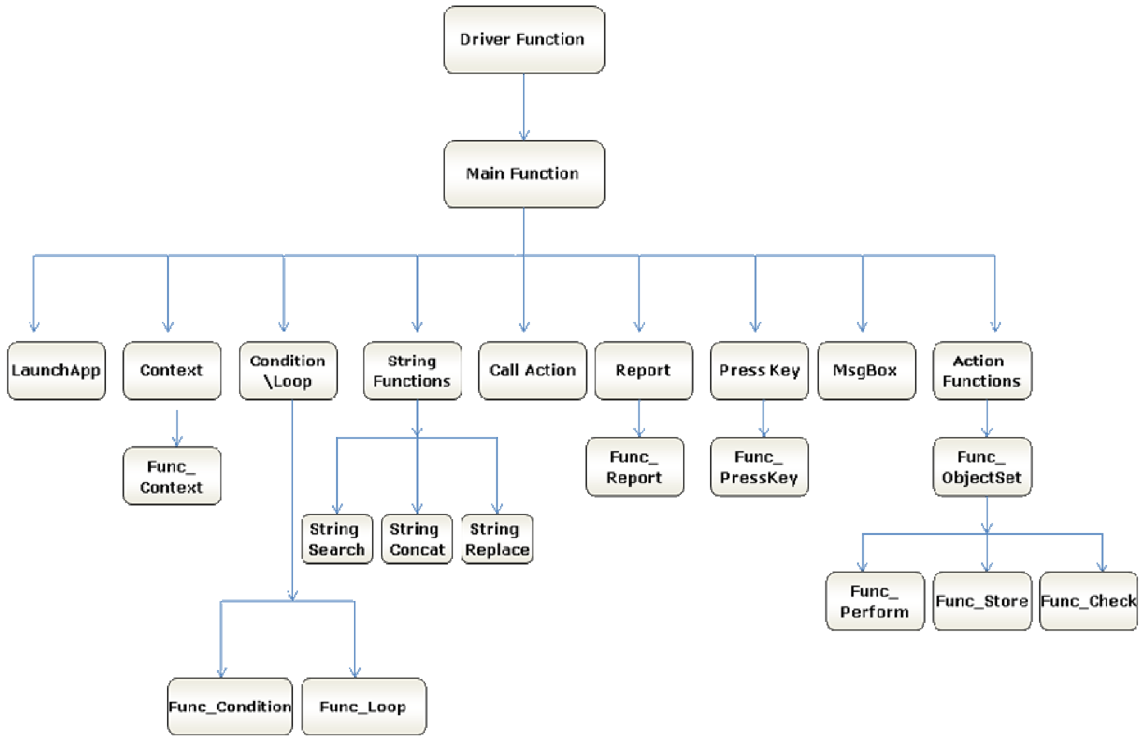
The scope of this document is to provide details about the Open Source Test Automation Framework code with its architecture and functions.

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 QTP.

Variables:

a) intRowCount - Loops through all the data table rows

b) intDataCounter - Stores the iteration count for looping

c) intSheet - Used as a flag to check for the presence of the keyword script in the Global Sheet

Functionality:

- Reads the values in the first column of the Global Sheet
- Whenever the value is 'r', it calls the main function (Keyword_Swf).
- When 'r' is not present in any of the cells in the first column, it skips the row and reads 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, please verify the Data Table".

3.2. Main Function

Name of the function: Keyword_Swf ()

Description: This is the main function, which interprets the keywords and performs the desired actions. All the keywords used in the Global Datasheet are processed in this function.

Parameters: NA

Assumptions: The Automation Script is present in the Global Sheet of HP QuickTest Professional (QTP).

Variables:

a) initial - Stores the value in the second column of the Global Data Sheet

b) objName - Stores the value in the third column of the Global Data Sheet

c) arrAction() - Stores the object type and name

d) objPerform - Stores the value present in the fourth column of the Global Data Sheet

e) arrKeyValue() - Stores two values in the fourth column, separated with delimiter ":"

f) arrKeyIndex() -Stores all the values in the fourth column, separated by delimiter ":"

Functionality:

- Reads the values in the second, third, and fourth columns in the Global Data Sheet and stores the value into the variables (Please refer to details in the Variables section.)
- Based on the value in the variable initial (second column), it calls different functions
- If the value is other than 'perform', 'storevalue', or 'check', the Keyword_Swf ()calls the respective functions. For example, if the value is 'report', it will call Func_Report ().
- If the value is 'perform', 'storevalue', or 'check', it calls the function Func_ObjectSet () to set the object and then it calls the respective functions. For example if the value is 'perform', **it will call Func_Perform() to perform the required action against the application under test (AUT)**.

