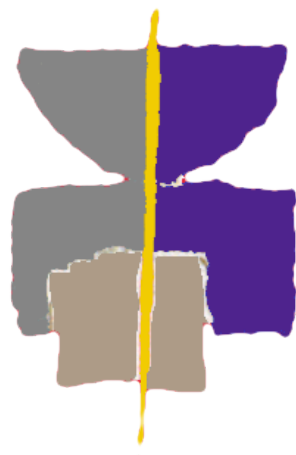


# ogTools Generics



## Component's user data sheet

- ◆ colors
  - ~ 4D and MD spaces
- ◆ io
  - ~ alert, confirm, more, notification, progress
- ◆ json
  - ~ editor for ogTools parameters
- ◆ picker
  - ~ date, date month, date week, day, period, search, time, year
- ◆ rc
  - ~ record choice manager
- ◆ window
  - ~ window placement manager



from v16

| Version | Date      | Rédacteur | Commentaires              |
|---------|-----------|-----------|---------------------------|
| Initial | 1/12/2018 | OG        | first version of generics |



|                                     |          |
|-------------------------------------|----------|
| <b>Overview</b>                     | <b>4</b> |
| <b>Objects options</b>              | <b>5</b> |
| <b>Methods</b>                      | <b>6</b> |
| Component                           | 6        |
| <i>wog_init</i>                     | 6        |
| <i>wog_setSerial</i>                | 6        |
| General                             | 7        |
| <i>wog__getModules</i>              | 7        |
| General parameters                  | 8        |
| <i>wog__getGeneral</i>              | 8        |
| <i>wog__get_colors_RWAL</i>         | 8        |
| <i>wog__get_current_year</i>        | 8        |
| <i>wog__get_fiscal_month</i>        | 9        |
| <i>wog__set_colors_RWAL</i>         | 9        |
| <i>wog__set_current_year</i>        | 9        |
| <i>wog__set_fiscal_month</i>        | 10       |
| Overload objects                    | 11       |
| <i>wog__ob_overload</i>             | 11       |
| <i>wog__ob_overload_exist</i>       | 11       |
| <i>wog__ob_overloads</i>            | 11       |
| <i>wog__ob_overloads_exist</i>      | 12       |
| <i>wog__ob_update_exists</i>        | 12       |
| Overload options and general        | 13       |
| <i>wog__setOptions</i>              | 13       |
| <i>wog__setOptions_general</i>      | 13       |
| Instances: data and options getters | 14       |
| <i>wog_getValue</i>                 | 15       |
| <i>wog_getValues</i>                | 15       |
| <i>wog_getValue_date</i>            | 15       |
| <i>wog_getValue_dates</i>           | 16       |
| <i>wog_getValue_ob</i>              | 16       |
| <i>wog_getValue_txt</i>             | 16       |
| <i>wog_getOptions</i>               | 17       |
| <i>wog_getRWAL</i>                  | 17       |



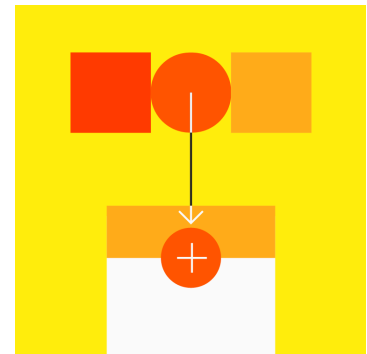
|  |           |
|--|-----------|
| <i>wog_getEditable</i>                             | 17        |
| Instances: data and options setters                | 18        |
| <i>wog_setValue</i>                                | 18        |
| <i>wog_setValue_date</i>                           | 18        |
| <i>wog_setValue_dates</i>                          | 19        |
| <i>wog_setValue_ob</i>                             | 19        |
| <i>wog_setValue_txt</i>                            | 19        |
| <i>wog_setOptions</i>                              | 19        |
| <i>wog_setRWAL</i>                                 | 20        |
| <i>wog_setEditable</i>                             | 20        |
| <i>wog_setValue_ptr</i>                            | 21        |
| <i>wog_setValues_ptr</i>                           | 21        |
| Instances: miscellaneous                           | 22        |
| <i>wog_resize</i>                                  | 22        |
| <i>wog_redraw</i>                                  | 22        |
| <i>wog_calculateRWAL</i>                           | 23        |
| <i>wog_calculate_newModif</i>                      | 23        |
| <i>wog_calculate_newModif</i>                      | 23        |
| <i>wog_setRWAL_fromField</i>                       | 24        |
| <b>Examples and usage</b>                          | <b>25</b> |
| <i>Method in the widget (single value, color)</i>  | 25        |
| <i>Method in the widget (single value, date)</i>   | 25        |
| <i>Method in the widget (double values, dates)</i> | 26        |



