**Test Automation Function/Method Parameter Details**

**Object Type Key Words**

The following key words are the valid web element object types supported by the test automation framework. The object types are not explicitly used as parameters by the two most useful and important generic methods **ObjectActionSet** and **ObjectPropertyValueGet** in the test automation framework. But the object types are implicitly used and mapped to the given GUI object name for determining the default action to perform on the given object when indicated by the **DEFAULT** key word supplied in these two methods as parameters.

1. **TEXTBOX :** Indicates that the web element is a text field where text input is typed in.With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **TEXTINPUT**. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **CURRTEXT**.
2. **LISTBOX :** Indicates that the web element contains just a dropdown list or both a text field and a dropdown list. The given text input can either be typed in the text field or the given item string can be selected from the dropdown list. In this case the item to be selected is case sensitive. With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **ITEMSELECT**. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **CURRTEXT**.
3. **COMBOBOX :** Indicates that the web element contains a text field (optionally), a dropdown list and check boxes with each dropdown item. The given text input can either be typed in the text field or the given item string can be selected and/or check marked from the dropdown list. In this case the item to be selected is case sensitive. With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **ITEMSELECT**. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **CURRTEXT**.
4. **DROPDOWN :** Indicates that the web element is a dropdown object containing a predefined list of items for selection, usually initiated by clicking on the **LISTBOX** or **COMBOBOX** object types. With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **ITEMSELECT**. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **CURRITEM**.
5. **PUSHBUTTON :** Indicates that the web element is a push button with a label that can be pressed by the mouse pointer. With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **LCLICK**. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **OBJLABEL**.
6. **ICONBUTTON :** Indicates that the web element is a push button with no label that can be pressed by the mouse pointer. These buttons are usually marked with icons like magnifying glass, plus sign, cross sign, pen symbol, trash can symbol and etc. With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **LCLICK**. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **OBJICON**.
7. **RADIOBUTTON :** Indicates that the web element is a kind of an option button, usually round in shape, which are mutually exclusive while being set by left mouse button click. In the group, while one radio button is set, the other previously set radio button is automatically unset. Only one of these radio buttons can be set at any given time. With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **SETON**. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **CURRSTATE**.
8. **CHECKBOX :** Indicates that the web element is a kind of a toggle box, usually square in shape, that can be checked or unchecked by two consecutive mouse pointer clicks. With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **SETON**. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **CURRSTATE**.
9. **CALENDARBOX :** Indicates that the web element is a box type object which can be clicked to invoke a separate calendar control for date picking purpose. In this case, the date must be supplied in the supported format. With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **DATEPICK**. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **CURRTEXT**.
10. **CALENDARCTRL** : Indicates that the web element is a date picker calendar control object where any given date (day,month,year) can be picked. In this case, the date must be supplied in the supported format. With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **DATEPICK**. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **CURRTEXT**.
11. **TITLETOOLBAR :** Indicates that the web element is of type title bar containing the title text of the page or the pertinent parent object. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **TITLETEXT**.
12. **DIALOGPAGE :** Indicates that the web element is of type dialog page that gets initiated from the active main page as a popup dialog. With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **XCLOSE**. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **TITLETEXT**.
13. **TOGGLEBUTTON :** Indicates that the web element is a toggle object in the form of a button or an icon which can be toggled via two consecutive mouse clicks for either activating/deactivating or enabling/disabling or revealing/concealing another dependent object, i.e. Document type tabs under Documents information. The button can be toggled by the action type **LCLICK**. The desired toggle state itself can be optionally indicated by the key words **ON** or **OFF** passed in the **ValueToBeSet** parameter of the pertinent method call. With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **SETON**. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **OBJLABEL**.
14. **SELECTBUTTON :** Indicates that the web element is a select button which can be selected via a single mouse click. When selected, usually the color code of the button changes to indicate selected state. This type of objects can also appear in a cluster as mutually exclusive buttons. In this case pressing one of the un-highlighted buttons makes it highlighted and active while the other active one becomes un-highlighted and inactive simultaneously. The button can be selected by the action type **LCLICK** or **DEFAULT**. The framework confirms the selection by checking the selected state automatically. The **ValueToBeSet** parameter is irrelevant for this object type and should be set to **NULL**. With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **LCLICK**. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **OBJLABEL**.
15. **EXPANDBUTTON :** Indicates that the web element is a toggle object in the form of a button which can be toggled via two consecutive mouse clicks for either expanding or collapsing a dropdown web data elements segment. This type of button usually contains a down or left arrow depicting expanded or collapsed state respectively. The state can be toggled by the action type **LCLICK**. The desired state itself can be optionally indicated by the key words **ON** (to **EXPAND**) or **OFF** (to **COLLAPSE**) passed in the **ValueToBeSet** parameter of the pertinent method call. With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **EXPAND**. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **OBJLABEL**.
16. **WEBLINK :** Indicates that the web element is a link containing a web address which can be clicked to navigate to that web page. With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **LCLICK**. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **URLSTRING**.
17. **WEBTABLE :** Indicates that the web element is a table object containing data grid in the row and column format. The actual data is presented in each cell of each row. The row indicated by zero(0) index is the very first row containing the column headers. With the **WebTableCellActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **COLUMNSORT** on the given column header.
18. **DATACELL :** Indicates that the web element is a particular cell in the data grid of the given web table containing any kind of data. With the **WebTableCellActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **LCLICK**. With the **WebTableCellValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **CURRTEXT**.
19. **HEADERTAB :** Indicates that the web element is a tab like object in the top section within the web page which can be clicked to navigate to the web page represented by the tab. With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **LCLICK**. The framework confirms the selection by checking the selected state automatically. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **CURRSTATE**.
20. **MENUBUTTON :** Indicates that the web element is a menu type object or a button that can be clicked to initiate a dropdown menu listing the menu items. With action type **LCLICK**, the desired state itself can be optionally indicated by the key words **ON** (to **EXPAND**) or **OFF** (to **COLLAPSE**) passed in the **ValueToBeSet** parameter of the pertinent method call. With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **EXPAND**. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **CURRSTATE**.
21. **BUTTONBOX :** Indicates that the web element is a place holder box containing a specific and meaningful logo which can be clicked to navigate to the main page represented by the logo. The module buttons in the home page of Ababil are the examples of such buttons. With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **LCLICK**. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **OBJLABEL**.
22. **TOOLTIPBOX** : Indicates that the web element is a tooltip type object containing a text string which can be obtained by the **ObjectPropertyValueGet** method call. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **CURRTEXT**.
23. **POPUPBOX** : Indicates that the web element is a pop-up message box type object containing a text message string which can be obtained by the **ObjectPropertyValueGet** method call. With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **XCLOSE**. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **CURRTEXT**.
24. **ACTIVEPAGE** : Indicates that the named web element itself is the currently active web page and the values of the various properties of the web page itself can be obtained by the **ObjectPropertyValueGet** method call. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **TITLETEXT**.
25. **ONOFFSWITCH**  : Indicates that the web element is an input switch type object which can be shifted to the left or right or up or down in order to toggle between on and off position. The object can be toggled by the action type **LCLICK**. The desired toggle state itself can be optionally indicated by the key words **ON** or **OFF** passed in the **ValueToBeSet** parameter of the pertinent method call. With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **SETON**. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **CURRSTATE**.
26. **GROUPTITLEBAR** : Indicates that the web element is of type title bar containing the title text of the container box (mat-card) holding a set of various web elements that are grouped together on purpose. Since there exist too many group title bar objects of this type in the application, the **VISIBLE** or **ABSENT** state of such object is checked dynamically by the pertaining methods using the supplied title in the ExtraParam parameter. The **ObjectVisibilityAssert**, **ObjectInvisibilityAssert**, **ObjectCurrentStateAssert** and **ObjectCurrentStateCheck** methods can be used for this purpose.
27. **FILEUPLOADBOX** : Indicates that the web element is a file container box type object containing attached and uploaded file. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **TITLETEXT**, the file header text of the box.
28. **FILEIMAGE** : Indicates that the web element is an image type object representing an uploaded file. With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **LCLICK**. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **CURRSTATE**.
29. **DROPDOWNITEM** : Indicates that the web element is an individual item type object in the dropdown list, i.e. a menu item. With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **LCLICK**. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **CURRTEXT**.
30. **BROWSERTAB** : Indicates that the web element of this type of object is a tab of the active browser window, i.e. ababil-home tab. With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **LCLICK**. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **CURRTEXT**.
31. **DATALISTITEM** : Indicates that the web element of this type of object is a list of various data of different types grouped together in a data list box, i.e. Joint Type account Individual Lookup data list boxes, Organization Type account Owners data list boxes with Edit, Delete icon buttons and so on . With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **LCLICK**. With the **ObjectPropertyValueGet** method call**,** the property types supported are **NAME**, **FATHERNAME**, **MOTHERNAME**, **MOBILE**, **EMAIL**, **TIN**, **NID, DESIGNATION** (Owner Type)**, CATEGORY** (Individual, Organization etc.)**, SHAREAMOUNT** (Percentage/ration of the share), **INSTANCENO** (serial number of the occurrence starting from 1) and **OBJECTTYPE** (PUSHBUTTON, ICONBUTTON, CHECKBOX etc.). One property type can be used as a reference in the extra property identifier column of the datasheet (the **ExtraParam** parameter of the method) to get the value of the other. For example, to get the **FATHERNAME** or the **SHAREAMOUNT** propertyvalue**,** the **NAME**:”Jane Doe” or to indicate the 2nd occurrence of the given name, the **NAME**:”John Doe”|**INASTANCENO**:”2” indicator in the same syntax using the same delimiters can be used in the **ExtraParam** parameter. An example call of the set and get methods would look like the following:

**ObjectActionSet** (**DATALISTITEM**, **LCLICK**, **NULL**, **NAME:**”MS ANNIE”) – Clicking on the given name text

**ObjectActionSet** (**DATALISTITEM**, **DEFAULT**, **EDIT**, **NAME:**”MS ANNIE”) – Clicking on the Edit button for the given Name property

**ObjectPropertyValueGet** (**DATALISTITEM**, **SHAREAMOUNT**, OutPropertyValue, **NAME**:”John Doe”|**INASTANCENO**:”2”)

**ObjectPropertyValueGet (DATALISTITEM**, **NAME**, OutPropertyValue, **MOBILE**:”01702345678”)

1. **SLIDERBAR** : Indicates that the web element of this type of object is a slider bar with a predefined range allowing to set the share percentage for joint/organization account owners for example. With the **ObjectActionSet** method call**,** the default action type indicated by the **DEFAULT** key word for this type of object is **VALUESET**. With the **ObjectPropertyValueGet** method call**,** the default property type indicated by the **DEFAULT** key word for this type of object is **CURRTEXT**.

**Action Type Key Words**

The following key words are the valid action type parameters for the **ObjectActionSet** method in the test automation framework. These key words represent the valid actions that can be performed on the different web objects on the currently active web page.

1. **DEFAULT** : Indicates default action on the object for the given pertinent object type as defined in the **ObjectActionSet** and **ObjectPropertyValueGet** methods.
2. **XCLOSE** : Indicates the close action of the pop up message box type objects. This is also the **DEFAULT** action for the pertinent object types.
3. **LCLICK** : Indicates the left mouse button click action on the object of the given pertinent object type. The mouse click point coordinates “x,y” may be optionally given in the **ExtraParam** parameter, relative to the object itself where the coordinates of the upper left corner point of the object being 0,0.
4. **RCLICK** : Indicates the right mouse button click action on the object of the given pertinent object type. The mouse click point coordinates “x,y” may be optionally given in the **ExtraParam** parameter, relative to the object itself where the coordinates of the upper left corner point of the object being 0,0.
5. **MCLICK** : Indicates the middle mouse button click action on the object of the given pertinent object type. The mouse click point coordinates “x,y” may be optionally given in the **ExtraParam** parameter, relative to the object itself where the coordinates of the upper left corner point of the object being 0,0.
6. **DLCLICK** : Indicates the double click action of the left mouse button on the object of the given pertinent object type. The mouse click point coordinates “x,y” may be optionally given in the **ExtraParam** parameter, relative to the object itself where the coordinates of the upper left corner point of the object being 0,0.
7. **MROLL** : Indicates the middle mouse button roll action on the object of the given pertinent object type. The number of mouse pointer rotation cycles may be optionally given in the **ValueToBeSet** parameter.
8. **MOUSEOVER** : Indicates the mouse pointer move over action on the object of the given pertinent object type without any mouse button click. The mouse pointer move-over coordinates “x,y” may be optionally given in the **ExtraParam** parameter, relative to the object itself where the coordinates of the upper left corner point of the object being 0,0.
9. **TEXTINPUT** : Indicates the text value input action on the object of the given pertinent object type.
10. **ITEMSELECT** : Indicates the drop-down list item selection action on the object of the given pertinent object type.
11. **ITEMDESELECT** : Indicates the drop-down list item deselection action on the object of the given pertinent object type.
12. **EXPAND** : Indicates to expand a hidden section or a dropdown via the left mouse button click action on the object of the given pertinent object type. The mouse click point coordinates “x,y” may be optionally given in the **ExtraParam** parameter, relative to the object itself where the coordinates of the upper left corner point of the object being 0,0.
13. **COLLAPSE** : Indicates to collapse a hidden section or a dropdown via the left mouse button click action on the object of the given pertinent object type. The mouse click point coordinates “x,y” may be optionally given in the **ExtraParam** parameter, relative to the object itself where the coordinates of the upper left corner point of the object being 0,0.
14. **DATEPICK** : Indicates the date picking action on the object of the given pertinent object type. The date must be a valid date string given in the supported date format by the system.
15. **SETON** : Indicates the setting of the check marking **ON** action or the setting of the option/radio/input-switch/select button **ON** action on the object of the given pertinent object type.
16. **SETOFF** : Indicates the setting of the check marking **OFF** action or the setting of the option/radio/input-switch/select button **OFF** action on the object of the given pertinent object type.
17. **HSCROLL** : Indicates to perform the horizontal scrolling action (left, right) on the object of the given pertinent object type. Any one of the predefined key words **GOLEFT**, **GORIGHT, PAGELEFT, PAGERIGHT, SCROLLMIN** or **SCROLLMAX** must be given in the **ExtraParam** parameter for horizontal scrolling. An appropriate integer value, i.e. 2, 5, 12, 20 and so on, can be given in the **ValueToBeSet** parameter to indicate the repeat count of the given scrolling action.
18. **VSCROLL** : Indicates to perform the vertical scrolling action (up, down) on the object of the given pertinent object type. Any one of the predefined key words **GOUP, GODOWN, PAGEUP, PAGEDOWN, SCROLLMIN** or **SCROLLMAX** must be given in the **ExtraParam** parameter for vertical scrolling. An appropriate integer value, i.e. 2, 5, 12, 20 and so on, can be given in the **ValueToBeSet** parameter to indicate the repeat count of the given scrolling action.
19. **SETBLANK** : Indicates the blank out action on the object of the given pertinent object type. In this case the existing value is set to nothing or in other words blanked out if the system permits.
20. **COLUMNSORT** : Indicates to perform a column sort in ascending or descending order by clicking on the given column header of the given web table. The sort order itself is indicated by the key words **ASCN** (ascending, **DEFAULT**) or **DSCN** (descending) passed in the **ValueToBeSet** parameter of the pertinent method call.
21. **VALUESET** : Indicates to set the given value on the object of the given pertinent object type, i.e. **SLIDERBAR**.