Refer to the framework code structure diagram to learn more about 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 refers to the object on which the specific operation needs to be performed.

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

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

d) arrKeyIndex – This is an additional parameter that is required in some of the cases to identify the object where the action needs to be performed. For ex: It holds the value of the specific action type or the row and column values for table operations.

e) intRowCount – This holds the current row number in the Global Datasheet.

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

Variables:

a) curtime :Stores the current time

b) strParam – Stores the value of the fifth column of the data table

c) strSQL : Stores the SQL query to be executed

d) strReplace : Replaces the variables with their values in the SQL query and stores the same

e) dbConn : Database connection object

f) dbRs : Object used to execute the query

g) intNum – Stores the value in arrObj(1)

h) strvar – Stores the string value in the fourth column of the Global Datasheet

i) strsplit – Stores the array after a split operation

j) strlen – Stores the string value in the fifth column of the Global Datasheet

k) strstore1 – Stores the elements present in the fourth column

l) arrVals – Stores the split array elements

m) strvarstore – Stores the split element

n) intval : Stores the integer value of the element of the array after a split operation. It needs to be stored in the specified variable.

Functionality:

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

i) activate: Activates the current parent object that is stored in variable curParent

ii) activateitem: Checks for the value in arrObj(0). If the value is 'listview' or 'listbox', it activates the item stored in arrKeyValue(1).

iii) activatecell: Checks for the value in arrObj(0). If the value is 'table', it activates a particular cell of the table with the row number specified by arrKeyValue(1) variable and the column number specified by arrKeyValue(2) variable.

iv) close: Checks for the value in arrObj(0). If the value is 'window' or 'dialog', it restores or activates and then closes it.

v) click: Performs click operation on the current object.

vi) clickitem: Checks for the value in arrObj(0). If the value is 'listview', it activates the item specified by arrKeyValue(1) variable.

vii) collapse: Checks for the value in arrObj(0). If the value is 'treeview', it collapses the tree.

viii) check: Checks for the value in arrObj(0). If the value is 'treeview' or 'listview', it checks the item specified by arrKeyValue(1) variable. If the value is 'checkbox' the object is checked.

ix) deselect: Checks for the value in arrObj(0). If the value is 'listview' or 'listbox', it deselects the item specified by arrKeyValue(1) variable.

x) deselectindex: Checks for the value in arrObj(0). If the value is 'listview' or 'listbox', it deselects the item with the index number specified by arrKeyValue(1) variable.

xi) doubleclick: Checks for the value in arrObj(0). If the value is 'listview', it double clicks the item specified by arrKeyValue(1) variable or if arrObj(0) is 'textbox' it clicks the object.

xii) expand: Checks for the value in arrObj(0). If the value is 'treeview', it expands the node specified by arrKeyValue(1) variable.

xiii) expandall: Checks for the value in arrObj(0). If the value is 'treeview', it expands the current node, specified by arrKeyValue(1) variable, as well as its subnodes.

xiv) extendselect: Checks for the value in arrObj(0). If the value is 'listview' or 'listbox', it selects the item specified by arrKeyValue(1) variable keeping the previous selection intact.

xv) extendselectindex: Checks for the value in arrObj(0). If the value is 'listview' or 'listbox', it selects the item with the index number specified by arrKeyValue(1) variable, keeping the previous selection intact.

xvi) maximize: Checks for the value in arrObj(0). If the value is 'window', 'dialog', or 'popupwindow', it checks if the object is 'enabled' and 'maximizable'. If both the conditions are met the object is maximized.

xvii) minimize: Checks for the value in arrObj(0). If the value is 'window', 'dialog', or 'popupwindow', it checks if the object is 'enabled' and 'minimizable'. If both the conditions are met the object is minimized.

xviii) next: Checks for the value in arrObj(0). If the value is 'spinner', it takes the spinner to the next value.

xix) nextline: Checks for the value in arrObj(0). If the value is 'scrollbar', it takes the scrollbar to the next position.

xx) nextpage: Checks for the value in arrObj(0). If the value is 'scrollbar', it takes the scrollbar to the next page.

xxi) press: Checks for the value in arrObj(0). If the value is 'toolbar', the item specified by th

xxii) e arrKeyValue(1) variable is pressed.

xxiii) previous: Checks for the value in arrObj(0). If the value is 'spinner', it takes the spinner to the previous value.

xxiv) prevpage: Checks for the value in arrObj(0). If the value is 'scrollbar', it takes the scrollbar to the previous page.

xxv) prevline: Checks for the value in arrObj(0). If the value is 'scrollbar', it takes the scrollbar to the previous position.

