Open2Test PowerBuilder Test Automation Framework for QTP

Version 1.0

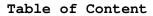**April 2011**

TABLE OF CONTENTS

1. Purpose of the Document

The purpose of this document is to describe the Open2Test Test Automation Framework code in detail.

1.1. Scope

The scope of this document is to provide details about Open2Test Test Automation Framework code and 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 – Checks whether a keyword script is present 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_Pb).

- 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".

3.2.   Main Function

Name of the function: Keyword_Pb ()

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) strIexplorePath – Store the path of Internet Explorer

b) initial – Stores the value in the second column

c) objName – Stores the value in third column

d) arrAction() – Stores the object type and name

e) objPerform – Stores the value present in the fourth column

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

g) arrKeyIndex() –Store all the values in fourth column separated by delimiter ":"

Functionality:

- Reads the values in the second, third, and fourth columns in the data sheet and stores the value in to the variables. (Please refer to details in the Variables section.)

- Based on the value in the variable initial (2ndcolumn), it calls different functions.

- If the value is other than 'perform', 'storevalue, ' or 'check', Keyword_Pb ()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 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 AUT.

Parameters:

a) Object – This refers to the object on which the specific operation needs to be performed.

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

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 the 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) rowcnt1 – Stores the value of the row count

b) colcnt1 – Stores the value of the column count

c) a – Stores the looping counter for row

d) b – Stores the looping counter for column

e) c – Stores the looping arrkeyindex(1)

f) valcompare – Stores the data of a cell

g) matchindex – Stores the value of arrKeyIndex(2) as integer

h) colname – Stores the name ofd the column

i) propSplit1 – This variable is used to store the split array of 4th Column of DataSheet when delimiter '_' is used

j) strstatus – This variable is used to store the split array of propSplit1(0) when delimiter '_' is used

Functionality:

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

  i)   click: This operation is performed on the current object.

  ii)   rightclick: Right clicks the object

  iii)   rightclick:<menu to be selected> : Right clicks and selects the menuitem

  iv)   doubleclick: Doubleclicks the object

v) setsecure:<encrypted text to be entered in the edit field> :

The SetSecure method is recorded when a password or other secure text is entered. The text is encrypted while recording, and decrypted during the test run. While the SetSecure method enables you to hide passwords on the screen when running a test, it is not intended to be a secure way to protect password information.

vi) set:d_currenttime : Sets current time in textbox

vii) set:d_currentdate : Sets current date in textbox

viii) set:d_day;<val of days to be added> : Adds the required number of days to the current date

ix) set:d_m;<val of months to be added> : Adds the required number of months to the current date

x) set:d_y;<val of months to be added> : Adds the required number of years to the current date

xi) set:<value> : Sets the value in the textbox

xii) type:<value> : Types the <value> in the text box

xiii) set:<value> : Sets the scroll bar position. <value> is the the scroll bar position to be set.

xiv) set:<value> : Sets the state of the checkbox.<value> can be ON, OFF or Dimmed.

xv) nextline: Moves the scroll bar downward, or to the right,one line

xvi) nextline:<number of lines> : Moves the scroll bar downward, or to the right, the specified number of lines

xvii) prevline: Moves the scroll bar upward, or to the left, one line

xviii) prevline:<number of lines> : Moves the scroll bar upward, or to the left, the specified number of lines

xix) nextpage: Moves the scroll bar downward, or to the right, one page

xx) nextpage:<number of pages>: Moves the scroll bar downward, or to the right, the specified number of pages.

xxi) prevpage: Moves the scroll bar upward, or to the left, one page

xxii) prevpage:<number of pages>: Moves the scroll bar upward, or to the left, the specified number of pages.

xxiii) type:<value>:Types the <value> in the text box

xxiv) seltexteditbox:<column at which selection starts>:<column at which selection ends> : Selects text in the winedit box.

xxv) mousemove:<X-Coordinate>:<Y-Coordinate>: Moves the mouse pointer to the designated position inside the object.