# Overview

## Introduction

The ogTools component is designed in an extremely regular and generic way. In this sense, most methods can be used for all widgets and pickers : getters, setters are the same. Here you will find all the generic methods for using this component.



## Objects

The use of objects allows the use of plenty parameters for widgets, pickers and managers.

- Widget: the basic flat graphical module, and the last end point. Example a calendar.
- Picker: an object which will open a form with a widget for modification. Example a date picker.
- Manager : some functionalities without ant interface. Example window manager.

In addition we introduce the notion, like in java, of « overload » which provides a very powerful way to change the parameters. This object given technology by 4D makes possible to design a totally generic component, with no to use of any process variable, for a benefice in memory.

- Each widget and picker has a general interprocess parameter object. This interprocess object is of course modifiable, either directly or by overload.
- Each created instance copies and works with a new duplicate set of settings.
- The instance object is obviously modifiable to have as instances as different behaviors.
- It is possible to send a partial object to the instance, and by the « overload » mechanism, the settings will only be updated with the partial object sent, others remained unchanged.

## Layout

The widgets and pickers are all drawable in the form according to your wishes. The content will be fully resized to the area you have chosen for your instance.

- Choice of size according to your needs
- Multi-instance



# Objects options

## Description

The component use two interprocess objects, one for general properties, one for all widgets and pickers options, regrouped by modules.

| Access to      | Method          | Contents                                      |
|----------------|-----------------|---|
| <b>general</b> | wog__getGeneral | current_year, fiscal_month, rwal[4]           |
| <b>modules</b> | wog__getModules | version, colors, io, json, picker, rc, window |

Modules contains all modules, and then all options (widgets and pickers), then properties. Some options properties are common to almost all options. They are listed here:

| Property             | Type    | Contents  |
|----------------------|---------|---|
| <b>is_autoAccept</b> | boolean | For a form, when closing it with the close button, or in case of popup window clicking outside, the behavior is like clicking on « ok » button. Else, you will be prompted with an « save - don't save - cancel » dialog. |
| <b>is_okCancel</b>   | boolean | Display « Ok » and « Cancel » button.   |
| <b>is_onMouseUp</b>  | boolean | For widget, the change to data is effective only on mouse up. Else, you will get a continuous update while moving elements on widgets.  |
| <b>ptr_external</b>  | pointer | Pointer to data. If not nil, this will be used in conjunction with the internal stored data.  |
| <b>ptr_external1</b> | pointer | Pointer to data1 in case of double data (dates, period...)  |
| <b>state_rwal</b>    | longint | version, colors, io, json, picker, rc, window   |
| <b>window_type</b>   | longint | Window type used for form opened in pickers. 0 Plain form window - 1 Pop up form window - 2 Sheet form window - 3 Palette form window   |



# Methods

## Component

### wog\_init

| wog_init                     |  |  |  |
|------------------------------|--|--|--|
| Do not require any parameter |  |  |  |
|                              |  |  |  |

### Description

The wog\_init command initialize the og\_tools component. You have to call this method before using any of the other methods of ogTools.