xxvi) restore: Checks for the value in arrObj(0). If the value is 'window', 'dialog', or 'popupwindow', it restores the object to its original dimension.

xxvii) showdropdown: Checks for the value in arrObj(0). If the value is 'toolbar' the child-items of the item specified by arrKeyValue(1) variable are displayed.

xxviii) set: Checks for the value in arrObj(0). If the value is 'scrollbar' the object is set to the position specified by arrKeyValue(1) variable. If the value is 'textbox' the text specified by arrKeyValue(1) variable is set in the object. If the value is 'combobox' the item specified by arrKeyValue(1) variable is selected and if the value is 'radiobutton' the object is selected.

xxix) select: Checks for the value in arrObj(0). If the value is 'toolbar', 'treeview','listview', 'menu'. 'combobox', 'listbox', or 'tab', the item specified by arrKeyValue(1) is selected.

xxx) selectindex: Checks for the value in arrObj(0). If the value is 'listview', 'combobox', 'listbox', or 'tab', the item with the index number specified by arrKeyValue(1) is selected.

xxxi) selectrange: Checks for the value in arrObj(0). If the value is 'listview' or 'listbox' the range of items between the two items specified by arrKeyIndex(1) and arrKeyIndex(2) variables is selected.

xxxii) selectrangeindex: Checks for the value in arrObj(0). If the value is 'listview' or 'listbox' the range of items between the two items with index numbers specified by arrKeyIndex(1) and arrKeyIndex(2) variables is selected

xxxiii) setselection: Checks for the value in arrObj(0). If the value is 'editor' the text, enclosed between the starting line number and column number (specified by arrKeyIndex(1) and arrKeyIndex(2) variables) and ending line number and column number (specified by arrKeyIndex(3) and arrKeyIndex(4) variables), is selected. If the value is 'textbox' then the text enclosed between the positions specified by arrKeyIndex(1) and arrKeyIndex(2) variables is selected.

xxxiv) setcaretpos: Checks for the value in arrObj(0). If the value is 'editor' the cursor is positioned at a line number specified by arrKeyIndex(1) and column number specified by arrKeyIndex(2) variables. If the value is 'textbox' then the cursor is positioned at the position specified by arrKeyIndex(1) variable.

xxxv) setdate: Checks for the value in arrObj(0). If the value is 'calendar' the date specified by arrKeyValue(1) variable is set.

xxxvi) settime: Checks for the value in arrObj(0). If the value is 'calendar' the time specified by arrKeyValue(1) variable is set.

xxxvii) tablesearch: Checks for the value in arrObj(0). If the value is 'table' the function func_tablesearch is called with object, arrKeyValue(1), strParam1(0) and strParam1(1) as arguments.

xxxviii) type: The value specified by arrKeyValue(1) is typed on the object.

xxxix) textclick: The function Func_SelectText is called with arrKeyIndex(1) as argument.

xl) uncheck: Checks for the value in arrObj(0). If the value is 'treeview' or 'listview', it unchecks the item specified by arrKeyValue(1) variable. If the value is 'checkbox' the object is unchecked.

- If the value of arrKeyValue(0) is not among the above listed then it will check for arrObjm 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 DB object by passing strSQL variable as argument
  - Closes the DB object

ii) sqlvaluecapture:

  - Calls the function Func_gfQuery() by passing the argument arrObj(1)

    Note: Func_gfQuery is explained in section 9.5 of this document

  - Stores the value returned by the function in the variable in the fourth column of the data table

iii) sqlcheckpoint:

  - Sets the TO property of the connection string
  - Changes the DB objects source(SQL) statement
  - Executes the DB checkpoint

iv) sqlmultiplecapture:

  - Sets the TO property of the connection string
  - Changes the DB objects source(SQL) statement
  - Executes the DB output checkpoint

- The operation returns the values of the variables specified while defining the DB Output Checkpoint

v) random:

- Stores the arrObj(1) value in the variable intNum
- Stores the value in the fourth column in the variable strvar
- Calls the Rnd function, passing the intNum variable as an argument
- Stores the return value from the function in the environment variable in the fourth column of the Excel sheet

vi) split:

- Stores the values of the variable arrObj(1) after splitting with the delimiter “^” in the variable strvar
- Each item in the array is checked whether it starts with # or not. If it has # then the environment value of that item is assigned to that variable.

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 an object on which the specified operation needs to be performed.

b) arrObj - This holds the type of the object and the name of the object from which the property has to be retrieved.

