# wox_Xlibrary

Common library

# Programmer's guide

**Component's user data sheet**

| Release | Version | Date | Rédacteur | Commentaires |
|---|---|---|---|---|
| Initial | | 28 janvier 2023 | OG | First version |
| R1 | | 2 février 2026 | OG | Adaptation to wox |
| | | | | |
| | | | | |
| | | | | |

**from v20R3, for project mode**

Protée
IT, app design

Protée sarl
31, rue Sainte-Marthe
31000 TOULOUSE

Mob : 06 3718 5941
www.protee.org / info@protee.org

**Overview** **3**
Introduction 3
Configuration 3
Library subclasses 3
**Methods** **4**
wox__ and wox_ 4
wox_app_ 6
wox_num_ 6
wox_prefs_ 7
wox_sounds_ 9
wox_str_ 11
wox_vJ_ 12
**Thanks** **13**

# Overview

## Introduction

The **ogKrolific** component first intent is to manage our licensing system, but it helps the developer too providing some very common methods for all our ogTools suite components.
This documentation gives informations on this library available for you.
All methods « wok_ » are intended for licensing system, and are not documented here. If you are interested with our licensing system, ask us directly.
We cover in this documentation all methods for basic generic needs, with methods beginning with « wox_ », with some exceptions.

## Configuration

When you initialise any components with wo_initRegister, the ogKrolific is automatically initialised.

## Library subclasses

The sub library classes are

- wok__ and wok_
- wox_app_ For the apps
- wox_num_ Methods for Numerics
- wox_prefs_ Prefs json manager
- wox_sounds_ Sounds manager
- wox_str_ Methods for String
- wox_vJ_ Methods for objects vJ

# Methods

The library is split into sub methods, depends on wox_xxx_, with xxx into : app, num, prefs, sounds, str, vJ, window.

## wox___ and wox_

### wox__storage_prefs

#### Description

This main object is accessible with this method, and stores all the global preferences.

| #DECLARE->$vJ_prefs : Object | | | |
|---|---|---|---|
| **Parameter** | **Type** | | **Description** |
| vJ | object | ↩ | Gives back the main prefs object of ogKrolific |

| Property | Type | Default | Contents |
|---|---|---|---|
| t_name | text | ogKrolific | Component name |
| t_version | text | 2.1.00 | Current version |
| t_lip_app | text | wok | License app name |
| l_top | number | 130 | pixels from top as non usable area for Form windows |
| t_formatDate | text | | Format date found based on system language. |
| prefs_root | text | ogToolsSuite | Root folder for prefs in Document |
| t_host_name | text | | Useful to get an additional folder for preferences path, making all preferences unique for that host. |
| snd_progress | text | "click_1" | Default sound for progress |
| snd_warning | text | « starwar_r2d2_07" | Default sound for warning |
| snd_error | text | "starwar_r2d2_06" | Default sound for error |
| snd_done | text | "bicycle" | Default sound for done |
| j_apps | object | | Splash screen manager options. See below |

### j_apps

| Property | Type | Default | Contents |
|---|---|---|---|
| l_seconds | num | 6 | Number of seconds for splash screen for components |
| l_x | num | 0 | Initial x for splash screen |
| l_y | num | prefs.l_top | Initial y for splash screen |
| l_count_hv | num | 5 | Barbers count, >0 horizontal, or vertical |
| l_gap | num | 4 | Gap between splash screen |

| Property | Type | Default | Contents |
|---|---|---|---|
| is_always | boolean | false | If true, the splash screen appears on every wo•_initRegister() |
| is_end_all | boolean | true | If true, all splash screen will be closed synchronised on the last to close. |

## wox_initRegister

### Description

This method initialise the full component. You have to call this method before using any of the other methods. There is no need of any parameters, they are here for compatibility with other wo•_initRegister().