### wog\_setSerial

| wog_setSerial (serial) -> isOk |         |   |   |
|--------------------------------|---------|---|---|
| Parameter                      | Type    |   | Description                                 |
| serial                         | txt     | → | Serial number of your ogTools component     |
| isOk                           | boolean | ↔ | True if given serial is ok, False otherwise |

### Description

The wog\_setSerial command initialize the serial of your og\_tools component. without this call, or if the licence is incorrect, you will stay in trial mode.

- your full unrestricted time is one hour.
- after one hour, you go in restricted mode.
- In this mode, you can still use ogTools, but you will be prompted with a menu asking you for a very simple answer.



# General

## wog\_\_getModules

| wog__getModules -> ob_modules |        |   |   |
|-------------------------------|--------|---|---|
| Parameter                     | Type   |   | Description   |
| ob_modules                    | object | ↩ | Object containing all modules, widgets & pickers & managers, properties |

## Description

The wog\_\_getModules command gives in return the object with all modules and widget & pickers & managers, then options. To give a quick access to each module, methods are given listed below:

| Method                | Direct access to module |
|-----------------------|-------------------------|
| wog__getModule_color  | color                   |
| wog__getModule_io     | io                      |
| wog__getModule_json   | json                    |
| wog__getModule_picker | picker                  |
| wog__getModule_rc     | rc                      |
| wog__getModule_window | window                  |

As the same, methods are given to give quick access to widgets, pickers, functions:

| Method  |
|---|
| wog__getOptions_alert, wog__getOptions_colorIdx, wog__getOptions_colorIdx_displa,<br>wog__getOptions_colorIdy, wog__getOptions_colorIdy_displa, wog__getOptions_colors,<br>wog__getOptions_colors_display, wog__getOptions_confirm, wog__getOptions_date,<br>wog__getOptions_date_display, wog__getOptions_day, wog__getOptions_iog, wog__getOptions_json,<br>wog__getOptions_json_display, wog__getOptions_json_modules, wog__getOptions_more,<br>wog__getOptions_notif, wog__getOptions_period, wog__getOptions_period_display,<br>wog__getOptions_progress, wog__getOptions_progress_type, wog__getOptions_rc,<br>wog__getOptions_rc_display, wog__getOptions_rotator, wog__getOptions_time,<br>wog__getOptions_time_display, wog__getOptions_window, wog__getOptions_year,<br>wog__getOptions_yearMonth, wog__getOptions_yearWeek |



# General parameters

## wog\_\_getGeneral

| wog__getGeneral -> ob_general |        |   |                                      |
|-------------------------------|--------|---|--------------------------------------|
| Parameter                     | Type   |   | Description                          |
| ob_general                    | object | ↔ | Object containing general parameters |

### Description

The wog\_\_getGeneral command in return the object with the general parameters.

## wog\_\_get\_colors\_RWAL

| wog__get_colors_RWAL (ptr_array) |      |   |   |
|----------------------------------|------|---|---|
| Parameter                        | Type |   | Description                               |
| ptr_array                        | ptr  | ↔ | Pointer to an array. Set back to 4 items. |

### Description

The wog\_\_get\_colors\_RWAL command will copy the pointer array with the colors in ogTools.

## wog\_\_get\_current\_year

| wog__get_current_year -> year |         |   |              |
|-------------------------------|---------|---|--------------|
| Parameter                     | Type    |   | Description  |
| year                          | longint | ↔ | Current year |

### Description

The wog\_\_get\_current\_year command gives in return the current year stored is ogTools.





## wog\_\_get\_fiscal\_month

| wog__get_fiscal_month -> month |         |   |                        |
|--------------------------------|---------|---|------------------------|
| Parameter                      | Type    |   | Description            |
| isOk                           | boolean | ↔ | Fiscal month (1 to 12) |

### Description

The wog\_\_get\_fiscal\_month command gives in return the fiscal month stored in ogTools.

