

# Lua API

## Version 6.1.0

### Table of Contents

1. darktable .....	4
1.1. darktable.print .....	4
1.2. darktable.print_log .....	4
1.3. darktable.print_error .....	4
1.4. darktable.register_event .....	4
1.5. darktable.register_storage .....	5
1.6. darktable.register_lib .....	9
1.7. darktable.films .....	10
1.8. darktable.new_format .....	11
1.9. darktable.new_storage .....	12
1.10. darktable.new_widget .....	12
1.11. darktable.gui .....	13
1.12. darktable.guides .....	30
1.13. darktable.tags .....	32
1.14. darktable.configuration .....	34
1.15. darktable.preferences .....	36
1.16. darktable.styles .....	39
1.17. darktable.database .....	42
1.18. darktable.collection .....	44
1.19. darktable.control .....	44
1.20. darktable.gettext .....	46
1.21. darktable.debug .....	48
2. types .....	51
2.1. types.lua_os_type .....	51
2.2. types.dt_lua_image_t .....	51
2.3. types.dt_imageio_module_format_t .....	58
2.4. types.dt_imageio_module_format_data_png .....	60
2.5. types.dt_imageio_module_format_data_tiff .....	60
2.6. types.dt_imageio_module_format_data_exr .....	60
2.7. types.dt_imageio_module_format_data_copy .....	61
2.8. types.dt_imageio_module_format_data_pfm .....	61
2.9. types.dt_imageio_module_format_data_jpeg .....	61
2.10. types.dt_imageio_module_format_data_ppm .....	61
2.11. types.dt_imageio_module_format_data_webp .....	61
2.12. types.dt_imageio_module_format_data_j2k .....	62
2.13. types.dt_imageio_module_format_data_pdf .....	63
2.14. types._pdf_mode_t .....	64
2.15. types._pdf_pages_t .....	64
2.16. types.dt_pdf_stream_encoder_t .....	65
2.17. types.dt_imageio_module_storage_t .....	65
2.18. types.dt_imageio_module_storage_data_email .....	66
2.19. types.dt_imageio_module_storage_data_latex .....	66
2.20. types.dt_imageio_module_storage_data_piwigo .....	67
2.21. types.dt_imageio_module_storage_data_gallery .....	67

2.22. types.dt_imageio_module_storage_data_disk .....	67
2.23. types.dt_lua_film_t .....	68
2.24. types.dt_style_t .....	69
2.25. types.dt_style_item_t .....	69
2.26. types.dt_lua_tag_t .....	70
2.27. types.dt_lua_lib_t .....	70
2.28. types.dt_lua_view_t .....	72
2.29. types.dt_lua_backgroundjob_t .....	72
2.30. types.dt_lua_snapshot_t .....	72
2.31. types.hint_t .....	73
2.32. types.dt_ui_container_t .....	73
2.33. types.snapshot_direction_t .....	74
2.34. types.dt_imageio_j2k_format_t .....	74
2.35. types.dt_imageio_j2k_preset_t .....	74
2.36. types.comp_type_t .....	75
2.37. types.lua_pref_type .....	75
2.38. types.dt_imageio_exr_compression_t .....	75
2.39. types.dt_lib_collect_params_rule_t .....	76
2.40. types.dt_lib_collect_mode_t .....	76
2.41. types.dt_collection_properties_t .....	77
2.42. types.dt_collection_sort_t .....	77
2.43. types.dt_collection_sort_order_t .....	78
2.44. types.dt_collection_filter_t .....	78
2.45. types.dt_collection_rating_comperator_t .....	79
2.46. types.dt_lua_orientation_t .....	79
2.47. types.dt_lua_align_t .....	79
2.48. types.dt_lua_ellipsize_mode_t .....	80
2.49. types.dt_lua_cairo_t .....	80
2.50. types.dt_ui_panel_t .....	86
2.51. types.dt_lighttable_layout_t .....	86
2.52. types.lua_widget .....	86
2.53. types.lua_container .....	88
2.54. types.lua_check_button .....	88
2.55. types.lua_label .....	89
2.56. types.lua_button .....	90
2.57. types.lua_box .....	91
2.58. types.lua_entry .....	91
2.59. types.lua_separator .....	92
2.60. types.lua_combobox .....	93
2.61. types.lua_file_chooser_button .....	94
2.62. types.lua_stack .....	95
2.63. types.lua_slider .....	96
2.64. types.lua_text_view .....	97
2.65. types.lua_section_label .....	98
3. events .....	99
3.1. events.intermediate-export-image .....	99
3.2. events.post-import-image .....	99
3.3. events.shortcut .....	100
3.4. events.post-import-film .....	101
3.5. events.view-changed .....	101
3.6. events.global_toolbox-grouping_toggle .....	102
3.7. events.global_toolbox-overlay_toggle .....	102
3.8. events.mouse-over-image-changed .....	103
3.9. events.exit .....	103
3.10. events.pre-import .....	103

4. attributes .....	105
4.1. attributes.write .....	105
4.2. attributes.has_tostring .....	105
4.3. attributes.implicit_yield .....	105
4.4. attributes.parent .....	105

## 1. darktable

The darktable library is the main entry point for all access to the darktable internals.