**Property Name Key Words**

The following key words are the valid property name parameters for the **ObjectPropertyValueGet** method in the test automation framework. These key words represent the valid property names of the different web objects on the currently active web page whose corresponding values can be obtained by the **ObjectPropertyValueGet** method call.

1. **DEFAULT** : Indicates default property value retrieval for the named object for the given object type as defined in the **ObjectPropertyValueGet** method.
2. **CURRTEXT** : Indicates the currently contained and always displayed text string in the object of the given object type.
3. **TITLETEXT** : Indicates the title text property of the object of the given object type. This property may be hidden or displayed depending on the type of the object. When used with the extra property identifiers (**ExtraParam** parameter) **CUSTOMERID** and **CUSTOMERNAME**, then the pertinent method call, like **ObjectPropertyValueGet** returns the shown customer id and customer name from the title text.
4. **GROUPTITLETEXT** : Indicates the group title text property (mat-card-title) of the given object. This title text belongs to the group container box (mat-card) within which the given object appears as one of the members.
5. **CURRITEM** : Indicates the currently selected and shown item string in the object of the given object type.
6. **CURRSTATE** : Indicates the current state of the object of the given object type, such as **ON, OFF, ACTIVE, INACTIVE, DISABLED, ENABLED, CLICKABLE, VISIBLE, ABSENT, FOCUSED** and so on.
7. **BGCOLOR** : Indicates the current background color in HEX format for the object of the given object type.
8. **FGCOLOR** : Indicates the current foreground color in HEX format for the object of the given object type.
9. **TXTCOLOR** : Indicates the text color in HEX format for the object of the given object type.
10. **OBJWIDTH** : Indicates the current width of the object of the given object type in the default measuring unit of the system.
11. **OBJHEIGHT** : Indicates the current height of the object of the given object type in the default measuring unit of the system.
12. **OBJLABEL** : Indicates the currently shown label on the object of the given object type.
13. **OBJICON** : Indicates the icon name or the image source property name of the shown icon/image on the object of the given object type.
14. **WRNLABEL** : Indicates the currently shown warning label on the object of the given object type, i.e. mandatory field value warning message shown under the object when left blank.
15. **URLSTRING** : Indicates the currently displayed URL address string in the address bar of the currently active Web page. When used with the extra property identifiers (**ExtraParam** parameter) **CUSTOMERID** and **CUSTOMERNAME**, then the pertinent method call, like **ObjectPropertyValueGet** returns the shown customer id and customer name from the application URL.
16. **XCOORD** : Indicates the current X coordinate of the object of the given object type relative to the web page containing the object where the coordinates of the upper left corner point of the web page being 0,0.
17. **YCOORD** : Indicates the current Y coordinate of the object of the given object type relative to the web page containing the object where the coordinates of the upper left corner point of the web page being 0,0.

**The Mapping Table Between The Object Type And The Parameters Of The ObjectActionSet Method**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Object Type** | **Action Type** | **Value To Be Set** | **Extra Action Indicator Parameter** |
| **1.** | **TEXTBOX** | **TEXTINPUT (DEFAULT)** | “John Doe”, **BLANK** (Resets to no selection) | **NULL** |
|  |  | **LCLICK, RCLICK** | **NULL**, **BLANK** (Resets to no selection) | **NULL** |
|  |  | **MOUSEOVER** | **NULL (Optional “x,y” Cartesian values in ExtraParam)** | **NULL** |
|  |  | **SETBLANK** | **NULL**, **BLANK** (Resets to no selection) | **NULL** |
| **2.** | **LISTBOX** | **ITEMSELECT (DEFAULT)** | “Single Account”, **BLANK** (Resets to no selection) | **NULL** |
|  |  | **LCLICK, RCLICK** | **NULL**, **BLANK** | **NULL** |
|  |  | **MOUSEOVER** | **NULL (Optional “x,y” Cartesian values in ExtraParam)** | **NULL** |
|  |  | **SETBLANK** | **NULL**, **BLANK** (Resets to no selection) | **NULL** |
| **3.** | **COMBOBOX** | **ITEMSELECT (DEFAULT)** | “Double Account”, **BLANK**, **ALL** (Selects all checkboxes) | **NULL**, **RETAIN** (Dropdown list left as is without closing) |
|  |  | **ITEMDESELECT** | “Individual single” (Item with checkbox is deselected) | **NULL**, **RETAIN** (Dropdown list left as is without closing) |
|  |  | **LCLICK, RCLICK** | **NULL**, **BLANK** | **NULL** |
|  |  | **MOUSEOVER** | **NULL** | **NULL,** “x,y” (Optional Cartesian values) |
|  |  | **TEXTINPUT** | “USD” (Types in search filter text box) | **NULL**, **RETAIN** (Dropdown list left as is without closing) |
|  |  | **SETBLANK** | **NULL**, **BLANK** (Resets to no selection) | **NULL** |
| **4.** | **DROPDOWN** | **ITEMSELECT (DEFAULT)** | “Joint Account” | **NULL**, **RETAIN** (Dropdown list left as is without closing) |
|  |  | **ITEMDESELECT** | “BDT” (Item with checkbox is deselected) | **NULL**, **RETAIN** (Dropdown list left as is without closing) |
| **5.** | **PUSHBUTTON** | **LCLICK (DEFAULT)** | **NULL** | **NULL** |
|  |  | **MOUSEOVER** | **NULL** | **NULL,** “x,y” (Optional Cartesian values) |
| **6.** | **SELECTBUTTON** | **LCLICK (DEFAULT)** | **NULL, ACTIVE** (Optional for **LCLICK**) | **NULL** |
|  |  | **MOUSEOVER** | **NULL** | **NULL,** “x,y” (Optional Cartesian values) |
| **7.** | **ICONBUTTON** | **LCLICK (DEFAULT)** | **NULL** | **NULL** |
|  |  | **MOUSEOVER** | **NULL** | **NULL,** “x,y” (Optional Cartesian values) |
| **8.** | **RADIOBUTTON** | **SETON (DEFAULT), SETOFF, LCLICK** | **NULL, ON** or **OFF** (Optional for **LCLICK**) | **NULL, “Managing Director”, “Chairman”, “Director”** (actual labels for dynamic/configurable radio buttons) |
| **9.** | **CHECKBOX** | **SETON (DEFAULT), SETOFF, LCLICK** | **NULL, ON** or **OFF** (Optional for **LCLICK**) | **NULL** |
| **10.** | **CALENDARBOX** | **DATEPICK (DEFAULT)** | 31-01-2016 (DD-MM-YYYY) | **NULL** |
|  |  | **LCLICK, RCLICK** | **NULL, BLANK** (Resets to no selection) | **NULL** |
|  |  | **MOUSEOVER** | **NULL** | **NULL,** “x,y” (Optional Cartesian values) |
| **11.** | **CALENDARCTRL** | **DATEPICK (DEFAULT)** | 31-01-2016 (DD-MM-YYYY) | **NULL** |
| **12.** | **DIALOGPAGE** | **XCLOSE (DEFAULT)** | **NULL** | **NULL** |
| **13.** | **ACTIVEPAGE** | **HSCROLL** | **NULL**, 2, 5, 12, 20 etc. (Any integer to indicate repeat count) | **GOLEFT, GORIGHT, PAGELEFT, PAGERIGHT, SCROLLMAX, SCROLLMIN** |
|  |  | **VSCROLL** | **NULL**, 2, 5, 12, 20 etc. (Any integer to indicate repeat count) | **GOUP, GODOWN, PAGEUP, PAGEDOWN, SCROLLMAX, SCROLLMIN** |
| **14.** | **TOGGLEBUTTON** | **SETON (DEFAULT), SETOFF, LCLICK** | **NULL, ON** or **OFF** (Optional for **LCLICK**) | **NULL** |
| **15.** | **EXPANDBUTTON** | **EXPAND (DEFAULT), COLLAPSE, LCLICK** | **NULL, ON** or **OFF** (Optional for **LCLICK**) | **NULL** |
| **16.** | **WEBLINK** | **LCLICK (DEFAULT)** | **NULL** | **NULL** |
| **17.** | **HEADERTAB** | **LCLICK (DEFAULT)** | **NULL, ACTIVE** (Optional for **LCLICK**) | **NULL** |
| **18.** | **MENUBUTTON** | **EXPAND (DEFAULT), COLLAPSE, LCLICK** | “Menu item”, **NULL, ON** or **OFF** (Optional for **LCLICK**) | **NULL** |
| **19.** | **BUTTONBOX** | **LCLICK (DEFAULT)** | **NULL** | **NULL** |
| **20.** | **ONOFFSWITCH** | **SETON (DEFAULT), SETOFF, LCLICK** | **NULL, ON** or **OFF** (Optional for **LCLICK**) | **NULL** |
| **21.** | **FILEIMAGE** | **LCLICK (DEFAULT)** | **NULL** | **NULL** |
| **22.** | **DROPDOWNITEM** | **LCLICK (DEFAULT)** | **NULL** | **NULL** |
| **23.** | **BROWSERTAB** | **LCLICK (DEFAULT)** | **NULL** | **NULL** |
| **24.** | **POPUPBOX** | **XCLOSE (DEFAULT)** | **NULL** | **NULL** |
| **25.** | **DATALISTITEM** | **LCLICK (DEFAULT)** | **EDIT, DELETE, SAVE, VIEW** (any button type to click on for the matching ref. string in the ExtraParam parameter) | **NAME:”MR. X”|INSTANCENO:”2”**  **NAME:”MR. Y”** (any ref. string to pick and/or click on) |
| **26.** | **SLIDERBAR** | **VALUESET (DEFAULT)** | **30, 40, 50, MIN, MAX** (share percentage value) | **NULL** |
| **27.** | **WEBTABLE** | **COLUMNSORT (DEFAULT)** | **ASCN (DEFAULT)**, **DSCN** | (Used by Web Table specific methods only) |
| **28.** | **DATACELL** | **LCLICK (DEFAULT), RCLICK, DCLICK** | **NULL** | (Used by Web Table specific methods only) |

