LibreOffice RefCard

LibreOffice Basic
Events

v. 1.10 – 12/02/2018

Inter.

Written using LibreOffice v. 5.3.3 – Platform: All

Events

LibreOffice objects know of various event types that can be fired in various situations. You may intercept these events and process accordingly.

Objets That Provide Events

- The application
- Documents
- Forms and controls
- Basic dialogues and controls
- Document objects:
  - OLE objects
  - Images
  - Frames
  - AutoTexts
  - ImageMaps
  - Hyperlinks

Associating An Event To A Macro

Principle

1. **Create the macro** to be executed.
   Follow this template:
   ```
   Sub MacroName()
   End Sub
   ```
   ☆ Name the macro according to the object, action and event type.
   example : `Sub OnButtonOKClick()`
   The Sub may bear a parameter. See below.
2. **Select the object** that carries the event to intercept (list above).
3. Go to the object settings (method varies depending on the object).
4. **Select the event** to intercept (see the lists in this refcard).
5. **point to the macro** to be executed when the event fires.

Getting Information About The Event Fired

The Sub may read the input parameter in order to get more information about the event :
```
Sub EventResponse(ByRef Event As Object)
End Sub
```
The `Event` object structure and properties depend upon the type of event that fires the Sub (see below).
Frequent situations for Basic dialogues controls:

To get to	**Read**
Calling **control** object	`Event.Source`
Control **model** object	`Event.Source.Model`
Dialogue object, control owner	`Event.Source.Context`

Event Categories And Properties

Dialogues provide four event categories (col. **Cat** in the last table):

Mouse	**M**	Events fired by actions using the mouse (ex: movements or clicks).
Keyboard	**K**	Events fired par keypresses.
Focus	**F**	Events fired when focus changes.
Specific	**S**	Events linked to some controls.

☠ The constants listed below are **case-sensitive**.

Mouse Events

Coordinates are in pixels, measured from the top-left control angle.

See the `com.sun.star.awt.MouseEvent` structure

`Buttons` (short)	The pressed button (const. `com.sun.star.awt.MouseButton`).
`X` (long) and `Y` (long)	X (resp. Y) coordinate of the mouse pointer.
`ClickCount` (long)	Number of clicks associated with the mouse event. When LibreOffice can respond fast enough, `ClickCount` is `1` for a double-click, because one single event is fired.
`PopupTrigger` (boolean)	`True` if contextual menu.

Constants defined in `com.sun.star.awt.MouseButton`

`LEFT`	Left button.	`RIGHT`	Right button	`MIDDLE`	Middle button.

☠ The VBA **Click** and **Doubleclick** events are not supported by LibO Basic. You may use the LibO Basic event **Mouse button released** as a replacement for a **Click** event and mimic the **Doubleclick** event by modifying the application logic.

Keyboard Events

Keyboard events are associated with logical key actions rather than to physical actions.

☞ Key combination = one single event.

☞ A unique action on a modification key (dead key, ex `SHIFT` or `ALT`) doesn't create an independent event.

The keyboard `Event` object holds the properties

`KeyCode` (short)	The keypressed code (`com.sun.star.awt.Key.XXX`). Modification keys (**Shift**, **Ctrl** and **Alt**) do not change that code.
`KeyChar` (String)	The entered character (takes modification keys into account).
`KeyFunc` (Integer)	The key functionality, as the constant in `com.sun.star.awt.KeyFunction.XXXX`
`Modifiers` (Integer)	Specifiies whether a modification was pressed, see constants `com.sun.star.awt.KeyModifier.XXX`.

Constants defined in `com.sun.star.awt.Key.XXX` (excerpts)

`NUM0` à `NUM9`	Numbers	`RETURN`	**Enter**	`POINT`	.
`A` à `Z`	Letters	`ESCAPE`	**Esc**	`COMMA`	,
`F1` à `F26`	Function keys	`TAB`	**Tab**	`LESS`	<
`UP`	Up	`BACKSPACE`	⌫	`GREATER`	>
`DOWN`	Down	`SPACE`	**Space**	`EQUAL`	=
`LEFT`	Left	`INSERT`	**Ins**	`CUT`	**Ctrl** + **X**
`RIGHT`	Right	`DELETE`	**Del**	`COPY`	**Ctrl** + **C**
`HOME`	Home	`ADD`	**+**	`PASTE`	**Ctrl** + **V**
`END`	End	`SUBTRACT`	**-**	`HELP`	**F1**
`PAGEUP`	Page up	`MULTIPLY`	*	`MENU`	⊞
`PAGEDOWN`	Page down	`DIVIDE`	/	`CONTEXTMENU`	☰

☞ These codes identify physical keys.