## wog\_\_set\_colors\_RWAL

| wog__set_colors_RWAL (ptr_array) |         |   |                                   |
|----------------------------------|---------|---|-----------------------------------|
| Parameter                        | Type    |   | Description                       |
| ptr_array                        | Pointer | → | Pointer to array longint, 4 items |

### Description

The wog\_\_set\_colors\_RWAL command gives to ogTools an array with the colors to be used for consultation, modification, creation, linked. This is mandatory to give a 4 lines array !

## wog\_\_set\_current\_year

| wog__set_current_year (year) |         |   |              |
|------------------------------|---------|---|--------------|
| Parameter                    | Type    |   | Description  |
| year                         | longint | → | Current year |

### Description

The wog\_\_set\_current\_year command gives to ogTools the current year.



## wog\_\_set\_fiscal\_month

| wog__set_fiscal_month (month) |         |   |              |
|-------------------------------|---------|---|--------------|
| Parameter                     | Type    |   | Description  |
| Month                         | longint | → | Fiscal month |

---

### Description

The wog\_\_set\_current\_month command gives to ogTools the fiscal month.



# Overload objects

## wog\_\_ob\_overload

| <b>wog__ob_overload (ob_source ; ob_target)</b> |             |   |                                |
|---|-------------|---|--------------------------------|
| <b>Parameter</b>                                | <b>Type</b> |   | <b>Description</b>             |
| ob_source                                       | object      | → | Source object for overloading  |
| ob_target                                       | object      | → | Target object to be overloaded |

### Description

The wog\_\_ob\_overload command will change, for each property in source, the same property in target, not recursively.

## wog\_\_ob\_overload\_exist

| <b>wog__ob_overload_exist (ob_source ; ob_target)</b> |             |   |                                |
|---|-------------|---|--------------------------------|
| <b>Parameter</b>                                      | <b>Type</b> |   | <b>Description</b>             |
| ob_source   | object      | → | Source object for overloading  |
| ob_target   | object      | → | Target object to be overloaded |

### Description

The wog\_\_ob\_overload\_exist command acts as wog\_\_ob\_overload method, but wont add any property not existing in target.

## wog\_\_ob\_overloads

| <b>wog__ob_overloads (ob_source ; ob_target)</b> |             |   |                                |
|--|-------------|---|--------------------------------|
| <b>Parameter</b>                                 | <b>Type</b> |   | <b>Description</b>             |
| ob_source  | object      | → | Source object for overloading  |
| ob_target  | object      | → | Target object to be overloaded |

### Description

The wog\_\_ob\_overloads command will change, for each property in source, the same property in target, recursively.



## wog\_\_ob\_overloads\_exist

| wog__ob_overloads_exist (ob_source ; ob_target) |        |   |                                |
|---|--------|---|--------------------------------|
| Parameter                                       | Type   |   | Description                    |
| ob_source                                       | object | → | Source object for overloading  |
| ob_target                                       | object | → | Target object to be overloaded |

### Description

The wog\_\_ob\_overloads\_exist command acts as wog\_\_ob\_overloads method, but wont add any property not existing in target.

## wog\_\_ob\_update\_exists

| wog__ob_update_exists (ob_source ; ob_target) |        |   |                                |
|---|--------|---|--------------------------------|
| Parameter                                     | Type   |   | Description                    |
| ob_source                                     | object | → | Source object for overloading  |
| ob_target                                     | object | → | Target object to be overloaded |

### Description

The wog\_\_ob\_update\_exists command will update ob\_target, recursively, where the « flat » ob\_source properties exist. This is useful to modify a specific property everywhere in all widgets.



# Overload options and general

## wog\_\_setOptions

| wog__setOptions (ob_source) |        |   |   |
|-----------------------------|--------|---|---|
| Parameter                   | Type   |   | Description                                       |
| ob_source                   | object | → | Source object to overload the main modules object |

### Description

The wog\_\_setOptions command gives you the ability to overload, recursively, the main object for modules.

