Open2Test Flex Test Automation Framework for QTP

Version 1.0

March 2011

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

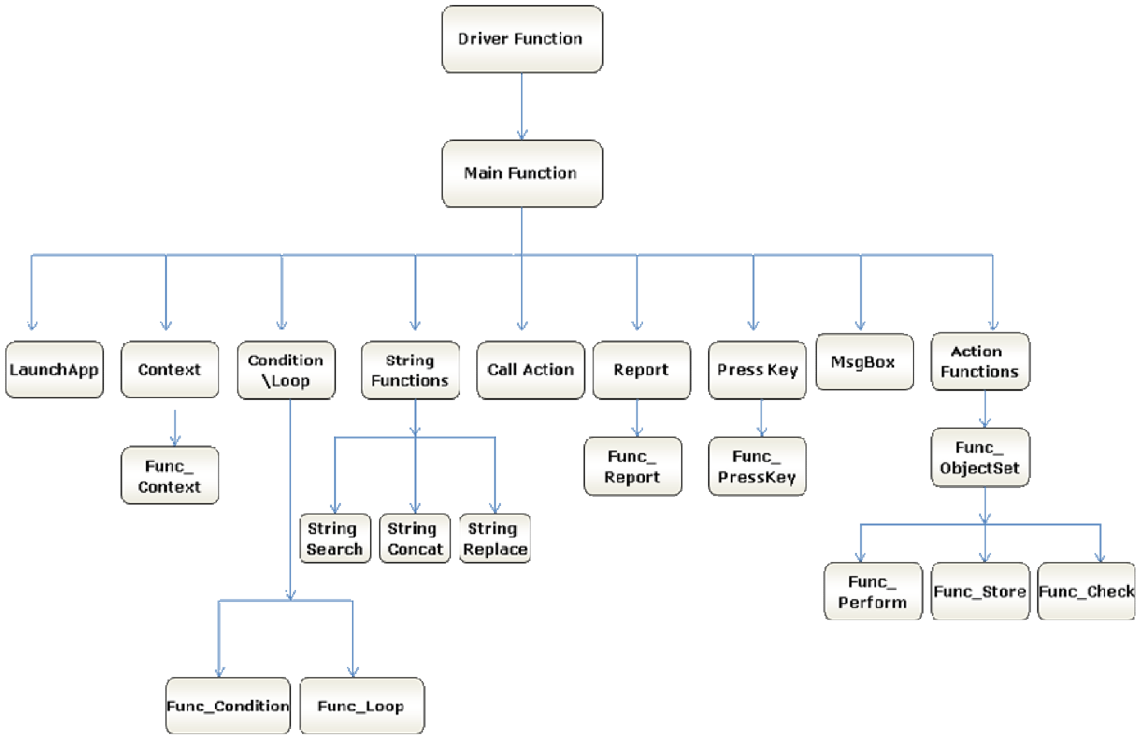
The scope of this document is to provide details about Open Source Test Automation Framework code and its architecture and functions.

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_Flex).
- 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_Flex ()

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) action() – Stores the object type and name

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

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

g) keyIndex() -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 (2nd column), it calls different functions.
- If the value is other than 'perform', 'storevalue, ' or 'check', Keyword_Flex()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) action() – This holds the type of the object on which the action has to be performed and the object name.

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

d) keyIndex - 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 grid 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) strParam - Stores the value of the fifth column of the data table

b) arrParam1 - Stores the values after splitting strParam with

the delimiter ";"

c) Row - Stores the number of rows in the grid

d) col - Stores the number of columns in the grid

e) checked - Stores the ROProperty "value" of the specified object

f) propSplit1 - Stores the split array of the fourth column of the data sheet when delimiter '_' is used

g) strstatus - Stores the split array of propSplit1(1) when delimiter '_' is used

h) flag - Stores the flag value

i) actualItem - Stores the value to be checked

j) loopCurrentItem - Performs looping

k) varName - Stores the value in strstatus(0)

l) strvar - Stores the string value in the fourth column of the data sheet

Functionality:

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

i) rownum: Stores the value in the fifth column in the variable strParam and calls the function Func_GetRowNum().

ii) set: Checks the value in action(0). If the value is 'textarea' or 'textinput' then the value in keyIndex(1) is set.