**The Mapping Table Between The Object Type And The Parameters Of The ObjectPropertyValueGet Method**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Object Type** | **Property Name** | **Sample Property Value Output** | **Extra Property Indicator Parameter** |
| **1.** | **TEXTBOX** | **CURRTEXT (DEFAULT)** | John Doe | **NULL** |
|  |  | **CURRSTATE** | **ENABLED**, **DISABLED** | **NULL** |
|  |  | **OBJWIDTH, OBJHEIGHT** | 30,10 | **NULL** |
|  |  | **OBJLABEL** | Last Name\* | **NULL** |
|  |  | **WRNLABEL** | Last Name is required. | **NULL** |
|  |  | **GROUPTITLETEXT** | Basic Information: (mat-card-title) | **NULL** |
|  |  | **XCOORD, YCOORD** | 40, 30 | **NULL** |
| **2.** | **LISTBOX** | **CURRITEM (DEFAULT)** | Dhaka | **NULL** |
|  |  | **CURRSTATE** | **ENABLED**, **DISABLED** | **NULL** |
|  |  | **OBJLABEL** | Division\* | **NULL** |
|  |  | **WRNLABEL** | Division is required. | **NULL** |
|  |  | **OBJWIDTH, OBJHEIGHT** | 30,10 | **NULL** |
|  |  | **GROUPTITLETEXT** | Basic Information: (mat-card-title) | **NULL** |
|  |  | **XCOORD, YCOORD** | 40,30 | **NULL** |
| **3.** | **COMBOBOX** | **CURRITEM (DEFAULT)** | Individual single | **NULL** |
|  |  | **CURRSTATE** | **ENABLED**, **DISABLED** | **NULL** |
|  |  | **OBJWIDTH, OBJHEIGHT** | 30,10 | **NULL** |
|  |  | **OBJLABEL** | Eligible customer type\* | **NULL** |
|  |  | **WRNLABEL** | Eligible customer type is required. | **NULL** |
|  |  | **GROUPTITLETEXT** | Basic Information: (mat-card-title) | **NULL** |
|  |  | **XCOORD, YCOORD** | 40,30 | **NULL** |
| **4.** | **DROPDOWN** | **CURRITEM (DEFAULT)** | USD (Selected item or items, in CSV format) | **NULL** |
|  |  | **CURRTEXT** | BDT,USD,EUR,JPY (All list items, in CSV format) | **NULL** |
|  |  | **CURRSTATE** | **ON** or **OFF** | **ALL**, “USD”, “Individual single” (Checkbox list item names) |
| **5.** | **PUSHBUTTON** | **OBJLABEL (DEFAULT)** | Login | **NULL** |
|  |  | **CURRSTATE** | **ENABLED**, **DISABLED** | **NULL** |
|  |  | **BGCOLOR** | #F1F1F4 | **NULL** |
|  |  | **TXTCOLOR** | #000000 | **NULL** |
|  |  | **GROUPTITLETEXT** | Basic Information: (mat-card-title) | **NULL** |
|  |  | **FGCOLOR** | #000000 | **NULL** |
| **6.** | **SELECTBUTTON** | **OBJLABEL (DEFAULT)** | Pending | **NULL** |
|  |  | **CURRSTATE** | **ON**, **OFF**, **DISABLED** | **NULL** |
|  |  | **BGCOLOR** | #F1F1F4 | **NULL** |
|  |  | **TXTCOLOR** | #000000 | **NULL** |
|  |  | **GROUPTITLETEXT** | Basic Information: (mat-card-title) | **NULL** |
|  |  | **FGCOLOR** | #000000 | **NULL** |
| **7.** | **ICONBUTTON** | **OBJICON (DEFAULT)** | ui-icon-edit | **NULL** |
|  |  | **CURRSTATE** | **ENABLED**, **DISABLED** | **NULL** |
|  |  | **BGCOLOR** | #F1F1F4 | **NULL** |
|  |  | **OBJLABEL** | Edit | **NULL** |
|  |  | **GROUPTITLETEXT** | Basic Information: (mat-card-title) | **NULL** |
|  |  | **FGCOLOR** | #000000 | **NULL** |
| **8.** | **EXPANDBUTTON** | **OBJLABEL (DEFAULT)** | Search a customer | **NULL** |
|  |  | **CURRSTATE** | **ON**, **OFF**, **DISABLED** | **NULL** |
|  |  | **BGCOLOR** | #F1F1F4 | **NULL** |
|  |  | **TXTCOLOR** | #000000 | **NULL** |
|  |  | **GROUPTITLETEXT** | Basic Information: (mat-card-title) | **NULL** |
|  |  | **FGCOLOR** | #000000 | **NULL** |
| **9.** | **TOGGLEBUTTON** | **OBJLABEL (DEFAULT)** | NID | **NULL** |
|  |  | **CURRSTATE** | **ON**, **OFF**, **DISABLED** | **NULL** |
|  |  | **BGCOLOR** | #F1F1F4 | **NULL** |
|  |  | **TXTCOLOR** | #000000 | **NULL** |
|  |  | **GROUPTITLETEXT** | Basic Information: (mat-card-title) | **NULL** |
|  |  | **FGCOLOR** | #000000 | **NULL** |
| **10.** | **RADIOBUTTON** | **CURRSTATE (DEFAULT)** | **ON**, **OFF**, **DISABLED** | **NULL**, **“Managing Director”, “Chairman”, “Director”** (actual labels for dynamic/configurable radio buttons) |
|  |  | **GROUPTITLETEXT** | Basic Information: (mat-card-title) | **NULL** |
| **11.** | **CHECKBOX** | **CURRSTATE (DEFAULT)** | **ON**, **OFF**, **DISABLED** | **NULL** |
|  |  | **GROUPTITLETEXT** | Basic Information: (mat-card-title) | **NULL** |
| **12.** | **ONOFFSWITCH** | **CURRSTATE (DEFAULT)** | **ON**, **OFF**, **DISABLED** | **NULL** |
|  |  | **GROUPTITLETEXT** | Basic Information: (mat-card-title) | **NULL** |
| **13.** | **CALENDARBOX** | **CURRTEXT (DEFAULT)** | 31-01-2016 | **NULL** |
|  |  | **CURRSTATE** | **ENABLED**, **DISABLED** | **NULL** |
|  |  | **GROUPTITLETEXT** | Basic Information: (mat-card-title) | **NULL** |
| **14.** | **WEBLINK** | **URLSTRING (DEFAULT)** | <https://www.mislbd.com/> | **NULL** |
|  |  | **CURRTEXT** | Home Link | **NULL** |
|  |  | **CURRSTATE** | **ENABLED**, **DISABLED** | **NULL** |
|  |  | **GROUPTITLETEXT** | Basic Information: (mat-card-title) | **NULL** |
| **15.** | **HEADERTAB** | **CURRSTATE (DEFAULT)** | **ON**, **OFF**, **DISABLED** | **NULL** |
|  |  | **CURRTEXT** | Documents | **NULL** |
| **16.** | **MENUBUTTON** | **CURRSTATE (DEFAULT)** | **ON**, **OFF**, **DISABLED** | **NULL** |
|  |  | **OBJICON** | fa-caret-down | **NULL** |
| **17.** | **BUTTONBOX** | **OBJLABEL (DEFAULT)** | Admin, CIS, Deposit | **NULL** |
| **18.** | **TOOLTIPBOX** | **CURRTEXT (DEFAULT)** | Username | **NULL** |
| **19.** | **POPUPBOX** | **CURRTEXT (DEFAULT)** | Error! Invalid Username | **NULL** |
| **20.** | **ACTIVEPAGE** | **TITLETEXT (DEFAULT)** | Customer | **NULL** |
|  |  | **URLSTRING** | <http://192.168.1.140:30080/ci/#/customer/personal/6817?customerName=John%20Doe> | **NULL, CUSTOMERID** (Extracted from URL)**, CUSTOMERNAME** (Extracted from URL) |
| **21.** | **DIALOGPAGE** | **TITLETEXT (DEFAULT)** | Customer Information File | **NULL** |
| **22.** | **TITLETOOLBAR** | **TITLETEXT (DEFAULT)** | Single Customer: John Doe | Customer ID: 6817 | **NULL** |
| **23.** | **CALENDARCTRL** | **CURRTEXT (DEFAULT)** | (Currently selected date in the supported format) | **NULL** |
| **24.** | **GROUPTITLEBAR** | **CURRSTATE (DEFAULT)** | **VISIBLE, ABSENT** | i.e. “Basic Information:”, “FATCA:”, “Upload document” (mat-card-title) |
| **25.** | **FILEUPLOADBOX** | **TITLETEXT (DEFAULT)** | Uploaded files: | **NULL** |
|  |  | **CURRSTATE** | **VISIBLE, ABSENT** |  |
| **26.** | **FILEIMAGE** | **CURRSTATE (DEFAULT)** | **VISIBLE, ABSENT** | **NULL** |
|  |  | **GROUPTITLETEXT** | Upload document (mat-card-title) |  |
| **27.** | **DROPDOWNITEM** | **CURRTEXT (DEFAULT)** | Logout | **NULL** |
| **28.** | **BROWSERTAB** | **CURRTEXT (DEFAULT)** | ababil-customer | **NULL** |
| **29.** | **SLIDERBAR** | **CURRTEXT (DEFAULT)** | 35% | **NULL** |
| **30.** | **DATALISTITEM** | **CURRSTATE (DEFAULT)** | **VISIBLE, ABSENT, DISABLED** | **INSTANCENO:”1”** |
|  |  | **NAME**, **FATHERNAME**, **MOTHERNAME**, **MOBILE**, **EMAIL**, **TIN**, **NID, DESIGNATION** (Owner Type)**, CATEGORY** (Individual, Organization etc.)**, SHAREAMOUNT** (Percentage/ration of the share)  **OBJICON, OBJLABEL** (For **EDIT**, **VIEW**, **DELETE**, **SAVE** buttons) | Value of the given property name is returned in the output parameter. | **INSTANCENO:”1”**  **NAME:”MS ANNIE”|INSTANCENO:”2”**  **MOBILE:”01701234567”**  **EMAIL:”** [**abcd@email.com**](mailto:abcd@email.com) **“**  **TIN: “1234567890123”**  **NID:”1234567890123”**  **MOBILE:”01701234567”|EDIT** (For **OBJLABEL**, **CURRSTATE**, **OBJICON** properties) |
| **31.** | **DATACELL** | **CURRTEXT (DEFAULT)** | Active | (Used by Web Table specific methods only) |
| **32.** | **WEBTABLE** | **GROUPTITLETEXT (DEFAULT)** | Financial Information: | **NULL** |

**The Test Automation Framework Method Prototypes And Details**

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

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

**Category 1: Application Feature Specific Methods**

1. **AppLaunch (String BrowserType, Boolean CheckFirst, String ExtraParam);**

This is an application feature specific method performing feature specific task only. It launches the AUT (Application Under Test) login page in the given web browser.

**BrowserType = CHROME**(Google, **DEFAULT**), **FIREFOX**(Mozilla), **MSEDGE**(Microsoft IE), **OPERA**(Otello Corp.)

**CheckFirst = TRUE** or **YES(DEFAULT), FALSE** or **NO**. If set to **TRUE** or **YES**, then it indicates to check first if the application is already launched in the given browser type and the Ababil Home tab (**GuiAppHomePageTab**) or the Ababil Login tab (**GuiLoginTab**) exists and if so then it switches to the existing tab. It uses the Web URL stored in the global parameter **GblAppWebUrl** to launch Ababil application in the default tab if not launched already. If set to **FALSE** or **NO**, then the method simply launches the application in a new browser window of the given browser type in the default tab.

1. **AppLogin (String UserName, String PassWord, Boolean CheckFirst, String ExtraParam);**

This is an application feature specific method performing feature specific task only. It logs in to the AUT with the given user credentials and initiates the AUT home page. Upon successful logging in, sets the string global variable **GblLoggedInUser** to the user name to indicate that this user is currently logged in to the application.

**UserName** = Contains the valid user name for logging in. The **DEFAULT** value is the predefined value in the global parameter **GblLoginUserName**.

**PassWord** = Contains the valid password for logging in. The **DEFAULT** value is the predefined value in the global parameter **GblLoginPassWord**.

**CheckFirst = TRUE** or **YES(DEFAULT), FALSE** or **NO**. If set to **TRUE** or **YES**, then it indicates to check first if the user is already logged in and the Ababil Home tab (**GuiAppHomePageTab**) exists and if so then it switches to the tab. If a different user is logged in, then it logs the user out and logs back in with the given user. If set to **FALSE** or **NO**, then the method simply makes the login attempt regardless.

1. **AppLogout (String LogOutOption, Boolean CheckFirst, String ExtraParam);**

This is an application feature specific method performing feature specific task only. It logs the current user out from the AUT. Upon successful logging out, it sets the string global variable **GblLoggedInUser** to blank (“”) to indicate that no user is currently logged in to the application.

**LogOutOption = HOMEPAGE**(From Application Home Page,**DEFAULT**), **MODULEPAGE**(From Active Module Home Page).

**CheckFirst = TRUE** or **YES(DEFAULT), FALSE** or **NO**. If set to **TRUE** or **YES**, then it indicates to check first if the user is already logged out and the Ababil Home tab (**GuiAppHomePageTab**) or the active module tab does not exist. If so, then the method simply returns gracefully doing nothing. If set to **FALSE** or **NO**, then the method simply makes the logout attempt regardless.

1. **AppDocInfoNidDataSet (String NidNumber, String NidIssueDate, String DocFileName, String DocFilePath, String ExtraParam);**

This is an application feature specific method performing feature specific task only. It fills in the given field values and attaches the given NID document to the AUT. The Document Information page must be the active page for this method to work properly.

**NidNumber =** Contains the NID 13-digit number. This parameter value is mandatory and cannot be **NULL** or **DEFAULT**.

**NidIssueDate** = Contains the NID issue date in a supported date format. The **NULL** value means no issue date.

**DocFileName** = Contains the name of the NID image document file to be attached. The **NULL** value means no file to attach.

**DocFilePath** = Contains the folder path name of the NID image document file to be attached. The **DEFAULT** value indicates to take the folder path name from the **GblDocAttachmentsFolderPath** global parameter.

1. **AppDocInfoDrivingLicenseDataSet (String LicenseNumber, String LicenseIssueDate, String LicenseExpiryDate, String DocFileName, String DocFilePath, String ExtraParam);**

This is an application feature specific method performing feature specific task only. It fills in the given field values and attaches the given driving license document to the AUT. The Document Information page must be the active page for this method to work properly.

**LicenseNumber =** Contains the Driving License number. This parameter value is mandatory and cannot be NULL or DEFAULT.

**LicenseIssueDate** = Contains the Driving License issue date in a supported date format. The **NULL** value means no issue date.

**LicenseExpiryDate** = Contains the Driving License expiry date in a supported date format. The **NULL** value means no expiry date.

**DocFileName** = Contains the name of the Driving License image document file to be attached. The **NULL** value means no file to attach.

**DocFilePath** = Contains the folder path name of the Driving License image document file to be attached. The **DEFAULT** value indicates to take the folder path name from the **GblDocAttachmentsFolderPath** global parameter.

1. **AppDocInfoPassportDataSet (String PassportNumber, String PassportIssueDate, String PassportExpiryDate, String PassportIssuePlace, String DocFileName, String DocFilePath, String ExtraParam);**

This is an application feature specific method performing feature specific task only. It fills in the given field values and attaches the given Passport document to the AUT. The Document Information page must be the active page for this method to work properly.

**PassportNumber =** Contains the Passport number. This parameter value is mandatory and cannot be **NULL** or **DEFAULT**.

**PassportIssueDate** = Contains the Passport issue date in a supported date format. The **NULL** value means no issue date.

**PassportExpiryDate** = Contains the Passport expiry date in a supported date format. The **NULL** value means no expiry date.

**PassportIssuePlace** = Contains the Passport issue place string. The **NULL** value means no issue place.

**DocFileName** = Contains the name of the Passport image document file to be attached. The **NULL** value means no file to attach.

**DocFilePath** = Contains the folder path name of the Passport image document file to be attached. The **DEFAULT** value indicates to take the folder path name from the **GblDocAttachmentsFolderPath** global parameter.