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

Variables:

a) strPropName - Stores the name of the property

b) arrPropSplit - Holds the property and variable names

c) intGRowNum - Stores the row number

d) intGColNum - Stores the column number

e) VarName - Stores the variable name where the value needs to be stored.

Functionality:

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

i) itemscount: Checks for the value in arrObj(0). If the value is ‘toolbar’ or ‘treeview’ the GetItemsCount property of the object and if the value is anything else the “items count” RO property of the object is stored in 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 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) maxlength: "max length" RO property of the object is stored in the variable in the fourth column in the data table (arrPropSplit(1)).

ix) allitems: Checks for the value in arrObj(0). If the value is 'toolbar' or 'treeview' the GetContent property of the object and if the value is anything else the "all items" RO property of the object is stored in the variable in the fourth column in the data table (arrPropSplit(1)).

x) selectiontype: "select type" RO property of the object is stored in the variable in the fourth column in the data table (arrPropSplit(1)).

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

xii) itemexist: Checks for the value in arrObj(0). If the value is 'toolbar' ItemExists property of the object is stored in the variable in the fourth column in the data table (arrPropSplit(1)).

xiii) selection: Checks for the value in arrObj(0). If the value is 'toolbar' or 'treeview' the GetSelection property of the object and if the value is anything else the "selection" RO property of the object is stored in the variable in the fourth column in the data table (arrPropSplit(1)).

xiv) selectioncount: "selected items count" RO property of the object is stored in the variable in the fourth column in the data table (arrPropSplit(1)).

xv) getcelldata: "GetCellData" method of the object is invoked by passing intGRowNum (row number) and intGColNum (column number) variables as arguments. The return value is stored in the variable in the fourth column in the data table (arrPropSplit(1)).

xvi) tablesearch: Checks for the value in arrObj(0). If the value is 'table' the function func_tablesearch is called with object, arrKeyValue(1), strParam1(0), and strParam1(1) as arguments, which return a True or False value based on the search operation performed on the table. This return value is stored in the variable specified in the fifth column.

- If the value in arrPropSplit(0) does not have any of the values listed above then arrPropSplit(0) ROProperty of the object is stored in the value 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 is the object on which the specified operation needs to be performed.

b) arrAction : This stores the elements after the value in the third column of the data table is split with delimiter ';'

c) arrKeyValue - This is the operation that needs to be performed on the object.

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

e) intRowCount - This 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) ActualValue -Stores the property value retrieved from the AUT

b) ExpectedValue - Stores the expected value to be returned from the AUT

c) strStatus - Stores a particular string based on the actual/expected property value during the check operation. This string is used in the report statement.

d) iStatus - Stores the final status of the check (pass, fail, done, etc.)

e) reportStep - Stores the expected results as part of the report

f) reportStepPass - Stores the actual results as part of the report when the check passes

g) reportStepFail - Stores the actual results as part of the report when the check fails

h) strStatus1 - Stores a particular string based on the actual/expected property value during the check operation. This string is used in the report statement

i) itemfound - Used as a flag in some of the exist checks.

j) actualcount - Stores the count of items in certain checks.

Functionality:

- Based on the values in arrKeyIndex(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 arrKeyIndex(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 arrKeyIndex (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 arrKeyIndex (1).

iv) itemcount: 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 arrKeyIndex(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 arrKeyIndex(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 arrKeyIndex(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 arrKeyIndex(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 arrKeyIndex(1)

ix) <u>exist</u>: The value of the exist 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 arrKeyIndex(1).

x) <u>checked</u>: 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 arrKeyIndex(1).

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

xii) <u>tabnotexist</u>: The value specified in arrKeyIndex(1) is searched in the entire item range of the Tab object and an appropriate report is generated.

xiii) <u>itemexist</u>: The value specified in arrKeyIndex(1) is searched in the entire item range of the object and an appropriate report is generated.

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

xv) itemspresent: The value specified in arrKeyIndex(1) is searched in the entire item range of the object and an appropriate report is generated.

xvi) table: The specified table is checked whether empty/not empty and an appropriate report is generated.

xvii) tablesearch: Checks for the value in arrObj(0). If the value is 'table' the function func_tablesearch is called with object, arrKeyValue(1) as arguments, which return a pass or fail report based on the search operation performed on the table.

- Based on the values in the PropertyVal variable, the Func_Check() function will call different RO properties of the current object and compare them with the value in the arrKeyIndex(1) variable. If the values match then it will generate a pass report. Otherwise it will generate a fail report.

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 variable holds the type of the object and the object name on which the context has to be set.

b) intRowCount - This variable holds the count of the current row in the data table.

Assumptions: The AUT is already up and running.