## wog\_\_setOptions\_general

| wog__setOptions_general (ob_source) |        |   |  |
|-------------------------------------|--------|---|--|
| Parameter                           | Type   |   | Description                                  |
| ob_source                           | object | → | Source object to overload the general object |

### Description

The wog\_\_setOptions command gives you the ability to overload, recursively, the general object.



## Instances: data and options getters

All widgets and pickers instances have a name. With this name, you can access all of them individually.

When you are inside the instance, which means inside the method of the instance, and you want to access this own instance, there is no need to give that name, this is done by default.

That is why in all the following methods, the {widget} parameter is optional.

From outside, you can use all those methods but giving this « widget » name to the methods.

To get and set values for widgets and pickers, you will see different methods depending of the type of data expected. This can be longint, text, date, object...

So it is important to use the correct getter and setter regarding the expected type for the widget.

For time value, this is managed with the longint type.



## wog\_getValue

| wog_getValue ({widget}) -> value |         |   |                                |
|----------------------------------|---------|---|--------------------------------|
| Parameter                        | Type    |   | Description                    |
| {widget}                         | text    | → | Widget name, or current widget |
| value                            | longint | ↩ | Value stored in widget         |

### Description

The wog\_getValue command gives back the value stored in longint format when this is relevant for a widget.

## wog\_getValues

| wog_getValues (ptrValue ; ptrValue1 ; {widget}) |      |   |                                |
|---|------|---|--------------------------------|
| Parameter                                       | Type |   | Description                    |
| value   | ptr  | → | Pointer to first value         |
| value1  | ptr  | → | Pointer to second value        |
| {widget}  | text | → | Widget name, or current widget |

### Description

The wog\_getValues command gives back the two values stored in longint format when this is relevant for a widget.

## wog\_getValue\_date

| wog_getValue_date ({widget}) -> date |      |   |                                |
|--------------------------------------|------|---|--------------------------------|
| Parameter                            | Type |   | Description                    |
| {widget}                             | text | → | Widget name, or current widget |
| date                                 | date | ↩ | Value stored in widget         |

### Description

The wog\_getValue\_date command gives back the value stored in date format when this is relevant for a widget.



## wog\_getValue\_dates

| wog_getValue_date (ptr_date ; ptr_date1 ; {widget}) -> date |         |   |                                |
|---|---------|---|--------------------------------|
| Parameter   | Type    |   | Description                    |
| ptr_date  | pointer | → | Pointer to date one            |
| ptr_date1   | pointer | → | Pointer to date two            |
| {widget}  | text    | → | Widget name, or current widget |
| date  | date    | ↩ | Value stored in widget         |

### Description

The wog\_getValue\_dates command gives back the date values stored in date format when this is relevant for a widget. Dedicated to period widgets.

## wog\_getValue\_ob

| wog_getValue_ob ({widget}) -> ob |        |   |                                |
|----------------------------------|--------|---|--------------------------------|
| Parameter                        | Type   |   | Description                    |
| {widget}                         | text   | → | Widget name, or current widget |
| ob                               | object | ↩ | Value stored in widget         |

### Description

The wog\_getValue\_ob command gives back the value stored in object when this is relevant for a widget.

## wog\_getValue\_txt

| wog_getValue_txt ({widget}) -> txt |      |   |                                |
|------------------------------------|------|---|--------------------------------|
| Parameter                          | Type |   | Description                    |
| {widget}                           | text | → | Widget name, or current widget |
| txt                                | text | ↩ | Value stored in widget         |

### Description

The wog\_getValue\_txt command gives back the value stored in text when this is relevant for a widget.