1. **AppDocInfoSmartCardDataSet (String SmartCardNumber, String SmartCardIssueDate, String DocFileName, String DocFilePath, String ExtraParam);**

This is an application feature specific method performing feature specific task only. It fills in the given field values and attaches the given Smart Carddocument to the AUT. The Document Information page must be the active page for this method to work properly.

**SmartCardNumber =** Contains the Smart Card 10-digit number. This parameter value is mandatory and cannot be **NULL** or **DEFAULT**.

**SmartCardIssueDate** = Contains the Smart Card issue date in a supported date format. The **NULL** value means no issue date.

**DocFileName** = Contains the name of the Smart Card image document file to be attached. The **NULL** value means no file to attach.

**DocFilePath** = Contains the folder path name of the Smart Card image document file to be attached. The **DEFAULT** value indicates to take the folder path name from the **GblDocAttachmentsFolderPath** global parameter.

1. **AppDocInfoTinDataSet (String TinNumber, String DocFileName, String DocFilePath, String ExtraParam);**

This is an application feature specific method performing feature specific task only. It fills in the given field values and attaches the given TIN document to the AUT. The Document Information page must be the active page for this method to work properly.

**TinNumber =** Contains the TIN 12-digit number. This parameter value is mandatory and cannot be **NULL** or **DEFAULT**.

**DocFileName** = Contains the name of the TIN image document file to be attached. The **NULL** value means no file to attach.

**DocFilePath** = Contains the folder path name of the TIN image document file to be attached. The **DEFAULT** value indicates to take the folder path name from the **GblDocAttachmentsFolderPath** global parameter.

1. **AppDocInfoTradeLicenseDataSet (String TradeLicNumber, String TradeLicIssuePlace, String TradeLicIssueAuthority, String TradeLicBusinessType, String TradeLicExpiryDate, String DocFileName, String DocFilePath, String ExtraParam);**

This is an application feature specific method performing feature specific task only. It fills in the given field values and attaches the given Trade License document to the AUT. The Document Information page must be the active page for this method to work properly.

**TradeLicenseNumber =** Contains the valid Trade License number. This parameter value is mandatory and cannot be **NULL** or **DEFAULT**.

**TradeLicIssuePlace** = Contains the Trade License issue place string. The **NULL** value means no issue place.

**TradeLicIssueAuthority** = Contains the Trade License authority string. The **NULL** value means no authority.

**TradeLicBusinessType** = Contains the Trade License business type string. This parameter value is mandatory and cannot be **NULL** or **DEFAULT**.

**TradeLicExpiryDate** = Contains the Trade License expiry date in a supported date format. This parameter value is mandatory and cannot be **NULL** or **DEFAULT**.

**DocFileName** = Contains the name of the Trade License image document file to be attached. The **NULL** value means no file to attach.

**DocFilePath** = Contains the folder path name of the Trade License image document file to be attached. The **DEFAULT** value indicates to take the folder path name from the **GblDocAttachmentsFolderPath** global parameter.

1. **AppHomePageModuleSet (String ModuleObjectName, Boolean CheckFirst, String ExtraParam);**

This is an application feature specific method performing feature specific task only. It selects the given application module from the application home page and then navigates to the module tab.

**ModuleObjectName** = GuiCisModule, GuiDepositModule, GuiAdminModule, GuiCashModule, GuiFinanceModule, GuiSanctionLimitModule (All are defined as objects in object repository)

**CheckFirst = TRUE** or **YES(DEFAULT), FALSE** or **NO**. If set to **TRUE** or **YES**, then it indicates to check first if the module is already set and the module tab exists. If so, then the method simply makes the module tab active and returns gracefully. If set to **FALSE** or **NO**, then the method simply selects and sets the named module active regardless.

1. **AppPageMenuSelect (String MenuType, String MenuPath, String ExtraParam);**

This is an application feature specific method performing feature specific task only. It initiates the dropdown menu list for the given menu type and then selects the given menu item from the menu.

**MenuType** = Indicates the type of menu to be initiated, i.e. **MAINMENU (DEFAULT), USERMENU, HOMEMENU, DROPMENU**. The **MAINMENU** indicates the principal menu of each Ababil module initiated by clicking on the side arrow icon on the left upper corner of the page. The **USERMENU** indicates the logged on user menu initiated by clicking on the user head icon inside the upper part of the main menu. The **HOMEMENU** indicates the logged on user menu initiated by clicking on the user head icon on the upper right corner of the Ababil home page. The **DROPMENU** indicates the dropdown menu initiated by clicking on the button containing a dropdown symbol depicted by the down arrow icon appearing in various pages, i.e. button labeled “Options”.

**MenuPath** = Indicates the full hierarchical path of the menu items to be selected. The path delimiter is “/”, i.e. “Deposit/Demand Deposit/Product” (for **MAINMENU)** or “Logout” (for **USERMENU** or **HOMEMENU**). The menu item names are case, spelling and space sensitive, so must be supplied exactly the same as they appear on the screen. For the **DROPMENU** type, the very first element of the path must be the label of the button that initiates the dropdown menu, i.e. “Options/Add Nominee”, here “Options” is the label of the dropdown menu initiator button and the rest are the menu item names exactly as they appear on the screen in the menu selection hierarchy.

**Category 2: Browser Window And Web Page Specific Methods**

1. **BrowserPageViaUrlOpen (String WebUrlAddress, String BrowserType, Boolean NewBrowserWindow, Boolean NewPageTab, String ExtraParam);**

This method opens the browser window of the given browser type.

**WebUrlAddress** = Contains the Web URL address of the web page to be opened.

**BrowserType =** Indicates the type of the browser to open. The following are the keywords representing valid browser types that are supported, i.e. **CHROME**(Google, **DEFAULT**), **FIREFOX**(Mozilla), **MSEDGE**(Microsoft IE), **OPERA**(Otello Corp.).

**NewBrowserWindow =** It is a Boolean parameter indicating whether to open a new browser window or use the existing one. These are the Boolean keyword indicators, **TRUE or YES(DEFAULT), FALSE or NO.**

**NewPageTab =** It is a Boolean parameter indicating whether to open a new tab in the browser window or use the existing one. These are the Boolean keyword indicators, **TRUE or YES(DEFAULT), FALSE or NO.**

1. **BrowserWindowClose (String BrowserWindowName, String ExtraParam);**

This method closes the named browser window.

**BrowserWindowName** = The **DEFAULT** keyword indicates currently active Browser window.

1. **BrowserWindowMinimize (String BrowserWindowName, String ExtraParam);**

This method minimizes the given browser window from its current state. If already minimized, then the method simply returns to the calling routine successfully.

**BrowserWindowName** = The **DEFAULT** keyword indicates currently active Browser window.

1. **BrowserWindowMaximize (String BrowserWindowName, String ExtraParam);**

This method maximizes the given browser window from its current state. If already maximized, then the method simply returns to the calling routine successfully.

**BrowserWindowName** = The **DEFAULT** keyword indicates currently active Browser window.

**ExtraParam** = The key word **FULLSCREEN** indicates to maximize to full screen mode (F11 key toggle to full screen and restore).

1. **BrowserWindowRestore (String BrowserWindowName, String ExtraParam);**

This method restores the given browser window to its normal mode from its current state (minimized, maximized or full screen mode). If already restored, then the method simply returns to the calling routine successfully.

**BrowserWindowName** = The DEFAULT keyword indicates currently active Browser window.

1. **BrowserTabSwitch (String TabObjectName, Integer InstanceNum, String ExtraParam);**

This method switches the browser tab from the current tab to the given tab of the given instance of the tab.

**TabObjectName** = **GuiLoginTab, GuiCisTab, GuiAdminTab, GuiDepositTab, GuiCashTab, GuiFinanceTab, GuiSanctionLimitTab, GuiAppHomePageTab** (All are defined as objects in object repository). The **DEFAULT** defaults to the **GuiAppHomePageTab** or the **GuiLoginTab**, whichever one exists.

**InstanceNum** = Indicates the instance number in case of multiple tabs are open by the same name. The **DEFAULT** defaults to 1. The **LAST** keyword indicates the last instance in the order the tabs with the same name are opened.

1. **BrowserTabClose (String TabObjectName, Integer InstanceNum, String ExtraParam);**

This method closes the given tab of the given instance of the tab in the active browser window.

**TabObjectName** = **GuiLoginTab, GuiCisTab, GuiAdminTab, GuiDepositTab, GuiCashTab, GuiFinanceTab, GuiSanctionLimitTab, GuiAppHomePageTab** (All are defined as objects in object repository). The **DEFAULT** defaults to the currently active tab. The **ALL** keyword indicates all of the tabs.

**InstanceNum** = Indicates the instance number in case of multiple tabs are open by the same name. The **DEFAULT** defaults to 1. The **LAST** keyword indicates the last instance in the order the tabs with the same name are opened. The **ALL** keyword indicates all of the open instances of the given tab with the same name.

1. **WebPageRefresh (Integer MaxTimeWait, String ExtraParam);**

This method refreshes the currently active web page. The method keeps trying to detect the completion of the page refresh event until the given maximum wait time expires. As soon as the method detects the completion of the page refresh event within the given time limit, it stops and returns to the calling routine.

**MaxTimeWait** = Maximum time to wait up to for the page refresh to finish. Integer type, time in seconds, the **DEFAULT** defaults to **GblMaxWaitTime** predefined value. The **NULL** or 0 (zero) value indicates no wait time to detect the completion of the page refresh event and the method simply returns successfully to the calling routine right after initiating the page refresh event.

**Category 3: External File Input Output Specific Methods**

1. **ScreenShotCapture (String ScreenShotType, String ImageType, String FileName, String FileNameSuffix, String FileLocation, String ExtraParam);**

This method is used for capturing screen shot of the active window or the desktop for record keeping purpose.

**ScreenShotType = FULLSCREEN**(Full Screen Desktop), **CURRWIN**(Currently Active Browser Window),**CURRPAGE**(Currently Active Web Page Frame Only). The **DEFAULT** value indicates to take the predefined value from the **GblScreenShotType** global parameter which is primarily set to **CURRWIN.**

**ImageType = JPG**(JPEG), **PNG**, **BMP**(Bitmap), **TIF**(TIFF), **PDF**. The **DEFAULT** value indicates to take the predefined value from the **GblScreenShotImageType** global parameter which is primarily set to **JPG.**

**FileName** = Contains the name of the screen shot file without the file extension. This parameter value is mandatory for customized names only. The **DEFAULT** value indicates to take the predefined value from the **GblScreenShotFileName** global parameter which is primarily set to the string value **“ScreenShot”** and dynamically suffixed by the current value of the global parameter **GblScreenShotCount** + 1 during runtime. For single digit count, 0 (zero) is padded as prefix, i.e. “ScreenShot01”…”ScreenShot09”.

**FileNameSuffix** = Any special tag to append to the file name. The System date-time stamp can be used by using the **DATETIMESTAMP, TIMESTAMP, DATESTAMP** key words. The global variables **GblDateFormat** (DDMMYYYY), **GblTimeFormat** (HHMMSS), **GblDateTimeFormat** (DDMMYYYYHHMMSS) contain the current formats and always can be set to new valid ones whenever needed. The **NULL** value indicates no suffix. The **DEFAULT** value indicates to take the System date-time (**DATETIMESTAMP**) in the predefined format from the **GblDateTimeFormat** global parameter. If the **GblDateTimeFormat** is **NULL**, then then the suffix is **NULL** as well.

**FileLocation** = Contains the folder path name of the screen shot file. The **DEFAULT** value indicates to take the folder path name from the **GblScrnShotFolderPath** global parameter**.**

1. **TextOuputLineWrite (String TheOutputLine, String TextToAppend, String FileName, String FileNameSuffix**, **String FileLocation, String E xtraParam);**

This method writes the given text message to the named output file. This is the only method in the framework that does all the printing of the preformatted textual messages to the output files. If the report message is given as a single line text message only and not as one of the array types, then the actual text message is formatted as per the following format and then also added to the **TestDetailLogArray** array only.

**[**HH24:mm:ss:SSS Report Message:\t**TextOuputLineWrite**::\t**TheOutputLine** + **TextToAppend**.\n**]**

**TheOutputLine** = Contains the string to be printed to the named output file. The **TESTREPORTARRAY** and **TESTLOGARRAY** keywords indicate to print all contents of the **TestResultReportArray** and **TestDetailLogArray** global arrays respectively to the named output file.

**TextToAppend** = Any extra tag to append to the output line to be printed in the output file, like for example the pass/fail status of the current test step which is always updated in the global variable **GblStepPassFailStatus**.

**FileName** = Contains the name of the test report or log file with the “.txt” or “.log” file extension respectively. This parameter value is mandatory, so cannot be **NULL**. The **DEFAULT** value indicates to take the predefined file names “TestReportSummary.txt” and “TestReportDetails.log” form the **GblDefaultTestReportFile** and the **GblDefaultTestLogFile** global parameters respectively.

**FileNameSuffix** = Any special tag to append to the file name. The System date-time stamp can be used by using the **DATETIMESTAMP, TIMESTAMP, DATESTAMP** key words. The global variables **GblDateFormat** (DDMMYYYY), **GblTimeFormat** (HHMMSS), **GblDateTimeFormat** (DDMMYYYYHHMMSS) contain the current formats and always can be set to new valid ones whenever needed. The **NULL** value indicates no suffix. The **DEFAULT** value indicates to take the System date-time (**DATETIMESTAMP**) in the predefined format from the **GblDateTimeFormat** global parameter. If the **GblDateTimeFormat** is **NULL**, then then the suffix is **NULL** as well.

**FileLocation** = Contains the folder path name of the test result file. The **DEFAULT** value indicates to take the folder path name from the **GblTestReportFolderPath** global parameter for the test report file and from the **GblTestLogFolderPath** global parameter for the test log file**.** Both of these global parameters must be predefined with valid values (same or different) and cannot be null.