Variables:

a) strReportData - Stores the contents of the fourth column of the current row in the Global Sheet

b) arrChildCell - Stores the elements separated by the delimiter '::'

c) arrFramed - Stores the elements according to the object type and name

d) contextData - Stores the value present in the fourth column

e) arrChild - Stores the child objects of the main window

Functionality:

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

  i) window: Sets the curParent object with class as "SwfWindow" and name as the value of variable arrObj(1)

  ii) dialog: Sets the curParent object with class as "Dialog" and name as the value of variable arrObj(1)

  iii) browser: Sets the curParent object with class as "Browser" and name as the value of variable arrObj(1)

  iv) popupwindow: Sets the curParent object with class as "Window" and name as the value of variable arrObj(1)

  v) vbwindow: Sets the curParent object with class as "vbWindow" and name as the value of variable arrObj(1)

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) arrObj - Holds the type of the object and the object name on which the context has to be set

b) intRowCount - Holds the count of the current row in the data table

Assumptions: The AUT is already up and running.

Variables:

a) curObjClassName – Stores the current object name

b) ObjectVal – Stores the Table object Class name

Functionality:

- This function will check for the value in the variable arrObj(0). If the value is "split", "random", "sqlvaluecapture", "sqlexecute", "sqlcheckpoint" or "sqlmultiplecapture" it will generate a fail report.
- If the value is:

i) frame: Sets the frame with the value in the variable curObjClassName

ii) window: Sets the window object that is stored in variable parent

iii) dialog: Sets the dialog object that is stored in variable parent

iv) listbox: Sets the 'object' object with SwfList class and name as the value in the variable curObjClassName

v) spinner: Sets the 'object' object with SwfSpin class and name as the value in the variable curObjClassName

vi) toolbar: Sets the 'object' object with SwfToolbar class and name as the value in the variable curObjClassName

vii) wintoolbar: Sets the 'object' object with WinToolbar class and name as the value in the variable curObjClassName

viii) button: Sets the 'object' object with SwfButton class and name as the value in the variable curObjClassName

ix) winbutton: Sets the 'object' object with WinButton class and name as the value in the variable curObjClassName

x) vbbutton: Sets the 'object' object with vbButton class and name as the value in the variable curObjClassName

xi) treeview: Sets the 'object' object with SwfTreeView class and name as the value in the variable curObjClassName

xii) label: Sets the 'object' object with SwfLabel class and name as the value in the variable curObjClassName

xiii) listview: Sets the 'object' object with SwfListView class and name as the value in the variable curObjClassName

xiv) menu: Sets the 'object' object with WinMenu class and name as the value in the variable curObjClassName

xv) object: Sets the 'object' object with SwfObject class and name as the value in the variable curObjClassName

xvi) textbox: Sets the 'object' object with SwfEdit class and name as the value in the variable curObjClassName

xvii) wintextbox: Sets the 'object' object with WinEdit class and name as the value in the variable curObjClassName

xviii) editor: Sets the 'object' object with SwfEditor class and name as the value in the variable curObjClassName

xix) checkbox: Sets the 'object' object with SwfCheckbox class and name as the value in the variable curObjClassName

xx) combobox: Sets the ‘object’ object with SwfComboBox class and name as the value in the variable curObjClassName

xxi) radiobutton: Sets the ‘object’ object with SwfRadiobutton class and name as the value in the variable curObjClassName

xxii) static: Sets the ‘object’ object with Static class and name as the value in the variable curObjClassName

xxiii) statusbar: Sets the ‘object’ object with SwfStatusbar class and name as the value in the variable curObjClassName

xxiv) calendar: Sets the ‘object’ object with SwfCalendar class and name as the value in the variable curObjClassName

xxv) scrollbar: Sets the ‘object’ object with SwfScrollbar class and name as the value in the variable curObjClassName

xxvi) tab: Sets the ‘object’ object with SwfTab class and name as the value in the variable curObjClassName

xxvii) table: Sets the ‘object’ object with SwfTable class and name as the value in the variable curObjClassName

xxviii) tabstrip: Sets the ‘object’ object with WbfTabStrip class and name as the value in the variable curObjClassName

xxix) ultragrid: Sets the ‘object’ object with WbfUltraGrid class and name as the value in the variable curObjClassName

xxx) webgrid: Sets the ‘object’ object with WbfGrid class and name as the value in the variable curObjClassName

- If the value of arrObj(0) is anything else, the framework will generate an error report stating “Keyword <arrObj(0)> not supported. Please verify the keyword entered”.

6. Reporting and Error-handling Functions

6.1. Reporting Function:

Name of the function: Func_Report ()

Description: This function is used for generating a customized report with specified user inputs through the use of keywords.