## wog\_getOptions

| wog_getOptions ({widget}) -> ob_options |        |   |                                |
|---|--------|---|--------------------------------|
| Parameter                               | Type   |   | Description                    |
| {widget}                                | text   | → | Widget name, or current widget |
| ob_options                              | object | ↔ | Value stored in widget         |

### Description

The wog\_getOptions command gives back the object value stored in options for a widget.

## wog\_getRWAL

| wog_getRWAL ({widget}) -> rwal |         |   |                                |
|--------------------------------|---------|---|--------------------------------|
| Parameter                      | Type    |   | Description                    |
| {widget}                       | text    | → | Widget name, or current widget |
| rwal                           | longint | ↔ | Value stored in widget         |

### Description

The wog\_getRWAL command gives back the value stored in options for rwal (read, write, add, link).

## wog\_getEditable

| wog_getEditable ({widget}) -> is_editable |         |   |                                |
|---|---------|---|--------------------------------|
| Parameter                                 | Type    |   | Description                    |
| {widget}                                  | text    | → | Widget name, or current widget |
| is_editable                               | boolean | ↔ | Value stored in widget         |

### Description

The wog\_getEditable command gives back the value stored in options based on rwal (1 or 2 gives true).



# Instances: data and options setters

## wog\_setValue

| <b>wog_setValue (value ; {widget})</b> |             |   |                                |
|--|-------------|---|--------------------------------|
| <b>Parameter</b>                       | <b>Type</b> |   | <b>Description</b>             |
| value                                  | longint     | → | Value to store in widget       |
| {widget}                               | text        | → | Widget name, or current widget |

### Description

The wog\_setValue command gives stores the value in longint format (when this is relevant for a widget).

## wog\_setValue\_date

| <b>wog_setValue_date (date ; {widget})</b> |             |   |                                |
|--|-------------|---|--------------------------------|
| <b>Parameter</b>                           | <b>Type</b> |   | <b>Description</b>             |
| date                                       | date        | → | Value to store in widget       |
| {widget}                                   | text        | → | Widget name, or current widget |

### Description

The wog\_setValue\_date command stores the value in date format (when this is relevant for a widget).



## wog\_setValue\_dates

| wog_setValue_dates (date ; date1 ; {widget}) |      |   |                                |
|--|------|---|--------------------------------|
| Parameter                                    | Type |   | Description                    |
| date   | date | → | Value to store in widget       |
| date1  | date | → | Value to store in widget       |
| {widget}                                     | text | → | Widget name, or current widget |

### Description

The wog\_getValue\_dates command store the values in date format (when this is relevant for a widget, dedicated to period widgets).

## wog\_setValue\_ob

| wog_setValue_ob (ob ; {widget}) |        |   |                                |
|---------------------------------|--------|---|--------------------------------|
| Parameter                       | Type   |   | Description                    |
| ob                              | object | → | Value to store in widget       |
| {widget}                        | text   | → | Widget name, or current widget |

### Description

The wog\_getValue\_ob command stores the value in object format (when this is relevant for a widget).

## wog\_setValue\_txt

| wog_setValue_txt (txt ; {widget}) |      |   |                                |
|-----------------------------------|------|---|--------------------------------|
| Parameter                         | Type |   | Description                    |
| txt                               | text | → | Value to store in widget       |
| {widget}                          | text | → | Widget name, or current widget |

### Description

The wog\_setValue\_txt command stores the value in text format (when this is relevant for a widget).

## wog\_setOptions



| <b>wog_getOptions (ob_options ; {widget})</b> |             |   |                                |
|---|-------------|---|--------------------------------|
| <b>Parameter</b>                              | <b>Type</b> |   | <b>Description</b>             |
| ob_options                                    | object      | → | Value to store in widget       |
| {widget}                                      | text        | → | Widget name, or current widget |

## Description

The wog\_setOptions command stores the value in object format to the options of the instance.