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

iv) selectindex: Checks for the value in action(0). If the value is 'combobox', 'listbox', 'advanceddatagrid' or 'dataggrid' then the item with the index number in the keyIndex(1) variable is selected in the object.

v) select: Checks for the value in action(0). If the value is 'radiobutton', 'combobox', 'datagrid', 'checkbox', 'advanceddatagrid', 'menu' or 'tree' then the item with the value in the keyIndex(1) variable is selected in the object.

vi) verifyselect: **Assigns the actualListCount variable with "items count" from the GetROProperty of the object.** It will loop for each value from 1 to actualListCount until the value of the variable actualItem is equal to keyIndex(1).If the actualItem value is equal, then it will select the item (keyIndex(1)) in the current object.

vii) close: Checks for the value in the variable action(0). If the value is "browser", "window" or "dialog", then it will perform a close operation on the parent object.

viii) navigate: Checks for the value in the variable action(0). If the value is "browser", then it will navigate the browser to the URL in keyIndex(1).

ix) check: Checks for the value in action(0). If the value is 'checkbox', then the checkbox is selected.

x) uncheck: Checks for the value in action(0). If the value is 'checkbox', then the checkbox is unselected.

xi) headerclick: Checks for the value in action(0). If the value is 'advanceddatagrid' or 'datagrid', then the header of the grid is clicked based on the keyIndex(1) and keyIndex(2).

xii) selecttext: Checks the value in action(0). If the value is 'textarea' or 'textinput' then a part of the text is selected based on keyIndex(1) and keyIndex(2).

xiii) setfocus: Sets the focus on any object.

xiv) maximize/minimize: Checks for the value in the variable action(0). If the value is "browser" or "window", then it will maximize or minimize the parent object.

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

  If the value of action(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 action(1)

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

iii) sqlcheckpoint:

Sets the current row in action sheet to 1

Sets the TO property of the connection string

Changes the DB objects source(SQL) statement

Executes the DB checkpoint

iv) random:

Stores the action(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 argument

Stores the return value from the function in the environment variable in fourth column of Excel sheet

v) split:

Stores the values of the variable action(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) propName – Stores the name of the property

b) propSplit – Holds the property and variable names

c) varName – Stores propSplit(1)

Functionality:

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

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

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

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

v) errortext: "error string" RO property of the object is stored in the variable in the fourth column in the data table (propSplit(1)).

vi) focussed: "focusenabled" RO property of the object is stored in the variable in the fourth column in the data table (propSplit(1)).

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

viii) maxlength: "max length" RO property of the object is stored in the variable in the fourth column in the data table (propSplit(1)).

ix) selecteditems: "selecteditem" RO property of the object is stored in the variable in the fourth column in the data table (propSplit(1)).

x) selectedindex: "selectedindex" RO property of the object is stored in the variable in the fourth column in the data table (propSplit(1)).

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

xii) visible: "visible" property of the object is stored in the variable in the fourth column in the data table (propSplit(1)).

xiii) getcelldata: "GetCellData" method of the object is invoked by passing intRowNum (RowNo) and intColNum (ColumnNo) variables as arguments. The return value is stored in the variable in the fourth column in the data table (propSplit(1)).

- If the value in arrPropSplit(0) does not have any of the values listed above then propSplit(0) ROProperty of the object is stored in the value in the variable propSplit(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) keyvalue – This is the operation that needs to be performed on the object.

d) keyIndex – 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

checking – Stores the property value retrieved from the AUT

Functionality:

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

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

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

iii) visible: The propertyVal variable is assigned with the value "visible".

vi) focused: The propertyVal variable is assigned with the value "focusenabled".

vii) itemscount: If the value in action(0) is combobox, listbox, datagrid or advanceddatagrid, the propertyVal variable is assigned with the value "items count" or "getitemscount".

viii) exist: If the keyIndex(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 keyIndex(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.