Parameters: None

Assumptions: NA

Variables:

a) reportobj – Stores the contents of the third column of the current row in the Global Sheet

b) reportcon – Stores the status of the report (Pass/Fail)

c) reportcon1 – Stores the actual message of the report

d) reporter0 – Stores the expected message of the report

e) expmess – Stores the concatenated expected message

f) actmess – Stores the concatenated actual message

g) reporter1 – Stores the split value of reportcon1(1)

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 Func_Report () function will generate different reports. If the value is :

i) Pass:

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

ii) Fail:

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

iii) Done:

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

iv) 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 the error 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', it 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 Func_Error () 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 - Stores the elements from the third column of the data table after splitting with the ";" delimiter

b) strMainString - Stores the main string (arrSplit(0))

c) strSubString - Stores the sub string(arrSplit(1))

d) intLen - Stores the length of the array "arrSplit"

e) ReturnVal - Stores the return value

Functionality:

- Based on the values in strCriteria, Func_StringOperations () 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:

- objRegEx is set as a new regular expression object.
- 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 a strString argument. This will execute a regular expression test and return '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 – Holds the pattern string

b) strString – Holds the main string

c) aIndex() – Stores the match position of the pattern being searched in the main string

d) aValue() – Stores the value of the match

Return Value: 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 – Holds a regular expression object

b) Match – Holds the counter that can loop through the matches in 'Matches' variable

c) Matches – Holds the collection of matches found in the main string

Functionality:

- Sets regEx as a new regular expression object
- 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 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 main string.

Return Value: Returns replaced text

Assumptions: None

Variables:

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

Functionality:

- regEx is set as a new regular expression object.
- 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. Table Operation Functions

8.1. Function for Table Search:

Name of the function: Func_tablesearch ()

Description: This function is used to search for the particular row and column in the table based on the two search criteria entered in the keyword script for the perform keyword. It is also used to check for text present in a table based on the two search criteria entered in the keyword script for the check keyword.

Parameters:

a) object – Holds the type of the object and the object name on which action should be performed or checked

b) strSearch – Holds the search criteria entered in the keyword script (e.g. <colname1>;<colValue2>::<colname1>;<colValue2>--5)

c) strOutVar1 – Stores the number of the output row when the 'perform' keyword is used

d) strOutVar2 – Stores the number of the output column when the 'perform' keyword is used

Assumptions: NA

Variables:

a) intCheck – Stores the row number in the table where the text is present

b) icheck – Stores the flag to check if text is found or not

c) arrCol – Stores the number of columns

d) strCell1 – Stores the value present in the fourth column

e) strCell2 – Stores the child objects of the main window

f) intA – Counter used for the search operation

g) intB – Counter used for the search operation

h) strHeader – Stores the actual column names of the table in AUT

i) intFlag1 – Flag used for the search operation

j) intFlag2 – Flag used for the search operation

k) intRows – Stores the actual number of rows of the table in the AUT

l) intCols – Stores the actual number of columns of the table in the AUT

m) intStart – Stores the number of columns to be checked for

n) strCriteria – Stores the elements after splitting the value in the fourth column of the data table with delimiter '::'

o) strTemp1 – Stores the column name and the column value from the first element of array strCriteria

p) strTemp2 – Stores the Column name and the column value from the second element of array strCriteria

q) strRow1 – Stores the column value from strTemp1

r) strRow2 – Stores the column value from strTemp2

s) strCol1 – Stores the column name from strTemp1

t) strCol2 – Stores the column name from strTemp2

Functionality:

- Based on the value of arrObj(0), func_tablesearch returns different values – if arrObj(0) is “perform” then the row number and column number of the specified search string are returned. If the value is “storevalue” a True or False value based on the search performed on the table is returned. If arrObj(0) is “check” then a pass or fail report is generated in the test results.

9. Common Functions

9.1. 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 action should be performed or checked.

Assumptions: NA

Variables:

a) arrSplitCheckData - Stores the row number in the table where the text is present

b) strParamName - Stores 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 of the 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 GetValue () function performs different actions. If the value is:

  i) p: Parameter value of the variable in arrSplitCheckData(0) is returned by the function

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

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

9.2. 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 holds the value of the third column in the data table.

Assumptions: NA

Variables:

a) WshShell - This object is created for shell scripting.

Functionality:

- 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'.

9.3. Function for Dynamic Wait:

Name of the function: Func_Wait ()

Description: This function is used for synchronization with the application.

Parameters:

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

b) arrKeyValue – This operation needs to be performed on the object.