1. **ExcelSheetDataFetch (String FileName, String FileLocation, String DataSheetName, String ExtraParam);**

This generic method reads the named Excel data sheet of the given Excel file from the given location and gets the entire data into an internal hash key based one/two dimensional array as necessary for easier and faster access during runtime. The one/two dimensional array is named from the given data sheet name internally in order to load and hold the data from the Excel data sheet into memory for fast and easy access. The word “Array” is literally concatenated to the array name.

**FileName** = Contains the name of the Excel test data file. This parameter value is mandatory.

**FileLocation** = Contains the folder path name of the Excel test data file. The **DEFAULT** value indicates to take the folder path name from the **GblTestsFolderPath** global parameter.

**DataSheetName** = “TestMain”, “TestParams”, “TestDataSheet01”, “TestDataSheet02”, “TestDataSheet03” and so on.

1. **ExcelSheetCellValueSet (String DataSheetName, String RefSearchText, Integer InstanceNo, String RefColName, String CellColName, String ValueToBeSet, String ExtraParam);**

This method allows resetting of any intended cell value of the named Excel data sheet during runtime as it becomes necessary for the test. This method assumes that given data sheet is already preloaded in to the memory. The method only updates the internal copy of the referred data sheet and the original data sheet is left as is.

**DataSheetName** = “TestParams”, “TestDataSheet01”, “TestDataSheet02”, “TestDataSheet03” and so on.

**RefSearchText** = Indicates the actual reference string to be searched for in the named Excel data sheet in order to reach the intended cell to be set somewhere in the same row.

**InstanceNo** = Indicates the number of occurrence of the actual reference string to be searched for in the named Excel data sheet. If the given occurrence number of the string is not found, then the method fails. The value **DEFAULT** or 1 indicates the very first occurrence of the reference search string. The value **ALL** indicates all occurrences of the reference search string.

**RefColName** = Indicates the name of the column that contains the reference search string. This serves as the filter to further narrow the search within the given column only. The keyword **ALL** indicates to search the entire table row by row and column by column without any filter. The keyword **DEFAULT** indicates to search for the reference string in the first column only.

**CellColName** = Indicates the name of the column containing the intended cell to be located based on the reference search string. This column name is mandatory and must be supplied.

**ValueToBeSet** = The actual value to be set or updated to the intended cell. This parameter is mandatory.

1. **ExcelSheetCellValueGet (String DataSheetName, String RefSearchText, Integer InstanceNo, String RefColName, String CellColName, String [OutCellValue], String ExtraParam);**

This method allows retrieving of any intended cell value of the named Excel data sheet during runtime as it becomes necessary for the test. This method assumes that given data sheet is already preloaded in to the memory. The method always gets the intended cell value from the internal copy of the referred data sheet and not from the original data sheet.

**DataSheetName** = “TestParams”, “TestDataSheet01”, “TestDataSheet02”, “TestDataSheet03” and so on.

**RefSearchText** = Indicates the actual reference string to be searched for in the named Excel data sheet in order to get the intended cell value somewhere in the same row.

**InstanceNo** = Indicates the number of occurrence of the actual reference string to be searched for in the named Excel data sheet. If the given occurrence number of the string is not found, then the method fails. The value **DEFAULT or** 1 indicates the very first occurrence of the reference search string. The value **LAST** indicates the last occurrence of the reference search string.

**RefColName** = Indicates the name of the column that contains the reference search string. This serves as the filter to further narrow the search within the given column only. The keyword **ALL** indicates to search the entire table row by row and column by column without any filter. The keyword **DEFAULT** indicates to search for the reference string in the first column only.

**CellColName** = Indicates the name of the column containing the intended cell to be located based on the reference search string. This column name is mandatory and must be supplied.

**[OutCellValue]** = Holds the output value of the intended cell when the method call returns to the calling routine.

1. **ExcelSheetDataReload (String DataSheetName, String ExtraParam);**

The method reloads the named data sheet again from the same original source in to the same internal array in order to refresh the data in case some prior updates were made to the internal copy via the **ExcelSheetCellValueSet** method call.

**DataSheetName** = “TestMain”, “TestParams”, “TestDataSheet01”, “TestDataSheet02”, “TestDataSheet03” and so on.

1. **WindowsDocumentAttach (String DocFileName, String DocFilePath, String ExtraParam);**

This method performs the attachment of the given file from the given location to the AUT (Application Under Test) via the Windows default file attachment GUI interface. This generic method may be called independently or from the application feature specific methods whenever file attachment is needed. If called independently, then the file upload window must be launched first by the pertinent button press via the **ObjectActionSet** method call. The **ExtraParam** parameter may be used to send keywords like **CANCEL**, **CLOSE** and **SCRNSHOT** for cancelling or closing the file upload window left open or capturing its screenshot with all default values. The keyword **CLEANUP** given in the **ExtraParam** parameter indicates to take the screenshot first and then properly close the file upload window in case if the file not found error occurs during file upload operation, otherwise the file upload window is left open in case of errors.

**DocFileName** = Indicates the name of the file to be attached. The **NULL/DEFAULT** keyword indicates no file name and the file name may be included in the file path parameter below.

**DocFilePath** = Indicates the folder path of the named file to be attached. This parameter is mandatory. The **DEFAULT** keyword indicates to take the predefined path name from the **GblDocAttachmentsFolderPath** global parameter. The **NULL** keyword indicates no file path. If both the file name and the file path parameters are set to **NULL**, then the method only performs action indicated in the **ExtraParam** parameter.

**Category 4: Math Specific Utility Methods**

1. **UtlMathNumbersAdd (String DelimitedNumbers, String [OutSumValue], String ExtraParam);**

This method performs simple addition of the given delimited string of numbers and returns the total sum as output in the given output parameter. If at least one of the numbers comes in floating format, then the sum is returned in floating format as well. If all the numbers come in integer format, then the sum is returned in integer format only.

**DelimitedNumbers** = Contains the string of numbers delimited by the “|” character, i.e. **“1”|”2”|”3”|”4”|”5”** or **“1”|”2.5”|”3”|”4”|”5.0”** or **InputNum1|”2”|InputNum2|”5.0”|InputNum3**. This parameter is mandatory and must have at least two delimited numbers, i.e. **“7”|”8”**.

**[OutSumValue]** = Holds the total sum of the given delimited numbers as output.

**Category 5: Object Data Set, Get And Verify Specific Methods**

1. **ObjectActionSet (String ObjectName, String ActionType, String ValueToBeSet, String ExtraParam);**

This method performs the given action on the given web element and sets the value in the named web element if provided. All kinds of object actions and value setting tasks are performed using this generic method in the framework.

**ObjectName** = Indicates the name of the web element defined in the corresponding object repository property file, i.e. GuiSave, GuiEditProfile and so on.

**ActionType** = Indicates the predefined keyword to represent the action to be performed on the given web element, i.e. **LCLICK, TEXTINPUT, ITEMSELECT, DATEPICK, SETON, SETOFF** and so on. The **DEFAULT** keyword indicates to perform the predefined default action for the given web element.

**ValueToBeSet** = Contains the actual value to be set in the given web element.

1. **ObjectPropertyValueGet (String ObjectName, String PropertyName, String [OutPropertyValue], String ExtraParam);**

This method, by default, retrieves the currently set value of the given web element and returns the obtained value in the output parameter. Also this generic method is used to retrieve value for all kinds of valid and framework supported property of the named web element.

**ObjectName** = Indicates the name of the web element defined in the corresponding object repository property file, i.e. GuiLastName, GuiDateOfBirth and so on.

**PropertyName** = Indicates the predefined keyword to represent the name of the property for which the value to be obtained for the given web element, i.e. **CURRTEXT, CURRITEM, CURRSTATE, TITLETEXT, URLSTRING** and so on. The **DEFAULT** keyword indicates to obtain the predefined default property value for the given web element.

**[OutPropertyValue]** = It is an output parameter that holds the actual obtained value of the given web element property as output.

**ExtraParam** = This extra parameter may be used to further qualify the **PropertyName** parameter with a sub property name or extra indicator keyword. For example, to obtain the customer id or customer name from the **PropertyName** indicator **URLSTRING** keyword, the **CUSTOMERID** or **CUSTOMERNAME** keywords can used in this parameter. Similarly, to obtain the color code of the label text property indicator **OBJLABEL** keyword, the keyword **TXTCOLOR** can be used in this parameter.

1. **ObjectCurrentStateAssert (String ObjectName, String StateToAssert, String ExtraParam);**

This method asserts the given state of the given object as the current state. If the given state is verified as the current state, then the method returns **PASS** otherwise it returns **FAIL**.

**ObjectName** = Contains the GUI name of the object as represented in the pertinent object repository property file.

**StateToAssert** = Contains the key words representing the current state of the object to be asserted, i.e. **ENABLED** (can be operated on), **DISABLED** (can’t be operated on), **ACTIVE** (having the focus), **INACTIVE** (not having the focus), **SET** (set with value), **UNSET** (not set with value),**CLICKABLE** (can be clicked upon),**VISIBLE** (exists and shown on the screen),**ABSENT** (may or may not exist and not shown on the screen),**HIGHLIGHTED** (marked with any highlighting color), **ON** (check marked), **OFF** (not check marked) etc.

1. **ObjectCurrentStateCheck (String ObjectName, String StateToCheck, Boolean [OutCheckStatus], String ExtraParam);**

This method checks to ensure if the given state of the given object is the current state. If the given state is verified as the current state, then the method returns **TRUE** in the output parameter, otherwise it returns **FALSE** in the output parameter.

**ObjectName** = Contains the GUI name of the object as represented in the pertinent object repository property file.

**StateToCheck** = Contains the key words representing the current state of the object to be asserted, i.e. **ENABLED** (can be operated on), **DISABLED** (can’t be operated on), **ACTIVE** (having the focus), **INACTIVE** (not having the focus), **SET** (set with value), **UNSET** (not set with value),**CLICKABLE** (can be clicked upon), **VISIBLE** (exists and shown on the screen), **ABSENT** (may or may not exist and not shown on the screen),**HIGHLIGHTED** (marked with any highlighting color), **ON** (check marked), **OFF** (not check marked) etc.

**[OutCheckStatus]** = Holds **TRUE** or **FALSE** as the value of the findings.

1. **ObjectVisibilityAssert (String ObjectName, Integer MaxTimeWait, String ExtraParam);**

This method asserts or confirms that the given web element is currently visible on the screen. The method keeps trying to assert the visibility state of the named web element until the given maximum wait time expires. As soon as the method is able to assert the visibility of the named web element within the given time limit, the method stops and returns to the calling routine.

**ObjectName** = Contains the GUI name of the object as represented in the pertinent object repository property file.

**MaxTimeWait** = Maximum time to wait up to for the existence check. Integer type, time in seconds, the **DEFAULT** defaults to **GblMaxWaitTime** predefined value. The **NULL** or 0 (zero) value indicates no wait time after the first attempt to verify object visibility.

1. **ObjectInvisibilityAssert (String ObjectName, Integer MaxTimeWait, String ExtraParam);**

This method asserts or confirms that the given web element is currently not visible on the screen. The method keeps trying to assert the invisibility state of the named web element until the given maximum wait time expires. As soon as the method is able to assert the invisibility of the named web element within the given time limit, the method stops and returns to the calling routine.

**ObjectName** = Contains the GUI name of the object as represented in the pertinent object repository property file.

**MaxTimeWait** = Maximum time to wait up to for the non-existence check. Integer type, time in seconds, The **DEFAULT** defaults to **GblMaxWaitTime** predefined value. The **NULL** or 0 (zero) value indicates no wait time after the first attempt to verify object invisibility.

1. **ObjectGroupActionSet (String DataSheetName, Integer RepeatLoopCount, String RowMarkingTags, String ExtraParam);**

This method allows performing of various types of actions on a group of web elements one after another as part of a single particular test step. This approach saves a lot of tedious repetition of the same method call over and over again in the Test Main data sheet and also makes the efforts fully data driven. If one single action fails, then the method sets the **GblStepPassFailStatus** global parameter to FAIL and returns FAIL as a whole.

**DataSheetName** = Indicates the name of the Excel data sheet containing the actual data value to be set in the named web elements via the **ObjectActionSet** or **WebTableCellActionSet** method calls only. The very first row of the data sheet contains the parameter names as the column headers and the names are fixed. The rest of the data rows contain the parameter values that are mapped to the pertinent parameters of the method before the method execution. The names of the data sheet columns are fixed to the following names: **RowMarkingFlag, ObjectName, ActionType, ValueToBeSet, ExtraActionIndicator, TestReportFlag, MiscComments.** The **RowMarkingFlag** column by default can be set to **“ROIFLAG”** marker indicating Rows Of Interest for execution. If this column is set to **“ROWSKIP”** or **“##”** marker, then the row is skipped from execution. Also any meaningful custom tag like **BASICINFO1, BASICINFO2, FINANINFO, …CREDITCARDINFO** and so on can be used to indicate to the Test Engine to selectively execute the named set or sets of rows only. The **“ROIFLAG:SCRNSHOT”** or **“BASICINFO1:SCRNSHOT”** tag captures the screen shot with default settings after executing the marked row with this tag. If this column is set to **“WEBTABLE”** tag, then it indicates that the action is intended to be performed on the given web table. In this case, the **ExtraActionIndicator** column must contain the formatted and delimited identifying string for the cell or the object within the cell on which the action is to be performed. The delimited string will look like this **"RI=1|CN=No/Yes|OT=ONOFFSWITCH"** where **RI** stands for Row Index, **CN** stands for Column Name and **OT** stands for Object Type. The other supported formatting indicators are **CI** for Column Index, **RS** for Reference Search String, **RC** for Reference Search Column Name, **II** for Instance Index (No.) and **PN** for Property Name. The combination of these indicators is necessary for the pertinent web table specific method in order to locate the web table cell or the cell object in any particular row of the table during runtime. The **“WEBTABLE:SCRNSHOT”** tag captures the screen shot with default settings after executing the marked row with this **WEBTABLE** tag. The screen shot file name suffix can also be supplied using the colon (“:”) delimiter, i.e. **“WEBTABLE:SCRNSHOT:FATCAChecklist”, “BASICINFO2:SCRNSHOT:RedColorCodedLabel”** and so on. Among many screen shot files, the meaningful file name suffix can be very useful to know what screen shot the file contains without having to open each file. The **MiscComments** column can be used to print pertinent information to the test report file if comes with the report message indicator prefix “RMI:”. The variable parameters can be used in the message enclosed in back quotes, i.e. `OutPropertyValue`, so that the actual parameter value can be replaced there. This column can also be used optionally for supplying meaningful string as the file name suffix for screen shot file names when the row is marked with the **SCRNSHOT** tag. In this case, the string must be prefixed with the “FSI:” (Filename Suffix Indicator) tag, i.e. “FSI:RedColorCodedLabels”. Among many screen shot files, the meaningful file name suffix can be very useful to know what screen shot the file contains without having to open each file. This column provides an additional option to supply the file name suffix for that purpose. If the **TestReportFlag** column is set to **YES** or **TRUE**, then the report message is formatted as per the following format before adding to the arrays.