xxvi) activatecell:<row>:<column> : Activates (double-clicks) the specified cell in the table .<row>:The row containing the cell you want to activate.<column>:The column containing the cell you want to activate

xxvii) selectcell:<row>:<column> : Selects (clicks) the specified cell in the table. <row>:The row containing the cell you want

to activate.<column>:The column containing the cell you want to activate

xxviii) setcelldata:<row>:<column>:<value to be entered in the cell>: Sets the contents of a cell to the specified text. <row>:The row containing the cell.<column>:The column containing the cell.

xxix) selectcellindex:<rownumber>:<internalcolnumnumber>: This function is used to set the contents of a cell of datawindow identified by the <rownumber> and <internalcolumnnumber> to the specified text. <internacolumnnumber> takes into account the hidden columns

xxx) selectcell:<visiblerownumber>:<visiblecolnumnumber>: Selects (clicks) the specified cell in the table. <row>:The visible row containing the cell you want to activate.< visiblecolnumnumber >:The visible column number containing the cell you want to activate

xxxi) selectcellname:<rownumber>:<internalcolumnname>: Selects (clicks) the specified cell in the table. <row>:The row containing the cell you want to select.<internalcolumnname >:The internal column name containing the cell you want to select

xxxii) selectcellname:<rownumber>:<visiblecolumnname> : Selects (clicks) the specified cell in the table. <row>:The row containing the cell you want to select.< visiblecolumnname >:The visible column name containing the cell you want to select

xxxiii) setcelldataindex:<#varrownumber>:<#varcolumnnumber>:<value to be entered in the cell>: This function is used to set the contents of a cell of datawindow identified by the <#varrownumber> and <#varcolumnnumber> to the specified text. #varrownumber and #varcolumnnumber are variables that contain value for row number and column number

xxxiv) setcelldataindex:<rownumber>:<internalcolumnnumber>:<value to be entered in the cell> : This function is used to set the contents of a cell of datawindow identified by the <rownumber> and <internalcolumnnumber> to the specified text. <internacolumnnumber> takes into account the hidden columns

xxxv) setcelldataindex:<rownumber>:<visiblecolumnnumber>:<value to be entered in the cell> : This function is used to set the contents of a cell of datawindow identified by the <rownumber> and <visiblecolumnnumber> to the specified text.

xxxvi) setcelldataname:<rownumber>:<visiblecolumnname>:<value to be entered in the cell>: This function is used to set the contents of a cell of datawindow identified by the <rownumber> and <visiblecolumnname> or the column header to the specified text.

xxxvii) setcelldataname:<rownumber>:<internalcolumnname>:<value to be entered in the cell> : This function is used to set the contents of a cell of datawindow identified by the <rownumber> and <internalcolumnname>

xxxviii) selectrow:<rownumber> or <#varrownumber> : This function is used to click on the cell of datawindow specified by the <rownumber> or the variable <#varrownumber> thereby selecting the whole row.

xxxix) selectrow:<val1>:<val2>:<val3> : This function is used to click on the cell of datawindow specified by the <rownumber> or the variable <#varrownumber> thereby selecting the whole row

xl)   selectcellindex:<#varrownumber>:<#varcolnumber>   :   This function is used to click on the cell of datawindow specified by the <#varrownumber> and <#varcolumnnumber>. #varrownumber and # varcolumnnumber are variables that contain values for row number and column number.

xli)   selectcellindex: <rownumber>:<internalcolumnnumber> : This function is used to click on the cell of datawindow specified by the <row> and <internalcolumnnumber>.

xlii)   selectcellname:<rownumber>:<internalcolumnname> : This function is used to click on the cell of datawindow specified by the <rownumber> and <internalcolumnname>.

xliii) selectcellname:<rownumber>:<visiblecolumnname> : This function is used to click on the cell of datawindow specified by the <#row> and <#column>.

xliv)   selectmenu:<item name1;itemname2;> : Selects the menu based_on the name provided

xlv)   select:"<item to be selected>": Selects an item from listbox or listview

xlvi)   selectindex:<index of item to be selected> : Selects an item from listbox or listview based on index

xlvii) selectrange:<index of first item in range>:<index of last item in range> : Selects all list items between (and including) the two specified items.

xlviii) deselect:"<item to be deselected>" : Deselects an item from listbox or listview

xlix)   deselectindex:<index of item to be deselected>: Deselects an item from listbox or listview based on index

l)   extendselect:"<item to be selected>": Selects an additional item from the multi-selection list.

li)   setitemstate:"<The item to select from the list>":<value>:

Sets the state of the check box icon of the specified item in the list

<value> can hold the following values:

1: represents action "Check"

0: represents action "Uncheck"

-1: represents action "Click"

-2: represents action "Double Click"

lii)   setitemindexstate:<index of the item to select>:<value>:

Sets the state of the check box icon of the specified item in the list.