| #DECLARE($vT_host_name : Text; $vT_serial : Text)->$is_serial_ok : Boolean | | | |
|---|---|---|---|
| **Parameter** | **Type** | | **Description** |
| {$vT_host_name} | text | → | Host name associated to the serial. |
| {$vT_serial} | text | → | Serial of your component. |
| $isOk | boolean | ↩ | True if register successful. |

# wox_app_

## wox_app_prefs_get_path

### Description

Based on the t_name of your caller and t_host_name saved in ogKrolific preferences, it gives back a cs.folder on the one to use with methods for preferences. If you give $is_root true, then the t_host_name is not taken into account.

| #DECLARE($vJ_app_prefs : Object; $is_root : Boolean)->$vJ_path : Object | | | |
|---|---|---|---|
| **Parameter** | **Type** | | **Description** |
| $vJ_app_prefs | object | → | Used to collect the caller wo•__storage_prefs.t_name |
| $is_root | boolean | → | If true, will force not to use wok__storage_prefs.t_host_name |
| $vJ_path | object | ↩ | cs.folder on prefs folder for that $vJ_app_prefs |

## wox_app_set_t_host_name

### Description

Set in ogKrolific preferences,  the t_host_name. If not empty, all prefs for your host and ogToolsSuite components will use this value as a subFolder inside the root prefs folder (Documents/ogToolSuite).

| #DECLARE($vT_host_name : Text) | | | |
|---|---|---|---|
| **Parameter** | **Type** | | **Description** |
| $vT_host_name | text | → | Put the value inside ogKrolific.prefs.t_host_name. It must be a valid file name compliant. |

# wox_num_

Methods for numeric conversions.

## wox_num_hexToDec

### Description

Giving an hex string, you get back the decimal value of it.

| #DECLARE($vT_HexValue : Text)->$vL_DecValue : Integer | | | |
|---|---|---|---|
| **Parameter** | **Type** | | **Description** |
| $vT_HexValue | text | → | Hex representation of a number |
| $vL_DecValue | longint | ↩ | Decimal value of the supplied hex representation |

# wox_prefs_

Methods to manage preferences for all components. Only « .json » files are taken into account.

## wox_prefs_choose

### Description

Open a menu with all files in the provided $vJ_path, and gives back $vJ_file to the selected file. If none selected, $vJ_file is Null.
The $is_save option add an item « Choose a file » at first, and if this item is selected, it asks you to enter a filename, and this is the $vJ_file given back.

| #DECLARE($vJ_path : Object; $is_save : Boolean)->$vJ_file : Object | | | |
|---|---|---|---|
| **Parameter** | **Type** | | **Description** |
| $vJ_path | object | → | Optionnel host options. If not given, uses prefs.j_host |
| $is_save | boolean | → | Mode choose for save |
| $vJ_file | object | ↩ | True if Build successful. |

## wox_prefs_load

### Description

Parse a json prefs file and check if exists and if version is ok. If not, generate a new object ready for default values.

| #DECLARE($vJ_pref_file : Object; $vP_vJ_prefs : Pointer; $vL_version : Integer)->$isOk : Boolean | | | |
|---|---|---|---|
| **Parameter** | **Type** | | **Description** |
| $vJ_pref_file | object | → | Prefs file to open |
| $vP_vJ_prefs | pointer | ⇄ | Pointer to an object to get back the result of the json file. |
| $vL_version | longint | → | Version the file must have. If the file does not exist, or if the version stored in the file is lower than the value, a new object is created with the new version property « __version », and isOk is false to make you put the new default values in it. |
| $isOk | boolean | ↩ | True if prefs file loaded properly. |

## wox_prefs_save

### Description

Save a json prefs file.

| #DECLARE($vJ_pref_file : Object; $vJ_prefs : Object) | | | |
|---|---|---|---|
| **Parameter** | **Type** | | **Description** |
| $vJ_pref_file | object | → | Prefs file to save |
| $vJ_prefs | object | → | Pointer to an object to get back the result of the json file. |