For **TestResultReportArray** array (FAIL message always added to regardless of the **TestReportFlag** setting):

If **MiscComments** is null or does not contain the “RMI:” prefix, then

**[RowMarkingFlag**:**ObjectName**:\tThe setting of action **ActionType** and value **ValueToBeSet** PASS/FAIL.\n**]**

Else “RMI:” prefix is stripped off

**[RowMarkingFlag**:**ObjectName**:\t**MiscComments** + PASS/FAIL.\n**]**

For **TestDetailLogArray** array (Always added to regardless of the **TestReportFlag** setting):

If **MiscComments** is null or does not contain the “RMI:” prefix, then

**[**HH24:mm:ss:SSS **RowMarkingFlag**:**ObjectName**:\t**ObjectGroupActionSet**::\tThe setting of action **ActionType** and value **ValueToBeSet** PASS/FAIL.\n**]**

Else “RMI:” prefix is stripped off

**[**HH24:mm:ss:SSS **RowMarkingFlag**:**ObjectName**:\t**ObjectGroupActionSet**::\t**MiscComments** + PASS/FAIL.\n**]**

**Notes:** For reporting **WEBTABLE** specific information to the test report and log files, the **ObjectName** will be the trimmed off title text of the web table in init-cap format if available, else it will display the name of the output parameter containing the web table identity.

**RepeatLoopCount** = This outer loop count indicates exactly how many times the execution of the whole set of rows given in the named data sheet will be repeated as a set. This approach is meant for a group of objects each having the similar set of actions and verification requirements in the test case. This approach reduces a significant number of repeated method calls in the Test Main data sheet, thus making the data sheet concise and robust. The **DEFAULT** or **NULL** value defaults to 1.

**RowMarkingTags =** Indicates the set of rows to be executed by the use of reserved or custom tags in the **RowMarkingFlag** column of the data sheet, i.e. **“ROIFLAG”, ”ROISET1”, “ROISET2”, “BASICINFO”, “CCINFO”** and so on. The rest of the rows that are not marked with the same tag in the data sheet will be skipped in this case. This approach allows bundling similar test data in one data sheet instead of multiple data sheets and then using different sets of data in different test steps within the same test as necessary. Multiple tags representing multiple sets can also be supplied in this parameter using the “|” delimiter, i.e. “**ROISET1”|”BASICINFO”|”CCINFO**”. The key words **DEFAULT** or **NULL** indicate all rows to be considered for execution, unless marked with **“ROWSKIP”** tag or **“##”** marker.

The set indicating tag can be further delimited by colon “:” to indicate a subset within a set. For example, **“WEBTABLE:SET1”**, **“WEBTABLE:SET2”**, **“WEBTABLE:SET3”** and so on. In this case, if the parameter value is set to **“WEBTABLE:SET1”**|**“WEBTABLE:SET3”**, it indicates to execute the **SET1** and **SET3** of the **WEBTABLE** tag and skip the **SET2** of the same.

1. **ObjectGroupActionVerify (String DataSheetName, Integer RepeatLoopCount, String RowMarkingTags, String ExtraParam);**

This method allows retrieval of various types of property values for verification purposes for a group of web elements one after another as part of a single particular test step. This approach saves a lot of tedious repetition of the same method call over and over again in the Test Main data sheet and also makes the efforts fully data driven. If one single action/verification fails, then the method sets the **GblStepPassFailStatus** global parameter to FAIL and returns FAIL as a whole.

**DataSheetName** = Indicates the name of the Excel data sheet containing the GUI object repository object names whose given property values are to be obtained via the **ObjectPropertyValueGet** and **WebTableCellValueGet** method calls for the sole purpose of group value comparison via the **UtlDataValueCompare** method call. The very first row of the data sheet contains the parameter names as column headers and the names are fixed. The rest of the data rows contain the parameter values that are mapped to the pertinent parameters of the methods before the method execution. The names of the data sheet columns are fixed to the following names: **RowMarkingFlag, ObjectName, PropertyName, ValueToGetParam, ExtraPropertyIndicator, ComparisonOperator, ValueToCompare, ValueType, CaseSensitivity, TestReportFlag, MiscComments.** The **RowMarkingFlag** column by default can be set to **“ROIFLAG”** marker indicating Rows Of Interest for execution. If this column is set to **“ROWSKIP”** or **“##”** marker, then the row is skipped from execution. Also any meaningful custom tag like **BASICINFO1, BASICINFO2, FINANINFO, …CREDITCARDINFO** and so on can be used to indicate to the Test Engine to selectively execute the named set or sets of rows only. The **“ROIFLAG:SCRNSHOT”** or **“BASICINFO1:SCRNSHOT”** tag captures the screen shot with default settings after executing the marked row with this tag. If this column is set to **“WEBTABLE”** tag, then it indicates that the verification is intended to be performed on the given web table. In this case, the **ExtraPropertyIndicator** column must contain the formatted and delimited identifying string for the cell or the object within the cell on which the verification is to be performed. The delimited string will look like this **"RI=1|CN=No/Yes|OT=ONOFFSWITCH"** where **RI** stands for Row Index, **CN** stands for Column Name and **OT** stands for Object Type. The other supported formatting indicators are **CI** for Column Index, **RS** for Reference Search String, **RC** for Reference Search Column Name, **II** for Instance Index (No.) and **PN** for Property Name. The combination of these indicators is necessary for the pertinent web table specific method in order to locate the web table cell or the cell object in any particular row of the table during runtime.The **“WEBTABLE:SCRNSHOT”** tag captures the screen shot with default settings after executing the marked row with this **WEBTABLE** tag. The screen shot file name suffix can also be supplied using the colon (“:”) delimiter, i.e. **“WEBTABLE:SCRNSHOT:FATCAChecklist”, “BASICINFO2:SCRNSHOT:RedColorCodedLabel”** and so on. Among many screen shot files, the meaningful file name suffix can be very useful to know what screen shot the file contains without having to open each file. The **MiscComments** column can be used to print pertinent information to the test report file if comes with the report message indicator prefix “RMI:”. The variable parameters can be used in the message enclosed in back quotes, i.e. `OutPropertyValue`, so that the actual parameter value can be replaced there. This column can also be used optionally for supplying meaningful string as the file name suffix for screen shot file names when the row is marked with the **SCRNSHOT** tag. In this case, the string must be prefixed with the “FSI:” (Filename Suffix Indicator) tag, i.e. “FSI:RedColorCodedLabels”. Among many screen shot files, the meaningful file name suffix can be very useful to know what screen shot the file contains without having to open each file. This column provides an additional option to supply the file name suffix for that purpose. If the **TestReportFlag** column is set to **YES** or **TRUE**, then the report message is formatted as per the following format before adding to the arrays.

For **TestResultReportArray** array (FAIL message always added to regardless of the **TestReportFlag** setting):

If **MiscComments** is null or does not contain the “RMI:” prefix, then

**[RowMarkingFlag**:**ObjectName**:\tComparisonoffield value **PropertyOutValue** toexpected value **CompareValue** with operator **CompareType** PASS/FAIL.\n**]**

Else “RMI:” prefix is stripped off

**[RowMarkingFlag**:**ObjectName**:\t**MiscComments** + PASS/FAIL.\n**]**

For **TestDetailLogArray** array (Always added to regardless of the **TestReportFlag** setting):

If **MiscComments** is null or does not contain the “RMI:” prefix, then

**[**HH24:mm:ss:SSS **RowMarkingFlag**:**ObjectName**:\t**ObjectGroupActionVerify**::\tComparisonoffield value **PropertyOutValue** toexpected value **CompareValue** with operator **CompareType** PASS/FAIL.\n**]**

Else “RMI:” prefix is stripped off

**[**HH24:mm:ss:SSS **RowMarkingFlag**:**ObjectName**:\t**ObjectGroupActionVerify**::\t**MiscComments** + PASS/FAIL.\n**]**

**Notes:** For reporting **WEBTABLE** specific information to the test report and log files, the **ObjectName** will be the trimmed off title text of the web table in init-cap format if available, else it will display the name of the output parameter containing the web table identity.

**RepeatLoopCount** = This outer loop count indicates exactly how many times the execution of the whole set of rows given in the named data sheet will be repeated as a set. This approach is meant for a group of objects each having the similar set of actions and verification requirements in the test case. This approach reduces a significant number of repeated method calls in the Test Main data sheet, thus making the data sheet concise and robust. The **DEFAULT** or **NULL** value defaults to 1.

**RowMarkingTags =** Indicates the set of rows to be executed by the use of reserved or custom tags in the **RowMarkingFlag** column of the data sheet, i.e. **“ROIFLAG”, ”ROISET1”, “ROISET2”, “BASICINFO”, “CCINFO”** and so on. The rest of the rows that are not marked with the same tag in the data sheet will be skipped in this case. This approach allows bundling similar test data in one data sheet instead of multiple data sheets and then using different sets of data in different test steps within the same test as necessary. Multiple tags representing multiple sets can also be supplied in this parameter using the “|” delimiter, i.e. “**ROISET1”|”BASICINFO”|”CCINFO**”. The key words **DEFAULT** or **NULL** indicate all rows to be considered for execution, unless marked with **“ROWSKIP”** tag or **“##”** marker.

The set indicating tag can be further delimited by colon “:” to indicate a subset within a set. For example, **“WEBTABLE:SET1”**, **“WEBTABLE:SET2”**, **“WEBTABLE:SET3”** and so on. In this case, if the parameter value is set to **“WEBTABLE:SET1”**|**“WEBTABLE:SET3”**, it indicates to execute the **SET1** and **SET3** of the **WEBTABLE** tag and skip the **SET2** of the same.

1. **ObjectGroupActionRepeat (String DataSheetName, Integer RepeatLoopCount, String RowMarkingTags, String ExtraParam);**

This method allows bundling, parameterizing and executing a fixed set of multiple method calls for one web element and then repeat the same set of actions for any number of web elements during runtime one after another as part of a single particular test step. This approach saves a significant amount of tedious repetition of the same set of method calls over and over again in the Test Main data sheet. It also makes the efforts fully parameter driven with a new set of test data for each iteration. If one single method execution fails, then the method sets the **GblStepPassFailStatus** global parameter to FAIL and returns FAIL as a whole.

**DataSheetName** = Indicates the name of the Excel data sheet containing the subset of method call commands, just like in the TestMain data sheet in terms of format and functionality, that are to be executed repeatedly as a group in the same order inside the test for various action and verification purposes. This approach helps make the TestMain data sheet concise and more robust without having lengthy repeats. The very first row of the data sheet contains the parameter names and the names are fixed. The rest of the data rows contain the parameter values that are mapped to the pertinent parameters of the called methods before the method execution. The names of the data sheet columns are fixed to the following names: **RowMarkingFlag, MethodName, ParameterList, TestReportFlag, MiscComments.** The **RowMarkingFlag** column by default can be set to **“ROIFLAG”** marker indicating Rows Of Interest for execution. If this column is set to **“ROWSKIP”** or **“##”** marker, then the row is skipped from execution. The **“ROIFLAG:SCRNSHOT”** tag captures the screen shot with default settings after executing the marked row with this tag. The screen shot file name suffix can also be supplied using the colon (“:”) delimiter, i.e. **“ROIFLAG:SCRNSHOT:Customer”, “ROISET2:SCRNSHOT:RedColorCodedLabel”** and so on. Among many screen shot files, the meaningful file name suffix can be very useful to know what screen shot the file contains without having to open each file. The **MiscComments** column can be used to print pertinent information to the test report file if comes with the report message indicator prefix “RMI:”. The variable parameters can be used in the message enclosed in back quotes, i.e. `OutPropertyValue`, so that the actual parameter value can be replaced there. This column can also be used optionally for supplying meaningful string as the file name suffix for screen shot file names when the row is marked with the **SCRNSHOT** tag. In this case, the string must be prefixed with the “FSI:” (Filename Suffix Indicator) tag, i.e. “FSI:RedColorCodedLabels”. Among many screen shot files, the meaningful file name suffix can be very useful to know what screen shot the file contains without having to open each file. This column provides an additional option to supply the file name suffix for that purpose. If the **TestReportFlag** column is set to **YES** or **TRUE**, then the report message is formatted as per the following format before adding to the arrays.