Assumptions: NA

Variables: NA

Functionality:

- Upper boundary of the array arrKeyValue is checked to be greater than or equal to zero or not.
- If yes then the Func_Wait () function will check for the value of the variable arrKeyValue(0).
- If the value is not equal to "exist" and "visible" then the Func_Wait () function will call the WaitProperty method of the object 'curParent' by passing the arguments "visible", True, and 10000.
- If the value is not equal to "enabled", "exist", and "visible" then the Func_Wait () function will check for the value in arrObj(0).
- If the value of arrObj(0) is "button", "checkbox", "textbox", or "radiobutton" then the Func_Wait() function will call the WaitProperty method of the object 'curParent' by passing the arguments "disabled", 0, and 10000
- If the value of arrObj(0) is "table", "combobox", "element", or "link" then the Func_Wait() function will call the WaitProperty method of the object 'curParent' by passing the arguments "visible", True, and 10000.

9.4. Function for Arithmetic Operations:

Name of the function: Func_arith ()

Description: This function is used to perform addition (+) and subtraction (-) 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 - Stores the environment value of the variables specified in the third column of the data table.

d) iFlag - Flag used for arithmetic operations

Functionality:

- This function will search for '#' in 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) '+': arrSplit1 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 arrSplit1(0) and arrSplit1(1). Then the Func_arith () 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.

ii) '*': arrSplit1 array is stored with the values in strX after splitting with the delimiter '*'. Then the Func_arith () function will retrieve the environment values if they start with '#'. Then the strX variable is replaced with values in the variables arrSplit1(0) and arrSplit1(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.

iii) '/': arrSplit1 array is stored with the values in strX after splitting with the delimiter '/'. Then the Func_arith () function will retrieve the environment values if they start with '#'. Then the strX variable is replaced with values in the variables arrSplit1(0) and arrSplit1(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) '-': arrSplit1 array is stored with the values in strX after splitting with the delimiter '-'. Then the Func_arith () function will retrieve the environment values if they start with '#'. Then the strX variable is replaced with values in the variables arrSplit1(0) and arrSplit1(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) '^': arrSplit1 array is stored with the values in strX after splitting with the delimiter '^'. Then the Func_arith () function will retrieve the environment values if they start with '#'. Then the strX variable is replaced with values in the variables arrSplit1(0) and arrSplit1(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.

9.5. Function for Querying Database:

Name of the function: Func_gfQuery ()

Description: This function is used to query the database.

Parameters:

a) strSQL – This variable 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 – Stores the database connection object

b) dbRS – Holds the result of the database operation performed

c) strReplace – Replaces the variables with their values in the SQL query stored in strSQL

d) connectionString – Creates and stores the DB connection string using the variables mentioned in e, f, g, h, and i

e) dbUID – Stores the user name to connect to the database

f) dbPWD – Stores the password to connect to the database

g) dbServer – Stores the database server name

h) dbHost – Stores the database host name

i) dbDRIVER – Stores 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 an 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 connection object and sets dbConn to 'Nothing'

9.6. 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 variable holds the data type into which the variable is to be converted.

Assumptions: NA

Variables:

a) strObject- Stores the variable to be converted

b) arrConvert - Stores elements such as the conversion type and variable to be stored in

c) strObject1 - Stores 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 variable strObject.
- The Func_Convert () function will store the values in variable strconverttype after splitting the delimiter ':' into the array arrConvert.
- Based on the values in arrConvert(0), the Func_Convert () 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 environment value of arrConvert(1).

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

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

v) 'cstr': It will convert the datatype of the value in strObject1 into 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).

9.7. 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 variable 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 - Stores the path with .xls

b) strDataPath - Locates and stores the full path

c) strSheetName - 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 Excel workbook in the path specified by the variable 'strDataPath' is imported to the Action1 sheet.

9.8. 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 variable holds the type of object being used for FSO, such as File, Folder, etc.

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

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

Assumptions: NA

Variables:

a) strFuncType – Stores the type of object(Ex: File, Folder) to be used

b) strFuncDetails – Stores 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 arguments.

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

9.9. 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 variable holds the details to be used while using FSO.

Assumptions: NA

Variables:

a) arrFolderpath – Stores the elements of the folder path separated by delimiter "\"

b) intFolderloc - Stores the element number of the Folder Name

c) DestFolder - Stores the Destination Folder

d) Foldername - Stores the Folder Name

e) oFSO - Stores the created File System Object

f) arrCellData - Stores the details of the operation to be performed

g) oFolder - Stores the details of 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 Func_Folder 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.
- If the folder is present then the folder is deleted.

iii) 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].

iv) 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].

9.10. 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 variable holds the details to be used while using FSO.