Constants in `com.sun.star.awt.KeyFunc.XXX`

`DONTKNOW`	None	`CUT`	Cut	`CLOSE`	Close
`NEW`	New	`COPY`	Copy	`QUIT`	Quit
`OPEN`	Open	`PASTE`	Paste	`PROPERTIES`	Properties
`SAVE`	Save	`UNDO`	Undo	`FIND`	Find
`SAVEAS`	Save as	`REDO`	Redo	`FINDBACKWARD`	Find backwards
`PRINT`	Print	`REPEAT`	Repeat	`FRONT`	To front

Constants in `com.sun.star.awt.KeyModifier.XXX` (additive with +/And)

`MOD1`	**Ctrl**	`MOD2`	**Alt**	`SHIFT`	**Maj**

Focus Events

The focus `Event` objects provide the following properties

`FocusFlags` (short)	The focus change reason. See the `com.sun.star.awt.FocusChangeReason` constants.
`NextFocus` (Object)	The object that gets the focus (only for the **On focus lost**) event.
`Temporary` (Boolean)	True if the focus is temporarily lost.

Constants as defined in `com.sun.star.awt.FocusChangeReason`

`TAB`	Tab has been pressed.	`BACKWARD`	Former control.
`CURSOR`	A direction key was pressed.	`AROUND`	From last to first control (forward) or from the first to the last (backward).
`MNEMONIC`	A shortcut key was pressed.	`UNIQUEMNEMONIC`	A shortcut key to a unique control was pressed.
`FORWARD`	Next control.		

Specific Events

Some events (ex : **When initiating**) can be fired by actions on some controls (ex : radio buttons). No action is executed to know whether the control status has actually changed. Avoid such "blind events" by recording the control former value in a global variable, then check whether the value was changed when the event is fired.

Properties for the `Item Status Changed` event:

`Selected` (long)	The currently selected entry.
`Highlighted` (long)	The currently highlighted entry.
`ItemId` (long)	The entry ID.

Document Events

Propriétés de l'événement entrant (voir chapitre suivant) :

`EventName` (string)	Le nom de l'événement.
`Source` (object)	Le document à l'origine de l'événement.
`ViewController` (object)	Le contrôleur d'affichage concerné, sinon `Null`.
`Supplement` (variant)	Des infos complémentaires, sinon `Empty`.

Document Or Application Events

Tools > Customize, **Events** tab.
Available events:

Event	The assigned macro is executed…
Start application	After the application start.
Close application	Before the application closes.
Document created	After **File > New**.
New document	After creating the document from a template.
Document loading finished	After the document was reloaded.
Open document	After **File > Open**.
Document is going to be closed	Before the document is closed.
Document closed	After the document has been closed. ☞ Note that the **Document closed** event may also be fired on document saving, just before the document is closed.
View created	After the document view has been created.
View is going to be closed	Before the document view is closed.
View closed	After the document view has been closed.
Activate document	After display of the document in the foreground.
Deactivate document	After display of another document in the foreground.
Save document	Before **File > Save**, if the document name is specified. ☞ See **Document closed**.
Document has been saved	After **File > Save**, if the document name is specified.
Saving of document failed	After the document saving has failed.
Save document as	Before **File > Save as** (or **File > Save** if the document name is not specified).
Document has been saved as	After **File > Save as** (or **File > Save** if the document name is not specified).
'Save as' has failed	After a **Save as** error.
Storing or exporting copy of document	
Document copy has been created	After the creation of the copy.
Creating of document copy failed	After error on copy creation.
Print document	After the **Print** dialog closes, but before actual printing.
'Modified' status changed	After the modification status has changed.
Document title changed	After the document title has changed.
Printing of form letters started	After the **Print** dialog closes, but before actual printing.
Printing of form letters finished	After mail merge printing.
Merging of form fields started	
Merging of form fields finished	
Changing the page count	When the page number changes.

☞ **View** events are fired when the document display changes: Preview mode or New Window mode.

Document Events Sequences

Opening An Existing Document (All Methods)
Open document > View created

Closing The Active Document (All Methods)
View is going to be closed > Document is going to be closed > View closed > Document closed

Creating A Document From A Template
New document > View created

Interacting With Document Objects

Object properties, then misc.: **Macro** tab, **Macro** button, **Events** button, etc.