## wox_prefs_windows_get

### Description

This will set the position and size for a Form window, based on Form.prefs properties left, top, width, height.
When you load preferences file, you are supposed to put it in .prefs object before calling Dialog. This has to be called at your Form On Load event. So this method works with wox_prefs_windows_set.

| #DECLARE($is_centered : Boolean) | | | |
|---|---|---|---|
| **Parameter** | **Type** | | **Description** |
| $is_centered | boolean | → | If true, the left & top are not used and the from window is centred, only the size is made based on width and height. |

## wox_prefs_windows_set

### Description

This will get the position and size of a Form window, and will write back on Form.prefs properties left, top, width, height, updating the .prefs ready for save on the caller. This has to be called at your Form On Unload, or On CloseBox event.
So this method works with wox_prefs_windows_get.

### Example of Form call

```
#DECLARE->$isOk : Boolean

var $vT_form : Text
$vT_form:="mng_methods_vars"

var $vJ_pref_file : Object
$vJ_pref_file:=wod__prefs_get_path.file($vT_form+".json")

var $vJ_prefs : Object
$vJ_prefs:=New object
$isOk:=wox_prefs_load($vJ_pref_file; ->$vJ_prefs; 1)  // Version, increase to reset
If (Not($isOk))
      $vJ_prefs.method_name:="initial"
      $vJ_prefs.method_colors:=58975  // rgb
End if

var $vJ_form : Object
$vJ_form:=New object
$vJ_form.prefs:=$vJ_prefs

var $vL_formType : Integer
$vL_formType:=Plain form window
var $vL_windowRef : Integer
$vL_windowRef:=Open form window($vT_form; $vL_formType; Horizontally centered; Vertically centered)
SET WINDOW TITLE(Current method name; $vL_windowRef)
DIALOG($vT_form; $vJ_form)
CLOSE WINDOW($vL_windowRef)
wox_prefs_save($vJ_pref_file; $vJ_prefs)
$isOk:=(OK=1)  // SAVE
```

# wox_sounds_

## wox_sounds_choose

### Description

Based on a given collection, or on what is in the sounds/ folder, this open a menu and gives back the selected sound name in the collection.

| #DECLARE($vC_sounds : Collection)->$vT_sound : Text | | | |
|---|---|---|---|
| **Parameter** | **Type** | | **Description** |
| $vC_sounds | collection | → | Optionnel collection of sounds. Not given, will use wox_sounds_getFiles |
| $vT_sound | text | ↩ | Sound name to play, or empty if none selected |

## wox_sounds_getFiles

### Description

Gives back all the files in the sounds folder, without extension as they are all supposed to be mp3.

| #DECLARE->$vC_sounds : Collection | | | |
|---|---|---|---|
| **Parameter** | **Type** | | **Description** |
| $vC_sounds | collection | ↩ | All the files in resources/sounds/, without the .mp3 extension |

## wox_sounds_play

### Description

Play the given sound from sounds/ folder. Don't give any extension has it is added as « .mp3 ».

| #DECLARE($vT_sound : Text) | | | |
|---|---|---|---|
| **Parameter** | **Type** | | **Description** |
| $vT_sound | text | → | Sound name, with no extension. |

## wox_sounds_play_done
## wox_sounds_play_error
## wox_sounds_play_progress
## wox_sounds_play_warning

### Description

It plays sound based on wok__storage_prefs, and associated properties. This does a generic call to usual sounds, and the capacity to change the one to use for each of those 4 cases.

## wox_sounds_play_random

### Description

It plays a random sound from the sounds folder.