For **TestResultReportArray** array (FAIL message always added to regardless of the **TestReportFlag** setting):

If **MiscComments** is null or does not contain the “RMI:” prefix, then

**[RowMarkingFlag**:**MethodName**:\tThe Method Return Code with parameter list + **ParameterList** + PASS/FAIL.\n**]**

Else “RMI:” prefix is stripped off

**[RowMarkingFlag**:**MethodName**:\t**MiscComments** + PASS/FAIL.\n**]**

For **TestDetailLogArray** array (Always added to regardless of the **TestReportFlag** setting):

If **MiscComments** is null or does not contain the “RMI:” prefix, then

**[**HH24:mm:ss:SSS **RowMarkingFlag**:**MethodName**:\t**ObjectGroupActionRepeat**::\tThe Method Return Code with parameter list + **ParameterList** + PASS/FAIL.\n**]**

Else “RMI:” prefix is stripped off

**[**HH24:mm:ss:SSS **RowMarkingFlag**:**MethodName**:\t**ObjectGroupActionRepeat**::\t**MiscComments** + PASS/FAIL.\n**]**

**RepeatLoopCount** = This outer loop count indicates exactly how many times the execution of the whole set of method calls given in the named data sheet will be repeated as a set. This approach is meant for a group of objects each having the similar set of actions and verification requirements in the test case. This approach reduces a significant number of repeated method calls in the Test Main data sheet, thus making the data sheet concise and robust. The **DEFAULT** or **NULL** value defaults to 1.

**RowMarkingTags =** Indicates the set of rows to be executed by the use of reserved or custom tags in the **RowMarkingFlag** column of the data sheet, i.e. **“ROIFLAG”, ”ROISET1”, “ROISET2”** and so on. The rest of the rows that are not marked with the same tag in the data sheet will be skipped in this case. This approach allows bundling similar test data in one data sheet instead of multiple data sheets and then using different sets of data in different test steps within the same test as necessary. Multiple tags representing multiple sets can also be supplied in this parameter using the “|” delimiter, i.e. “**ROIFLAG”|”ROISET1”|”ROISET2**”. The key words **DEFAULT** or **NULL** indicate all rows to be considered for execution, unless marked with **“ROWSKIP”** tag or **“##”** marker.

1. **ObjectGroupActionConditionEvaluate (String DataSheetName, Integer RepeatLoopCount, String RowMarkingTags, String ExtraParam);**

This method helps mimic the execution of a conditional statement like “If Ten Else Then” structure. As part of conditional decision making support (If Then Else Then) in the Framework during run time, this method checks the value of the current conditional parameter name previously set in the **GblCurrentCondParameter** global parameter by the prior call to the **UtlConditionalParameterSet** method. Every time this method is called, a new conditional statement is set. The value of this predefined parameter can only be a Boolean value, **TRUE** or **FALSE**. Based on the **TRUE/FALSE** value of this conditional parameter, this method performs the series of method executions as set forth in the given data sheet. The **RowMarkingFlag** column in the given data sheet is marked by **TRUE/FALSE** flag in order to indicate as to which set of the methods to be executed if the condition is **TRUE** and vice versa for the **FALSE** condition.

The given data sheet must be organized in such a way that the intended method call producing a **TRUE** or **FALSE** value will be executed first. The **RowMarkingFlag** column of the same row in the given data sheet is marked by **CONDITIONAL** tag in order to indicate conditional command and normal (not an exception due to system/detection error) **FAIL** return code will not be reported to the test report array only, unlike the usual case. Then the return value of this method is captured in a predefined return code or method output parameter. Then this returned value containing parameter is set as the current conditional parameter by the subsequent **UtlConditionalParameterSet** method call and the corresponding **RowMarkingFlag** column must be marked by the **CONDPARAM** tag. This in turn initiates the conditional statement. Now all subsequent commands that are marked with the **TRUE/FALSE** tag in the corresponding **RowMarkingFlag** column will be executed based on the **TRUE/FALSE** value of the current conditional parameter. Using the same approach, multiple or nested conditional statements can be evaluated and executed accordingly as set forth in the given data sheet. Based on the return code of the conditional set of commands execution, the method determines the final PASS/FAIL status to return and sets the **GblStepPassFailStatus** global parameter accordingly.

**DataSheetName** = Indicates the name of the Excel data sheet containing the subset of method call commands, just like in the TestMain data sheet in terms of format and functionality, that are to be executed repeatedly as a group in the same order inside the test for various conditional action and verification purposes. This approach helps make the TestMain data sheet concise and more robust without having lengthy repeats. The very first row of the data sheet contains the parameter names as column headers and the names are fixed. The rest of the data rows contain the parameter values that are mapped to the pertinent parameters of the called methods before the method execution. The names of the data sheet columns are fixed to the following names: **RowMarkingFlag, MethodName, ParameterList, TestReportFlag, MiscComments.** The **RowMarkingFlag** column by default can be set to **“ROIFLAG”** marker indicating Rows Of Interest for execution. If this column is set to **“ROWSKIP”** or **“##”** marker, then the row is skipped from execution. The **“ROIFLAG:SCRNSHOT”** tag captures the screen shot with default settings after executing the marked row with this tag. The screen shot file name suffix can also be supplied using the colon (“:”) delimiter, i.e. **“ROIFLAG:SCRNSHOT:Customer”, “ROISET2:SCRNSHOT:RedColorCodedLabel”** and so on. Among many screen shot files, the meaningful file name suffix can be very useful to know what screen shot the file contains without having to open each file. The **MiscComments** column can be used to print pertinent information to the test report file if comes with the report message indicator prefix “RMI:”. The variable parameters can be used in the message enclosed in back quotes, i.e. `OutPropertyValue`, so that the actual parameter value can be replaced there. This column can also be used optionally for supplying meaningful string as the file name suffix for screen shot file names when the row is marked with the **SCRNSHOT** tag. In this case, the string must be prefixed with the “FSI:” (Filename Suffix Indicator) tag, i.e. “FSI:RedColorCodedLabels”. Among many screen shot files, the meaningful file name suffix can be very useful to know what screen shot the file contains without having to open each file. This column provides an additional option to supply the file name suffix for that purpose. If the **TestReportFlag** column is set to **YES** or **TRUE**, then the report message is formatted as per the following format before adding to the arrays.

For **TestResultReportArray** array (**FAIL** message always added to regardless of the **TestReportFlag** setting. The normal **FAIL** message for the ones marked with **CONDITIONAL** flag will not be reported to this array.):

If **MiscComments** is null or does not contain the “RMI:” prefix, then

**[RowMarkingFlag:MethodName**:\tThe Method Return Code with parameter list + **ParameterList** + PASS/FAIL.\n**]**

Else “RMI:” prefix is stripped off

**[RowMarkingFlag:MethodName**:\t**MiscComments** + PASS/FAIL.\n**]**

For **TestDetailLogArray** array (Always added to regardless of the **TestReportFlag** setting):

If **MiscComments** is null or does not contain the “RMI:” prefix, then

**[**HH24:mm:ss:SSS **RowMarkingFlag:MethodName**:\t**ObjectGroupActionConditionEvaluate**::\tThe Method Return Code with parameter list + **ParameterList** + PASS/FAIL.\n**]**

Else “RMI:” prefix is stripped off

**[**HH24:mm:ss:SSS **RowMarkingFlag:MethodName**:\t**ObjectGroupActionConditionEvaluate**::\t**MiscComments** + PASS/FAIL.\n**]**

**RepeatLoopCount** = This outer loop count indicates exactly how many times the execution of the whole set of method calls given in the named data sheet will be repeated as a set. This approach is meant for a group of objects each having the similar set of actions and verification requirements in the test case. This approach reduces a significant number of repeated method calls in the Test Main data sheet, thus making the data sheet concise and robust. The **DEFAULT** or **NULL** value defaults to 1.

**RowMarkingTags =** Indicates the set of rows to be executed by the use of reserved or custom tags in the **RowMarkingFlag** column of the data sheet, i.e. **“ROIFLAG”, ”ROISET1”, “ROISET2”** and so on. The rest of the rows that are not marked with the same tag in the data sheet will be skipped in this case. This approach allows bundling similar test data in one data sheet instead of multiple data sheets and then using different sets of data in different test steps within the same test as necessary. Multiple tags representing multiple sets can also be supplied in this parameter using the “|” delimiter, i.e. “**ROIFLAG”|”ROISET1”|”ROISET2**”. The key words **DEFAULT** or **NULL** indicate all rows to be considered for execution, unless marked with **“ROWSKIP”** tag or **“##”** marker.

**Category 6: Non GUI Task Specific Utility Methods**

1. **UtlDataStringMerge (String PrimaryString, String MergerOperator, String StringToMerge, String [OutMergerString], String ExtraParam);**

This method performs the intended string merger. The given sub string is merged in to the given primary main string as per the criteria supplied in other parameters of the method.

**PrimaryString** = Contains the main string to be merged with the sub string. If a single sub string merger is intended somewhere inside the primary string, then the single or multiple merger positions must be marked with “Ӿ” (X marks the spot) indicator inside the primary string before passing it to this method. The keyword **IN** must be the **MergerOperator** parameter value in this case.

**MergerOperator** = **IB** (In the beginning of the string), **AE** (At the end of the string), **IN** (inside somewhere of the string marked by the merger indicator). If actual position is indicated in digit format, i.e.” 7”, then the merger starts exactly from the given positon in the primary string and the rest of the primary string is pushed back. But the given position can’t be greater than the size of the primary string, if so then it will fail.

**StringToMerge** = Contains the sub string to merge with the main string. Multiple substrings may be supplied in this parameter delimited by “Ӿ” character, i.e. for actual hardcoded values **“sub string 1”Ӿ”sub string 2”Ӿ”sub string 3”Ӿ”sub string 4”** or for variable parameters **InputVar1ӾInputVar2ӾInputVar3Ӿ”sub string 1”ӾInputVar4**. When multiple sub strings are supplied, the number of marked merger positions (“Ӿ”) in the primary string must match the number of supplied sub strings. Any mismatch will result into failure in this case. When a single sub string is supplied and intended to be merged in multiple positions in the main primary string, then each intended position must be individually marked with “Ӿ” marker inside the main primary string.

**[OutMergerString]** = Holds the resultant output string after the intended string merger.

1. **UtlDataValueCompare (String ValueParam1, String CompOperator, String ValueParam2, String CompValueType, Boolean CaseSensitivity, String ExtraParam);**

This method compares the two given data values as per the given data type, case sensitivity indicator and comparison operator.

**ValueParam1** = Indicates the primary value string to be used in the comparison. It is imperative to maintain the order of precedence of the two entities being compared. Based on which comparison operator is being used, the order of precedence will directly impact the outcome.

**CompOperator = EQ** (Is Equal To), **NE** (Is Not Equal To), **GT** (Is Greater Than), **LT** (Is Lesser Than), **GE** (Is Greater Or Equal To), **LE** (Is Lesser Or Equal To), **IN** (Matches anywhere in the string), **IB** (Matches in the beginning of the string), **AE** (Matches at the end of the string), **NI** (Matches nowhere in the string), **PM** (The parameter **ValueParam1** matches the Regular Expression pattern given in the parameter **ValueParam2**).

**ValueParam2** = Indicates the secondary value string to be used in the comparison. It is imperative to maintain the order of precedence of the two entities being compared. Each of the comparison operators above is associated with its definition enclosed in parenthesis and this definition phrase should be used to position the two entities before and after the comparison operator. For example, “5” **GT** “3” (5 Is Greater Than 3), “7” **LT** “8” (7 Is Lesser Than 8), “B” **IN** “ABCD” (“B” Matches anywhere in the string “ABCD”), “A” **NI** “EFGH” (“A” Matches nowhere in the string “EFGH”), “D” **AE** “ABCD” (“D” Matches at the end of the string “ABCD”), “E” **IB** “EFGH” (“E” Matches in the beginning of the string “EFGH”), “ABCD1234” **PM** “[A-Z].\*[0-9].\*” (“ABCD1234” matches the pattern “[A-Z].\*[0-9].\*”, Alpha-Numeric RegEx. Pattern matching).

**CompValueType =** Indicates whether to perform numeric data or string data type comparison. For numeric data type comparison, the values must be of type numeric only, i.e. integer, long, float, double and so on. For string comparison, the values can be of any valid data type. The keywords indicating the types are **NUM** (Numeric) and **STR** (String).

**CaseSensitivity =** Indicates whether to perform the string data type comparison in a case sensitive mode or not. The keywords indicating the modes are **TRUE or YES(DEFAULT), FALSE or NO.**

1. **UtlSyncTimeWait (Integer SecondsToWait, String ExtraParam);**

This method performs time synchronization by simply waiting out the given time in seconds.

**SecondsToWait** = Integer type, time in seconds, **DEFAULT** defaults to **GblMaxWaitTime** predefined value.

1. **UtlOnFailTestAbort (String ExtraParam);**

This method sets the Boolean global variable **GblTestStopper** to **TRUE** to indicate that the test run will be aborted when the next **FAIL** return code from any method call is encountered.

1. **UtlOnFailTestContinue (String ExtraParam);**

This method sets the Boolean global variable **GblTestStopper** to **FALSE** to indicate that the test run will continue even when the next **FAIL** return code from any method call is encountered.

1. **UtlParameterValueSet (String TestParamName, String ParamType, String ValueToBeSet, String ExtraParam);**

This method sets the given parameter with the given value during runtime.

**TestParamName** = Contains the name of the predefined parameter to be reset with the new value during run time.

**ParamType** = **LOCALPARAM(DEFAULT), GLOBALPARAM**. Indicates the type of the **TestParamName** parameter.

**ValueToBeSet =** Represents actual hard coded value in double quotes or the name of another variable parameter containing the actual value to be set.

1. **UtlConditionalParameterSet (String CondParamName, String ExtraParam);**

This method initiates the execution of conditional statement. As part of conditional decision making support (If Then Else Then) in the Framework during run time, this method sets the named parameter as the currently active conditional parameter. The **TRUE/FALSE** value of this parameter is used by the **ObjectGroupActionConditionEvaluate** method for deciding which set of commands to execute.

**CondParamName =** Contains the name of the predefined parameter to be set as the current conditional parameter for conditional decision making support. The value of this predefined parameter can only be either **TRUE** or **FALSE**. The name, not the value, of this conditional parameter is set to the **GblCurrentCondParameter** global parameter. That is why the name of the conditional parameter must be supplied within double quotes, i.e. “OutReturnCode”.