<value> can hold the following values:

1: represents action "Check"

0: represents action "Uncheck"

-1: represents action "Click"

-2: represents action "Double Click"

liii)   activate:"<name of the item to select from the list>" : Activates (double-clicks) an item in the list.

liv)   activateindex:<index of the item to select from the list>: Activates (double-clicks) an item in the list.

lv)   select:"<Root*Node1*Node2*."expandindex : Selects a node from the tree-view control. Note: * is the delimiter.

lvi)   selectindex:<index of item to be selected>collapseindex : Selects a node from the tree-view control.

lvii)   expand:"<Root*node1*node2>"expandallindex : opens the dropdown menu associated with the specified toolbar button by clicking the arrow part of the button.

lviii)   expandindex:<index of the node>: Displays hidden sub-nodes of the specified node in a tree-view control.

lix)   collapse:"<Root*node1*node2…>" : Hides the sub-nodes of the specified node in an expanded tree-view control.

lx)   collapseindex:<index of the node>maximize : Hides the sub-nodes of the specified node in an expanded tree-view control.

lxi)   expandall:"<Root*node1*node2…>"resize : Expands the node in the tree-view control and all of the nodes below it.

lxii)   expandallindex:<index of the node>pressindex : Expands the node in the tree-view control and all of the nodes below it.

lxiii)   setitemstate:"<value1>":<value2>showdropdownindex :

Sets the state of a check box icon of the specified node in a tree-view control.

<value1>: The full path of the node whose check box you want to set.

E.g: "Root*Child1*Child2"

Note: * is the delimiter.

<value2> can hold the following values:

1: represents action "Check"

0: represents action "Uncheck"

–1: represents action "Click"

–2: represents action "Double Click"

lxiv)   setitemindexstate:<index of the item to select>:<value> :

Sets the state of a check box icon of the specified node in a tree-view control.

<value> can hold the following values:

1: represents action "Check"

0: represents action "Uncheck"

–1: represents action "Click"

–2: represents action "Double Click"

lxv)   activate:"<Root*Child1*Child2….>" : Activates (double-clicks) a node in the tree-view control.

lxvi)   close: Closes the window

lxvii)   activate: Activates the window.

lxviii)   maximize: Maximizes the window to fill the entire screen.

lxix)   resize:<new width of window in pixels>:<new height of window in pixels> : Resizes the window to the specified dimensions.

lxx)   Restore : Restores the window to its previous size.

lxxi)   press:"<name of the toolbar button to press>" : press:"<name of the toolbar button to press>"

lxxii)   pressindex:<index of the toolbar button to press> : Clicks the specified toolbar button.

lxxiii)   showdropdown:"<toolbar button name for which you want to show the drop-down item>" : Opens the dropdown menu associated with the specified toolbar button by clicking the arrow part of the button.

lxxiv)

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.

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

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) rowcnt1– Stores the row number

d) colcnt1 – Stores the column number

e) varName – Stores variable Name to be used by Environmental variable

f) rowcnt1 – Stores row count

g) colcnt1 – Stores column count

h) selitem – Stores the selected object

i) selcount – Stores the item count of selected object

j) colname – Stores the column name

k) resultfn – Stores the content of the selected object

l) matchindex – Stores the value of arrKeyIndex(3) as integer

m) visiblecheck – Stores the value of a counter

Functionality:

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

   i)   colname: col name of object value will be stored in an Environment variable

  ii)   columncount : ColumnCount of object value will be stored in an Environment variable

 iii)   rowcount : RowCount of object value will be stored in an Environment variable

  iv)   text : RO property of the object is captured and stored in an Environment variable

   v)   defaultvalue: RO property of the object is captured and stored in an Environment variable

  vi)   maxlength: RO property of the object is captured and stored in an Environment variable

 vii)

viii)   selecteditems: selected items are stored in an Environment variable

  ix)   enabled: RO property of the object is stored in an Environment variable

   x)   focused : RO property of the object is stored in an Environment variable

  xi)   selectioncount: RO property of the object is stored in an Environment variable

 xii)   checked : RO property of the object is Boolean and is stored in an Environment variable

xiii)   item: Getitem(arrKeyIndex(2) is stored in an Environment variable

 xiv)   itemscount : GetitemsCount is stored in an Environment variable

  xv)   celldataindex : Environment variable is set to True or False depending on the value of arrKeyIndex(3)

 xvi)   celldataname: Environment variable is set to True or False depending on the value of arrKeyIndex(3)

xvii)   itemexist : Environment variable is set to True or False depending on the value of arrKeyIndex(2)

xviii)   itemnotexist : Environment variable is set to True or False depending on the value of arrKeyIndex(2)

 xix)   getrowno : row number is stored in Environment variable

  xx)   getcolno : col number is stored in Environment variable

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

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

d) arrKeyIndex – This additional parameter if 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:

propertyVal – Stores the property name that needs to be checked

strChecking – Stores the property value retrieved from the AUT

rowcnt1 – Stores the row count

colcnt1 – Stores the col count

matchindex – Store the value of arrKeyIndex(2) converted to Integer

resultfn – Stores the content of the object

colname – Stores the column name

actval – Stores the item in arrkeyindex(1)

count – Stores the count of items in the object

selItem – Stores the selected item

Functionality:

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

  i) enabled: The propertyVal variable is assigned with the value enabled.

  ii) focused: The propertyVal variable is assigned with the value focused.

  iii) item: If actval = arrKeyIndex(2) ,then it generates a pass report that item exists

  iv) checked: The propertyVal variable is assigned with the value checked.

  v) itemscount: If the value in arrObj(0) is combobox or radiobutton, then the propertyVal variable is assigned with the value items count.

  vi) exist: If the arrKeyIndex(1) value is true, then the function checks whether the current parent exists or not. If it exists,the Func_Check() function will generate a pass report. Otherwise it will generate a fail report. If the

arrKeyIndex(1) value is false, the function checks whether the current parent exists or not. If it exists, it will generate a fail report. Otherwise it will generate a pass report.

vii) Selecteditems : If Instr(1, selitem, arrKeyIndex(1), 1) >0 ,then it generates a report that arrObj(0) & arrObj(1) are selected

viii) Itemexist : If Instr(1, resultfn, arrKeyIndex(1), 1) >0 then it generates a report that item exists in arrObj(0) & arrObj(1)

ix) Itemnotexist : If Instr(1, resultfn, arrKeyIndex(1), 1) =0 then it generates a report that item exists in arrObj(0) & arrObj(1)

x) text: The propertyVal variable is assigned with the value text.

xi) Rowcount : If arrKeyIndex(1) = object.RowCount,then it generates a report that row count in arrObj(0) & arrKeyIndex(1) are equal

xii) Columncount : If arrKeyIndex(1) = object.ColumnCount, then it generates a report that col count in arrObj(0) & arrKeyIndex(1) are equal

xiii) Celldataindex : If arrKeyIndex(3) = object.GetCellData(Cint(arrKeyIndex(1)),Cint(arrKeyIndex(2))) Then it generates a report that content of cell in row arrKeyIndex(1) and column arrKeyIndex(2) is arrKeyIndex(3),

- 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: 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)   browser: Sets the curParent object with "Browser" Class with the name in the variable arrObj(1)

  ii)   window: Sets the curParent object with "Window" Class with the name in the variable arrObj(1)

  iii) window: Sets the curParent object with "Window" Class with the name in the variable arrObj(1)

  iv)   dialog: Sets the curParent object with "Dialog" Class with the name in the variable arrObj(1)

  v)   datawindow: Sets the curParent object with "DataWindow" Class with the name in the 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: AUT is already up and running.