## wog\_setRWAL

| <b>wog_setRWAL (rwal ; {widget})</b> |             |   |                                |
|--------------------------------------|-------------|---|--------------------------------|
| <b>Parameter</b>                     | <b>Type</b> |   | <b>Description</b>             |
| rwal                                 | longint     | → | Value to store in widget       |
| {widget}                             | text        | → | Widget name, or current widget |

## Description

The wog\_setRWAL command stores the value in longint format for rwal (read, write, add, link).

## wog\_setEditable

| <b>wog_setEditable (is_editable ; {widget})</b> |             |   |                                |
|---|-------------|---|--------------------------------|
| <b>Parameter</b>                                | <b>Type</b> |   | <b>Description</b>             |
| is_editable                                     | boolean     | → | Value to store in widget       |
| {widget}  | text        | → | Widget name, or current widget |

## Description

The wog\_setEditable command stores the value in boolean format to rwal (0 or 1).



## wog\_setValue\_ptr

| <b>wog_setValue_ptr (ptr ; {widget})</b> |             |   |                                |
|--|-------------|---|--------------------------------|
| <b>Parameter</b>                         | <b>Type</b> |   | <b>Description</b>             |
| ptr                                      | pointer     | → | pointer to a value provider    |
| {widget}                                 | text        | → | Widget name, or current widget |

### Description

The wog\_setValue\_ptr command stores a pointer to a value.

## wog\_setValues\_ptr

| <b>wog_setValues_ptr (ptr ; ptr1 ; {widget})</b> |             |   |                                       |
|--|-------------|---|---------------------------------------|
| <b>Parameter</b>                                 | <b>Type</b> |   | <b>Description</b>                    |
| ptr  | pointer     | → | Pointer to store in widget for value  |
| ptr1   | pointer     | → | Pointer to store in widget for value1 |
| {widget}   | text        | → | Widget name, or current widget        |

### Description

The wog\_setValue\_ptr command stores two pointers to two values.



# Instances: miscellaneous

## wog\_resize

| wog_resize ({border} ; {widget}) |         |   |  |
|----------------------------------|---------|---|--|
| Parameter                        | Type    |   | Description  |
| {border}                         | longint | → | Border to apply to widget. By default, « border none » is used. Pass a value <0 to get it unchanged. |
| {widget}                         | text    | → | Widget name, or current widget   |

### Description

The wog\_resize command is very important and as to be used in almost all widgets on event « on load ». This will:

- resize correctly the widget content to the instance size in the form
- remove any used border for layout purpose. As it is easier to see your widget with a border in a layout, don't worry as this method will set it to « border none » when called without parameters.
- in case of a color widget, this will manage the instance name to detect « § » in it. If yes, then the 3 next chars are used to determine the space, the sf mode, and the type of display.
  - « 4 » for 4D, « M » for MD
  - « Y » for double colors - single otherwise
  - « n » value from 0 to 9 for the type of color widget

## wog\_redraw

| wog_redraw ({widget}) |      |   |                                |
|-----------------------|------|---|--------------------------------|
| Parameter             | Type |   | Description                    |
| {widget}              | text | → | Widget name, or current widget |

### Description

The wog\_redraw command let you redraw the content based on the value stored in the widget, but first outside with the data referred with the pointer if used.

You can use it when you modify yourself the value from outside the widget, to refresh it.



## wog\_calculateRWAL

| wog_calculateRWAL (isModif ; isAdd) -> rwal |         |   |                           |
|---|---------|---|---------------------------|
| Parameter                                   | Type    |   | Description               |
| isModif                                     | boolean | → | true if modification mode |
| isAdd                                       | boolean | → | true if new record        |
| rwal  | longint | ↔ | Value for rwal            |

### Description

The wog\_calculateRWAL command gives back the value to use as a rwal indicator for a widget.