1. **UtlDateTimeStringFormat (String FormatOuputMask, String FormatInputMask, String InputDateTimeString, String [OutDateTimeString], String ExtraParam);**

This method formats the given date/time input string as per the given format mask.

**FormatOuputMask** = Contains the intended date/time or both format mask, i.e. “DD-MM-YYYY”, “HH24:mm:ss”, “DD:MM:YYYY HH24:mm:ss” etc. The date/time stamp can be used by using the **DATESTAMP**, **TIMESTAMP** and **DATETIMESTAMP** keywords. These keywords correspond to the global variables **GblDateFormat** (DDMMYYYY), **GblTimeFormat** (HHMMSS) and **GblDateTimeFormat** (DDMMYYYYHHMMSS) respectively. These global variables contain the current formats and always can be set to new valid ones whenever needed. This parameter is mandatory. The **DEFAULT** is **DATETIMESTAMP.**

**FormatInputMask** = Contains the format mask for the input date/time string so that the method knows how to parse the input string accordingly, i.e. “MM/DD/YYYY”, “hh2:mm:ss am/pm”, “DD-MON-YYYY HH24.mm.ss” etc. The **DEFAULT** value indicates the currently set system format.

**InputDateTimeString** = Contains the input string of the date/time or both that is to be formatted. Putting these key words here indicates the current system date/time, i.e. **SYSTEMDATE, SYSTEMTIME, SYSTEMDATETIME**. The **DEFAULT** is **SYSTEMDATETIME.**

**[OutDateTimeString]** = Holds the formatted date/time string as output.

**Category 7: Web Table Interaction Specific Methods**

1. **WebTableIdentitySet (String UniqueColumnNameOnPage, Integer ColumnIndex, String [OutIdentityString], String ExtraParam);**

This method dynamically sets the identity recognition string for the table that contains the given column name in the position of the given index. The column name must be unique on the web page and cannot match with any other column name of any other table on the same web page. The index, starting at 1, must correctly indicate the given column’s sequential position in the web table. This dynamic approach of obtaining the recognition string for the web table is very useful when there are numerous web tables to deal with during runtime without having to predefine each of the tables in the object repository. This approach significantly lessens the burden on the object repository.

**UniqueColumnNameOnPage** = Indicates any unique column name of the table for which the identity recognition string to be constructed dynamically. The column name must be unique on the web page and cannot match with any other column name of any other table on the same web page.

**ColumnIndex** = Indicates the given column’s exact serial position in the web table with index starting at 1.

**[OutIdentityString]** = Holds the dynamically constructed identity recognition string for the web table as output value. This string can be used during runtime for any subsequent reference to the same table.

1. **WebTableCellLocationGet (String WebTableObjName, String RefSearchText, Integer InstanceNo, String RefColName, String CellColName, String [OutCellRowId], String [OutCellColumnId], String ExtraParam);**

This method helps getting the location of any particular cell within the given web table by returning the corresponding row and column indices of the cell in the corresponding output parameters. The intended cell is located during runtime based on some search and filtering criteria supplied in other input parameters of the method. The found cell location is later used by other web table specific methods for performing any action on or property value verification of that particular web table cell.

**WebTableObjName** = Indicates the dynamically obtained GUI object identifying string via the prior call to the **WebTableIdentitySet** method for the given web table.

**RefSearchText** = Indicates the actual reference string to be searched for in the given web table in order to locate the intended cell somewhere in the same row.

**InstanceNo** = Indicates the number of occurrence of the actual reference string to be searched for in the given web table. If the given occurrence number of the string is not found, then the method fails. The value **DEFAULT** or 1 indicates the very first occurrence of the reference search string. The value **LAST** indicates the last occurrence of the reference search string.

**RefColName** = Indicates the name of the column that contains the reference search string. This serves as the filter to further narrow the search within the given column only. The keyword **ALL** indicates to search the entire table row by row and column by column without any filter. The keyword **DEFAULT** indicates to search for the reference string in the first column only.

**CellColName** = Indicates the name of the column containing the intended cell to be located based on the reference search string. This column name is mandatory and must be supplied.

**[OutCellRowId]** = Holds the output numeric row id of the intended cell when the method call returns to the calling routine.

**[OutCellColumnId]** = Holds the output numeric column id of the intended cell when the method call returns to the calling routine.

1. **WebTableCellActionSet (String WebTableObjName, Integer CellRowId, Variant CellColumnIdentity, String CellObjectType, String CellActionType, String ValueToBeSet, String ExtraParam);**

This method performs the given action on the given web table cell or the given object within the cell which is identified by the supplied row and column identity of the cell.

**WebTableObjName** = Indicates the dynamically obtained GUI object identifying string via the prior call to the **WebTableIdentitySet** method for the given web table.

**CellRowId** = Indicates the data grid row id of the cell on which the action to be performed. The row indicated by zero(0) index is the very first row containing the column headers. For column sorting purpose, the value of this parameter must be 0 (zero).

**CellColumnIdentity** = Indicates the data grid column id or the column header name of the cell on which the action to be performed. The column indicated by index 1 is the very first column of the column headers. In some special cases, there may not be a column header in the very first column and this type of column is indicated by zero(0) index.

**CellObjectType** = Indicates type of the object contained in the cell. Putting **DATACELL** or **DEFAULT** keyword in this parameter indicates only the textual value contained in the cell.

**CellActionType** = Indicates the type of the action to be performed on the cell, i.e. **LCLICK, DLCLICK** and so on. For column sorting purpose, the value of this parameter must be **COLUMNSORT**. Putting **DEFAULT** keyword in this parameter indicates to perform the predefined default action of the given object type contained in the cell.

**ValueToBeSet** = Indicates the actual value to be set in the object contained in the cell. The key word **NULL** indicates no value to be set. For column sorting purpose, the value of this parameter must be set to the keyword **ASCN** (ascending) or **DSCN** (descending).

1. **WebTableCellValueGet (String WebTableObjName, Variant CellRowIdentity, Variant CellColumnIdentity, String CellObjectType, String PropertyName, String [OutCellValue], String ExtraParam);**

This method retrieves the given property value of the given web table cell or the given object within the cell which is identified by the supplied row and column identity of the cell.

**WebTableObjName** = Indicates the dynamically obtained GUI object identifying string via the prior call to the **WebTableIdentitySet** method for the given web table.

**CellRowIdentity** = Indicates the data grid row id of the cell from which the value to be obtained. In case of the special web table with only two columns having only field (key) and value pair, the field name can be supplied as the row name in this parameter. In this case, the column id must be 1 to indicate the next cell from where the field value should be fetched.

**CellColumnIdentity** = Indicates the data grid column id or the column header name of the cell from which the value to be obtained. In case of the special web table with only two columns having only field (key) and value pair, the field name can be supplied as the row name in the previous parameter. In this case, the column id must be 1 to indicate the next cell from where the field value should be fetched.

**CellObjectType** = Indicates type of the object contained in the cell. Putting **DATACELL** or **DEFAULT** keyword in this parameter indicates only the textual value contained in the cell.

**PropertyName** = Indicates the name of the particular property of the object whose value to be obtained. Putting **DEFAULT** keyword in this parameter indicates to fetch the predefined default value of the given object type contained in the cell.

**[OutCellValue]** = Holds the output property value of the given object when the method call returns to the calling routine.

1. **WebTablePaginatorSet (String WebTableObjName, Variant PageNumOrNavDirection, Integer NavRepeatCount, String [OutCurrPageValue], String ExtraParam);**

This method performs the page navigation back and forth within the given web table as per the criteria supplied in other parameters of the method.

**WebTableObjName** = Indicates the dynamically obtained GUI object identifying string via the prior call to the **WebTableIdentitySet** method for the given web table.

**PageNumOrNavDirection =** Indicates the actual page number or one of the following navigation directions: **GONEXT, GOPREV, GOFIRST, GOLAST** to go to by clicking on the paginator buttons. If the page number is given, the method will perform the necessary navigation to reach to the given page. If the given page number does not exist, the method will fail.

**NavRepeatCount** = Indicates how many times the given navigation direction button will be clicked upon. This parameter is irrelevant when the actual page number is given in the previous parameter.

**[OutCurrPageValue]** = Holds the active page number that is currently in display after the method call returns to the calling routine.

1. **WebTableColSortOrderVerify (String WebTableObjName, String ColumnName, String SortOrderType, String ValueType, String CaseSensitivity, String ExtraParam);**

This method performs the intended column sorting in the given sort order on the given column of the given web table.

**WebTableObjName** = Indicates the dynamically obtained GUI object identifying string via the prior call to the **WebTableIdentitySet** method for the given web table.

**ColumnName =** The column header of the given web table column to be verified for the given sort order.

**SortOrderType = ASCN** (ascending)**, DSCN** (descending)

**ValueType = NUM**(Numeric),**STR**(String)

**CaseSensitivity = TRUE or YES(DEFAULT), FALSE or NO**

1. **WebTableDataValidate (String WebTableObjName, String DataSheetName, String ExtraParam);**

This method validates the data value of the given web table via comparison to the expected baseline data saved in the data sheet of the test data Excel file. The actual web table data value and the expected baseline data value must be in the same order in terms of row and column index. Otherwise there will be mismatch during data validation and the method will return FAIL status. If one single action/verification fails, then the method sets the **GblStepPassFailStatus** global parameter to FAIL and returns FAIL as a whole.

**WebTableObjName** = Indicates the dynamically obtained GUI object identifying string via the prior call to the **WebTableIdentitySet** method for the web table containing the data to be validated.

**DataSheetName** = Indicates the name of the Excel data sheet containing the expected data value to validate against the data value obtained from the given web table during runtime. The very first row of the data sheet contains the exact case sensitive names of the columns of the web table in the same order as they appear in the web table. The data of each column of each row is validated in the same order as it appears in the given web table. The titles of the fixed data sheet column headers are the following names: **RowMarkingFlag, TestReportFlag, MiscComments.** The **RowMarkingFlag** column by default can be set to **“ROIFLAG”** marker indicating Rows Of Interest for execution. If this column is set to **“ROWSKIP”** or **“##”** marker, then the row is skipped from execution. The **“ROIFLAG:SCRNSHOT”** tag captures the screen shot with default settings after executing the marked row with this tag. The exact case sensitive names of the web table columns are given in the data sheet column header row within brackets in the same order as they appear in the web table, i.e. **[Web Table Col 1], [Web Table Col 2]…[Web Table Col N]**. If any web table column contains a web element object within the cell instead of simple text value and the verification of a particular property value is to be performed on that object, then the web table column name must be suffixed with extra delimited object identifying indicators within bracket in the data sheet column header. For example, the **“No/Yes”** column in the FATCA checklist table contains a web element of **ONOFFSWITCH** object type. In order to perform the verification of the current state of this object, the column name has to be suffixed in the following format in the data sheet column header row, i.e. **[No/Yes|OT=ONOFFSWITCH|PN=DEFAULT]**, where **OT** stands for Object Type and **PN** stands for Property Name. The **MiscComments** column can be used to print pertinent information to the test report file if comes with the report message indicator prefix “RMI:”. The variable parameters can be used in the message enclosed in back quotes, i.e. `OutPropertyValue`, so that the actual parameter value can be replaced there. If the **TestReportFlag** column is set to **YES** or **TRUE**, then the report message is formatted as per the following format before adding to the arrays.

For **TestResultReportArray** array (FAIL message always added to regardless of the **TestReportFlag** setting):

If **MiscComments** is null or does not contain the “RMI:” prefix, then

**[RowMarkingFlag**:**TableNameString**:\tColumn Headers: [ColumnName1]\t[ColumnName2]\t[ColumnName3]… \t[ColumnNameN]\n**]**

**[RowMarkingFlag**:**TableNameString**:\tRow N: [DS]ColumnValue1 : [WT]ColumnValue1\t[DS]ColumnValue2 : [WT]ColumnValue2\t …\t[DS]ColumnValueN : [WT]ColumnValueN Data Validation PASS/FAIL.\n**]**

Else “RMI:” prefix is stripped off

**[RowMarkingFlag**:**TableNameString**:\tColumn Headers: [ColumnName1]\t[ColumnName2]\t[ColumnName3]… \t[ColumnNameN]\n**]**

**[RowMarkingFlag**:**TableNameString**:\t**MiscComments** + PASS/FAIL.\n**]**

For **TestDetailLogArray** array (Always added to regardless of the **TestReportFlag** setting):

If **MiscComments** is null or does not contain the “RMI:” prefix, then

**[**HH24:mm:ss:SSS **RowMarkingFlag**:**TableNameString**:\t**WebTableDataValidate**:: \tColumn Headers: [ColumnName1]\t[ColumnName2]\t[ColumnName3]… \t[ColumnNameN]\n**]**

**[**HH24:mm:ss:SSS **RowMarkingFlag**:**TableNameString**:\t**WebTableDataValidate**::\tRow N: [DS]ColumnValue1 : [WT]ColumnValue1\t[DS]ColumnValue2 : [WT]ColumnValue2\t …\t[DS]ColumnValueN : [WT]ColumnValueN Data Validation PASS/FAIL.\n**]**

Else “RMI:” prefix is stripped off

**[**HH24:mm:ss:SSS **RowMarkingFlag**:**TableNameString**:\t**WebTableDataValidate**:: \tColumn Headers: [ColumnName1]\t[ColumnName2]\t[ColumnName3]… \t[ColumnNameN]\n**]**

**[**HH24:mm:ss:SSS **RowMarkingFlag**:**TableNameString**:\t**WebTableDataValidate**::\t**MiscComments** + PASS/FAIL.\n**]**

**Notes:** For reporting to the test report and log files, the **TableNameString** will be the trimmed off title text of the web table in init-cap format if available. Else it will display the table name string in the data sheet as comment if suppliedor the name of the output parameter containing the web table identity.