# wox_sounds_random_get

## Description

Based on a given collection, or on what is in the sounds/ folder, this gives back a random sound name found in the collection.

| #DECLARE($vC_sounds : Collection)->$vT_sound : Text | | | |
|---|---|---|---|
| **Parameter** | **Type** | | **Description** |
| $vC_sounds | collection | → | Optionnel collection of sounds. Not given, will use wox_sounds_getFiles |
| $vT_sound | text | ↩ | Sound name to play |

# wox_str_

## wox_str_pluralise

### Description

Calculate the singular plural for a given word.

| #DECLARE($vL_Count : Integer; $vT_Singular : Text; $vT_Plural : Text)->$vT_String : Text | | | |
|---|---|---|---|
| **Parameter** | **Type** | | **Description** |
| $vL_Count | longint | → | how many value to determine plural |
| $vT_Singular | text | → | Singular form of word |
| {$vT_Plural} | text | → | Optional plural form of word. If not given, an « s » is added for plural. |
| $vT_String | text | ↩ | Resulting string |

## wox_str_squeeze

### Description

Calculate the singular plural for a given word.
Removes whitespace (eg. spaces, tabs, carriage returns) from the beginning and end of the supplied string.
Compresses whitespace within the string to a single character. The character chosen is the lowest index in the list
Carriage return ; Tab ; LF ASCII code ;  space ; 160 (non breaking space). e.g. <>SP+<>CR+<>TB is replaced with <>CR

| #DECLARE($vT_Input : Text)->$vT_Output : Text | | | |
|---|---|---|---|
| **Parameter** | **Type** | | **Description** |
| $vT_Input | text | → | how many value to determine plural |
| $vT_Output | text | ↩ | Resulting string |

# wox_vJ_

## wox_vJ_overload

### Description

If no added text properties are given, a single level copy is made from source to target.
Otherwise, only the existing properties of the given parameters are copied if exist.

| #DECLARE($vJ_source : Object; $vJ_target : Object {;$3+}) | | | |
|---|---|---|---|
| **Parameter** | **Type** | | **Description** |
| $vJ_source | object | → | Source for overload |
| $vJ_target | object | → | Target to overload |
| ${3} + | text | → | List of properties to overload |

## wox_vJ_overload_inex

### Description

From source, add at a single level all the properties that do not exist in target.

| #DECLARE($vJ_source : Object; $vJ_target : Object) | | | |
|---|---|---|---|
| **Parameter** | **Type** | | **Description** |
| $vJ_source | object | → | Source for overload |
| $vJ_target | object | → | Target to overload |

## wox_vJ_overloads

### Description

If no added text properties are given, a recursive levels copy is made from source to target.
Otherwise, it acts as wox_vJ_overload with properties.

| #DECLARE($vJ_source : Object; $vJ_target : Object {;$3+}) | | | |
|---|---|---|---|
| **Parameter** | **Type** | | **Description** |
| $vJ_source | object | → | Source for overload |
| $vJ_target | object | → | Target to overload |
| ${3} + | text | → | List of properties to overload |

## wox_vJ_overloads_back

### Description

Works like wox_vJ_overloads in reverse. Just for a better reading, you can use wox_vJ_overloads and exchanging the two first parameters.

| #DECLARE($vJ_target : Object; $vJ_source : Object {;$3+}) | | | |
|---|---|---|---|
| **Parameter** | **Type** | | **Description** |
| $vJ_target | object | → | Target to overload |
| $vJ_source | object | → | Source for overload |
| ${3} + | text | → | List of properties to overload |

## wox_vJ_overloads_exist

### Description

It is a recursive levels copy for all properties in source, only for existing properties in target.

| #DECLARE($vJ_source : Object; $vJ_target : Object) | | | |
|---|---|---|---|
| **Parameter** | **Type** | | **Description** |
| $vJ_source | object | → | Source for overload |
| $vJ_target | object | → | Target to overload |

## wox_vJ_updates_exist

### Description

It is a recursive levels copy for all properties in target, that exist in source. Source is without recursivity. This allows to set some properties given in source to all targets recursively existing in target.

| #DECLARE($vJ_source : Object; $vJ_target : Object) | | | |
|---|---|---|---|
| **Parameter** | **Type** | | **Description** |
| $vJ_source | object | → | Source for overload |
| $vJ_target | object | → | Target to overload |

# Thanks

Thanks to Patrick Emanuel for our long-term collaboration and help during the tests.

oooOOOooo