## wog\_calculate\_newModif

| wog_calculate_newModif ( table   field ; ptr_isNew ; ptr_isModif) -> is_new |          |   |  |
|---|----------|---|--|
| Parameter   | Type     |   | Description  |
| table   field   | pointer  | → | Pointer to a field or a table gives if record is modifiable or add mode. |
| ptr_isNew   | pointeur | → | true if new record   |
| ptr_isModif   | pointer  | → | true if record not locked and modifiable                                 |
| is_new  | boolean  | ↔ | redondant to allow the use with « if » statement to initialize value     |

### Description

The wog\_calculate\_newModif command gives back the information on the record (table or field).

## wog\_calculate\_newModif

| wog_calculate_newModif ( table   field ; ptr_isNew ; ptr_isModif) -> is_new |          |   |  |
|---|----------|---|--|
| Parameter   | Type     |   | Description  |
| table   field   | pointer  | → | Pointer to a field or a table to determine if record is modifiable or in add mode. |
| ptr_isNew   | pointeur | → | true if new record   |
| ptr_isModif   | pointer  | → | true if record not locked and modifiable   |
| is_new  | boolean  | ↔ | redondant to allow the use with « if » statement to initialize value               |

### Description

The wog\_calculate\_newModif command gives back the information on the record (table or field).



## wog\_setRWAL\_fromField

| wog_setRWAL_fromField ( table   field ; {widget} ) -> isNew |         |   |  |
|---|---------|---|--|
| Parameter   | Type    |   | Description  |
| table   field   | pointer | → | Pointer to a field or a table to determine if record is modifiable or in add mode. |
| {widget}  | text    | → | Widget name, or current widget   |
| isNew   | boolean | ↔ | redundant to allow the use with « if » statement to initialize value               |

### Description

The wog\_getEditable command gives back the value stored in options based on rwal (1 or 2 gives true).





# Examples and usage

## Description

This is a full generic example, where only the initialization in case of a new record will change, according to the type.

### Method in the widget (single value, color)

```
C_POINTER($field)
$field:=->[Evénements]colors

C_LONGINT($evt)
$evt:=Form event
Case of
  : ($evt=On_Load)
    wog_setOptions (<ob_colors_txt) // option for the colors widget
    wog_resize (Border Dotted)

    If (wog_setRWAL_fromField ($field))
      // Initialisation in case of a new record
      $field->:=wog_colorsMDrandom
    End if
    wog_setValue_ptr ($field)

End case
```



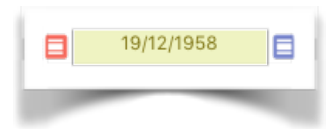
### Method in the widget (single value, date)

```
C_POINTER($field;$table)
$field:=->[Plans]DateSem

C_LONGINT($evt)
$evt:=Form event
Case of
  : ($evt=-On_Load)
    wog_resize
    If (wog_setRWAL_fromField ($field))
      $field->:=vDATE_Calcul (-4;Current date)
    End if
    wog_setValue_ptr ($field)
    v_AnneeSemDF:=F_Plan_weekDisplay_txt ([Plans]DateSem)

  : ($evt=On_Data_Change)
    v_AnneeSemDF:=F_Plan_weekDisplay_txt ([Plans]DateSem)
    F_PLA_PLL_groutine ("LB_REDRAW")

End case
```



You can add this to catch a change of data

```
: ($evt=On_Data_Change)
// Do what you need
```



## Method in the widget (double values, dates)

```

C_POINTER($ptr_date_start;$ptr_date_end)
$ptr_date_start:=->[Congés]DébutDate
$ptr_date_end:=->[Congés]FinDate

C_LONGINT($evt)
$evt:=Form event
Case of
: ($evt=On_Load)
  wog_resize
  If (wog_setRWAL_fromField ($ptr_date_start))
    $ptr_date_start->:=Current date
    $ptr_date_end->:=$ptr_date_start->+6
  End if
  wog_setValues_ptr ($ptr_date_start;$ptr_date_end)

: ($evt=On_Data_Change)
  // Do what you need

End case

```