ix) selected: If the value in action(0) is checkbox or radiobutton then the propertyVal variable is assigned with the value "selected". Also if the value of the variable keyIndex(1) is 'true' then 1 will be assigned to it. Otherwise, 0 will be assigned to the keyIndex(1)

- 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 keyIndex(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) childCell - Stores the elements separated by the delimiter '::'

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

d) Child - Stores the child objects of the main window

Functionality:

- Based on the values on action(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 action(1)

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

iii) dialog: Sets the curParent object with "Dialog" Class with the name in the variable action(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:
NA

Functionality:

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

i) Application: Sets the application with the value in the variable action(1)

ii) Canvas: Sets the canvas with the value in the variable action(1)

iii) textbox: Checks for the value in the variable parChild. If the Value in the parChild is "canvas", then the current object is set with textarea class with the name as the value in the variable action(1).

iv) button: Checks for the value in the variable parChild. If the Value in the parChild is "application" or "canvas", then the current object is set with FlexButton class with the name in the variable action(1).

v) combobox: The current object set with FlexCombobox class with the name in the variable action(1)

vi) checkbox: The current object set with FlexCheckbox class with the name in the variable action(1)

vii) radiobutton: The current object set with FlexRadiobutton class with the name in the variable action(1)

viii) image: The current object set with Image class with the name in the variable action(1)

ix) Advgrid: The current object set with Advanceddatagrid class with the name in the variable action(1)

x) element: The current object set with FlexElement class with the name in the variable action(1)

xi) link: The current object set with Link class with the name in the variable action(1)

xii) tree: The current object set with FlexTree class with the name in the variable action(1)

xiii) menu: The current object set with Flexmenu class with the name in the variable action(1)

xiv) menubar: The current object set with FlexTogglebuttonbar class with the name in the variable action(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_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:

A new regular expression object is created.

- The Pattern property of the above object is set with the value in the strPattern variable.
- The IgnoreCase property of the above object is set to ‘False’.
- The test method of the object is invoked by passing 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 – 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. Grid Operations

8.1 Function for Retrieving Row Number:

Name of the function: Func_GetRowNum ()

Description: This function is used to retrieve the row number in which specified text is present in table. This function is used for 'getrownum' 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

c) strReturnVal - Stores the output row number for perform keyword

Assumptions: NA

Variables:

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

b) intStart - Stores the flag to check text is found or not

c) arrCol - Stores the number of columns

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

e) strReturnVal1- Stores the child objects of the main window

Functionality:

- Based on the values in keyvalue(0),the Func_GetRowNum () function performs different actions

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 perform keyboard operations.

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 action(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 hold the value of the third column in the data table.

b) keyvalue - 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 keyvalue is greater than or equal to zero or not.
- If yes then the Func_Wait () function will check for the value of the variable keyvalue(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.

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.

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.

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) 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'

9.6. Function for Converting Data types:

Name of the function: Func_Convert ()

Description: This function is used to convert data types.

Parameters:

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

b) action(0) – This variable holds the data type into which the variable is to be converted.

Assumptions: NA

Variables:

a) action(1)– Stores the variable to be converted

b) action(0) – Stores elements such as the conversion

c) keyIndex(0) – 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 action(0), the Func_Convert () function will perform different actions. If the value is:

  i) 'date': It will convert the value in the action(1) variable based on the format available in action(0).

  ii) 'roundto': It will round the value in the variable action(1)and store it in environment value of keyIndex(0).

  iii) 'lcase': It will convert the value in action(1) into lowercase and store it in environment value of keyIndex(0).

  iv) 'ucase': It will convert the value in action(1) into uppercase and store it in environment value of keyIndex(0).

v) ‘cstr’: It will convert the datatype of the value in action(1)into string datatype and store the converted value in keyIndex(0).

vi) ‘ascii’: It will convert the value in action(1)into ASCII value and store the converted value in keyIndex(0).

vii) ‘trim': It will remove the extra spaces in the value in the variable action(1).

viii) ‘len’: It will return the length of the value in the variable action(1) and store the returned length in the variable keyIndex(0).

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 work book 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) 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:

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 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) strMess - 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.

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) 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.

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) 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) 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].

viii) 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 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.

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:

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_Flex 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) 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