Event	Fired when	OLE object	Image	Frame	AutoText	ImageMap	Hyperlink
Object clicked	The object is selected.	●	●	●			
Mouse hover	The mouse is moving above the object.	●	●	●		●	●
Hyperlink fired	The hyperlink assigned to the object is activated.	●	●	●			●
Mouse quit	The mouse pointer is moved outside of the object.	●	●	●		●	●
Image loading ended	The image loading is finished.		●				
Image loaded stopped	The images loading was interrupted by the user (ex: on page download).		●				
Image loading error	The image load failed (ex. image not found).		●				
Alphanumerical characters entered	Some text was entered using the keyboard.			●			
Non-alphanumerical characters entered	Non-printable chars were entered (tab, enter, etc.).			●			
Frame sized changed	The frame size was changed using the mouse.			●			
Frame moved	The frame was moved using the mouse.			●			
Before autotext insertion	Before the text block insertion.				●		
After autotext insertion	After the text block insertion.				●		

Calc Sheets Events

Sheet > Sheet Events (or right-click the sheet tab then **Sheet events**)

Event	The assigned macro is executed…
Activate document	After the document was displayed in the foreground.
Deactivate document	After another document was displayed in the foreground.
Selection changed	After the selection was changed.
Double click	After double-clicking a cell.
Right click	After right-clicking a range.
Formulas calculated	After formulae recalculation.
Content changed	After a cell contents has been changed.

Form Events

Control properties, **Events** tab.

Common Non-Database Events

Cat	Event	The assigned macro is executed…
F	**When receiving focus**	When the control receives the focus.
F	**When losing focus**	When a control loses the focus.
K	**Key pressed**	When a key is pressed on the focused control.
K	**Key released**	When a key is released on the focused control.
M	**Mouse inside**	When the mouse pointer is inside control boundaries.
M	**Mouse moved while key pressed**	When the mouse is moved while a key is pressed. Ex: pressing a key during a drag and drop sets the drag/drop mode (move or copy).
M	**Mouse moved**	While the mouse pointer moves over the control.
M	**Mouse button pressed**	When a mouse button is pressed while the pointer is over the control. ☞ The **Mouse button pressed** event is also used to notify of **context menu** calls. Then, the event `PopupTrigger` property is `TRUE`. On a right-click call, the event is fired twice: (1) on the context menu call and (2) for the click itself. If you're only interested in the click event, ignore calls where `PopupTrigger` is `TRUE`.
M	**Mouse button released**	When a mouse button is released while the pointer is over the control.
M	**Mouse outside**	While the mouse pointer is outside control boundaries.

Dialog Boxes Events

Some dialog boxes controls provide the supplemental events:

Cat	Event	The assigned macro is executed…
	Execute action	When the action starts. Ex: if the form has a **Send** button, the sending process is the action to take. ☞ Use for button click responses.
KM	**Item status changed**	When the control state has changed.
M	**While adjusting**	When moving a scroll bar or a spinbutton.

Database Events Only

Event	The assigned macro is executed…
After record action	After the current record was modified.
After record change	Just after the current record pointer was changed.
Before record action	Before the current record is modified. Allows to ask for a confirmation.
Before submitting	Before the form data is sent.
Before update	Before the modified control contents is written to the database. Ex. Stop this action by returning `FALSE`.
After update	After the modified control contents has been written to the database.
Prior to reset	Before a form is reset. The linked macro can, for example, prevent this action by returning `FALSE`. A form is reset if one of the following conditions is met: 1. The user presses an (HTML) button that is defined as a reset button. 2. A new and empty record is created in a form that is linked to a data source. For example, in the last record, the **Next Record** button may be pressed.
After resetting	After a form has been reset.
Before record change	Before the current record pointer is changed. Ex. To stop this action by returning `FALSE`.
Before unloading	Before the form is unloaded (separated from the database).
Before reloading	Before the form is reloaded. The data contents has not been updated yet.
Confirm deletion	As soon as the data have been suppressed from the form. Ex. Ask for a confirmation.
When loading	After the form was loaded.
When unloading	Just after the form was unloaded (separated from the database).
When reloading	Just after the form has been reloaded. The data have been already updated.
Fill parameters	When the to be loaded form needs parameters (SQL). This event is fired whenever a parameter can't be provided. For example, the form data source might be the SQL command: `SELECT * FROM address WHERE name=:name` where `:name` is a parameter that must be filled out when loading. The parameter is automatically filled out from the parent form if possible. If the parameter cannot be filled out, this event is called and a linked macro can fill out the parameter.
An error happened	Whenever an error occurs when accessing the database. ☞ Applies to forms, listboxes and comboboxes.

Credits

Author : Jean-François Nifenecker – jean-francois.nifenecker@laposte.net

We are like dwarves perched on the shoulders of giants, and thus we are able to see more and farther than the latter. And this is not at all because of the acuteness of our sight or the stature of our body, but because we are carried aloft and elevated by the magnitude of the giants (Bernard de Chartres [attr.])

History

Version	Date	Comments
1.01	11/09/2018	Minor updates
1.10	02/12/2018	Rework and complements