Variables:

a) ObjectVal – Stores the table object class name

Functionality:

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

- If the value is:

  vi)   button: Sets the button with the value in the variable arrObj(1)

  vii)   winbutton: Sets the winbutton with the value in the variable arrObj(1)

 viii)   winedit: Sets the winedit with the value in the variable arrObj(1)

  ix)   winmenu: Sets the winmenu with the value in the variable arrObj(1)

   x)   winscrollbar: : Sets the winscrollbar with the value in the variable arrObj(1)

  xi)   checkbox: Sets the checkbox with the value in the variable arrObj(1)

  xii)   textbox: Sets the textbox with the value in the variable arrObj(1)

 xiii)   combobox: Sets the combobox with the value in the variable arrObj(1)

  xiv)   list: Sets the list with the value in the variable arrObj(1)

  xv)   listview: Sets the listview with the value in the variable arrObj(1)

  xvi)   radiobutton: Sets the radiobutton with the value in the variable arrObj(1)

 xvii)   scrollbar: Sets the scrollbar with the value in the variable arrObj(1)

xviii)   tabstrip: Sets the tabstrip with the value in the variable arrObj(1)

  xix)   toolbar: Sets the toolbar with the value in the variable arrObj(1)

  xx)   treeview: Sets the treeview with the value in the variable arrObj(1)

  xxi)   object: Sets the object with the value in the variable arrObj(1)

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

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. When ever 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_Context (arrObj,intRowCount)

Description: This function is used to add new objects.

Parameters:

a) arrObj – An array of object names

b) intRowCount – Stores the current row number

Assumptions: None

Variables:

a) strReportData – Stores the contents of the fourth Column of current row in 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

7.2. Function for setting Parent and Child Objects:

Name of the function: Func_ObjectSet (arrObj,intRowCount)

Description: This function sets the parent and child objects.

Parameters:

c) arrObj – An array of object names

d) intRowCount – Stores the current row number

Assumptions: None

Variables:

a) ObjectVal– Stores the Table object Class name.

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: 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:

- 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 & 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:

- 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 & strReplace arguments. This will execute a regular expression Find & 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 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:

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

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

  vii)   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

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 object was created for shell scripting.

Assumptions: NA

Variables: NA

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'.

8.3.   Function for Dynamic Wait:

Name of the function: Func_Wait ()

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

Parameters:

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

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

Assumptions: NA

Variables: NA

Functionality:

- A function check for the upper boundary of the array arrKeyValue is 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 "exist" then the Func_Wait () function will call the WaitProperty method of the object 'currentparent' by passing the arguments "visible", True and 10000.

8.4.   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 as 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.

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 '*' or '/' 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 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) '*': strSplit 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 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.

  iii) '/': strSplit 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 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 Func_arith () 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.

8.5.   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) connectionString – Stores the connection string for the database

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

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

f) dbServer – Stores the database server name

g) dbHost – Stores the database host name

h) dbDRIVER – Stores the database driver name

Functionality:

- Creates DB object dbConn

- Calls the method open for the bdConn 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 object and sets dbConn to 'Nothing'

8.6.   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:

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

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

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

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

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

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

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

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

8.7.   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 work book in the path specified by the variable 'strDataPath' is imported to the Action1 sheet.

8.8.   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:

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

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

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

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

8.9. 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) intFolderlo – 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 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:

 xx) 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.

xxi) delete:

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

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

xxiii) 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].

xxiv) 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].

8.10. 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 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 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 Func_File function will perform different actions. If the value is:

xxv) 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.

xxvi) 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.

xxvii) 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].

xxviii) 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].

xxix) 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.

xxx) 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.

xxxi) 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).

8.11. 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

8.12. 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.

9. Reusable Functions

9.1. 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.

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 keyword script.

Parameters:

a) intRowCount – Holds the value of 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.

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 – Holds the query variable that has to be converted

b) strconverttype – 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 – 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 into the variable Counter.

- This function recursively calls 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 – 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) arrSplit2 – 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 load all the environment variables to the 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