Assumptions: NA

Variables:

a) arrFilepath - Stores the File path

b) DestFile - Stores the Destination File Name

c) strFilename - Stores the File name to be used

d) intFileLoc - Stores the element number of the File Name

e) iFSO - Stores the created File System Object

f) oFile - Stores the details of Object created

g) arrCellData1 - Stores the details of the operation to be performed

h) intf - Performs looping

i) stress - Stores the String that 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 Func_File 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 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).

9.11. 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 variable holds the details to be used while exporting in XML format.

b) sPath – This variable holds the Path where the XML has to be stored.

Assumptions: NA

Variables:

a) into - Performs looping

b) oRoot - Stores the Root Element for the Object

c) arrDocSplit - Stores the elements to be exported to XML and the document name

d) strDocName - Stores the document name

e) arrElementSplit - Stores the variables and the tag names to be exported to XML

f) arrElementName - Stores the current variable and its tag name to be exported to XML

g) oDoc - Stores 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.
- 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

9.12. Function for Deleting XMLs

Name of the function: Func_DeleteXML

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

Parameters:

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

Assumptions: NA

Variables:

a) dFileObj - Stores the XML file to be used

b) dFSO - Stores the XML Object

Functionality:

- File System Object 'dFSO' object is created.
- XML file in the path specified in the variable 'sPath' is accessed using GetFile the method.
- Using the delete method the above file is deleted.

  dFSO object is set to Nothing.

10. Reusable Functions

10.1. Function for Calling Reusable Actions

Name of the function: Func_CallAction ()

Description: This function is used to call a Re-usable Action.

Parameters:

a) strData – Holds the name of the reusable action

b) strInfo – Holds the parameters for the reusable action

Assumptions: NA

Variables:

a) arrParam – Stores the parameters to be passed to the reusable action

b) strActionName – Stores the reusable action name

Functionality:

- This function checks for the value in the variable strInfo.
- If the value is Null then the Func_CallAction () function will call the RunAction method by passing variable 'actionName' and value 'oneIteration' as arguments (without passing parameters).
- If the value is not Null then the Func_CallAction () 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 array paramSplit, this 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.

11. Condition and Looping Functions

11.1. 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 – Holds the value of the current row number of the data table

Assumptions: NA

Variables:

a) iFlag – Sets the flag

b) cndSplit – Stores the value of the fourth column of the Global Sheet

c) startRow – Stores the start row for the condition

d) endRow – Stores the end row for the condition

e) cstrCellData – Stores the condition to be checked

f) var1 – Stores the first element to be evaluated

g) var2 – Stores 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.
- cstrCellData array is stored with the 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.

11.2. 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: NA

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 – Stores the start row and end row values

b) intcntr – Stores the loop count

c) Counter – Stores the count value

d) endRow1 – Stores the end row for looping

e) loopRowCount – Stores the current loop count

f) intDataCounter – 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 a delimiter.
- The variable intcntr is assigned with value in the fourth column in the data table.
- Value in the variable intcntr is converted into an integer and is stored in the variable Counter.
- The value in the variable intcntr is stored in 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.

12. Debug Functions

12.1. Function for Debugging:

Name of the function: DebugFunc ()

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

Parameters:

a) StartLineNumber – Holds the start line of the execution

b) EndLinenumber – Holds the end line of the execution

c) PrintOption – Holds the option to provide the Quick Test Print dialog with the environment variables

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

Assumptions: All 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.

Variables:

a) strText – Stores the contents of the log file

b) ostrFile – Stores the file object

c) oFSO – Creates a file system object

d) strDesktop – Stores the path of the desktop

e) oQtApp – Stores the Quick Test Object

f) oWshShell – Stores the windows Shell object

g) strContents – Stores the contents of the log file

h) arrSplit12 – Stores the array

i) VarName – Stores the interim array

j) strVarFullName – Stores the interim array

k) strVariableName – Stores the environment variable name

l) strVariableVal – Stores the interim array

m) strVariableValue – Stores the environment variable values

n) strFileName – Stores the file name

o) EnvSplit – Stores the interim array

p) VariableName – Stores the variable name

Functionality:

- Checks if the log file is required for debugging, and then loads all the environment variables to QTP.
- Creates a Shell object to access the desktop
- Stores the value of the desktop path
- Creates a file system object0
- Creates a quick test application object to access the test name

- 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 Quick Test Print Option (if the PrintOption parameter is set to true), the Print window of QTP is activated with all the values of the variables used in the script shown on the print window.
- If the log file is required for the debugging, then it will store all the environment variables to the log file.
- Opens the file for appending
- Writes the environment variables
- Clears all object variables