To access the darktable specific functions you must load the darktable environment:

```
darktable = require "darktable"
```

All functions and data are accessed through the darktable module.

### 1.1. darktable.print

```
function(  
  message : string  
)
```

Will print a string to the darktable control log (the long overlaid window that appears over the main panel).

message

string

The string to display which should be a single line.

### 1.2. darktable.print\_log

```
function(  
  message : string  
)
```

This function will print its parameter if the Lua logdomain is activated. Start darktable with the "-d lua" command line option to enable the Lua logdomain.

message

string

The string to display.

### 1.3. darktable.print\_error

```
function(  
  message : string  
)
```

This function is similar to `darktable.print_log` but adds an ERROR prefix for clarity.

message

string

The string to display.

### 1.4. darktable.register\_event

```
function(  
  event_type : string,  
  callback : function,
```

```
... : variable
)
```

This function registers a callback to be called when a given event happens.

Events are documented in the event section.

**event\_type**

```
string
```

The name of the event to register to.

**callback**

```
function
```

The function to call on event. The signature of the function depends on the type of event.

...

```
variable
```

Some events need extra parameters at registration time; these must be specified here.

## 1.5. darktable.register\_storage

```
function(
  plugin_name : string,
  name : string,
  [store : function],
  [finalize : function],
  [supported : function],
  [initialize : function],
  [widget : types.lua_widget]
)
```

This function will add a new storage implemented in Lua.

A storage is a module that is responsible for handling images once they have been generated during export. Examples of core storages include filesystem, e-mail, facebook...

**plugin\_name**

```
string
```

A Unique name for the plugin.

**name**

```
string
```

A human readable name for the plugin.

**[store]**

```
function(
  storage : types.dt_imageio_module_storage_t,
```

```

    image : types.dt_lua_image_t,
    format : types.dt_imageio_module_format_t,
    filename : string,
    number : integer,
    total : integer,
    high_quality : boolean,
    extra_data : table
)

```

This function is called once for each exported image. Images can be exported in parallel but the calls to this function will be serialized.

**storage**

```
types.dt_imageio_module_storage_t
```

The storage object used for the export.

**image**

```
types.dt_lua_image_t
```

The exported image object.

**format**

```
types.dt_imageio_module_format_t
```

The format object used for the export.

**filename**

```
string
```

The name of a temporary file where the processed image is stored.

**number**

```
integer
```

The number of the image out of the export series.

**total**

```
integer
```

The total number of images in the export series.

**high\_quality**

```
boolean
```

True if the export is high quality.

**extra\_data**

```
table
```

An empty Lua table to take extra data. This table is common to the initialize, store and finalize calls in an export series.

### [finalize]

```
function(  
    storage : types.dt_imageio_module_storage_t,  
    image_table : table,  
    extra_data : table  
)
```

This function is called once all images are processed and all store calls are finished.

#### storage

types.dt\_imageio\_module\_storage\_t

The storage object used for the export.

#### image\_table

table

A table keyed by the exported image objects and valued with the corresponding temporary export filename.

#### extra\_data

table

An empty Lua table to store extra data. This table is common to all calls to store and the call to finalize in a given export series.

### [supported]

```
function(  
    storage : types.dt_imageio_module_storage_t,  
    format : types.dt_imageio_module_format_t  
) : boolean
```

A function called to check if a given image format is supported by the Lua storage; this is used to build the dropdown format list for the GUI.

Note that the parameters in the format are the ones currently set in the GUI; the user might change them before export.

#### storage

types.dt\_imageio\_module\_storage\_t

The storage object tested.

#### format

types.dt\_imageio\_module\_format\_t

The format object to report about.

#### return

boolean

True if the corresponding format is supported.

## [initialize]

```
function(  
  storage : types.dt_imageio_module_storage_t,  
  format : types.dt_imageio_module_format_t,  
  images : table of types.dt_lua_image_t,  
  high_quality : boolean,  
  extra_data : table  
) : table or nil
```

A function called before storage happens

This function can change the list of exported functions

### storage

types.dt\_imageio\_module\_storage\_t

The storage object tested.

### format

types.dt\_imageio\_module\_format\_t

The format object to report about.

### images

table of types.dt\_lua\_image\_t

A table containing images to be exported.

### high\_quality

boolean

True if the export is high quality.

### extra\_data

table

An empty Lua table to take extra data. This table is common to the initialize, store and finalize calls in an export series.

### return

table or nil

The modified table of images to export or nil

If nil (or nothing) is returned, the original list of images will be exported

If a table of images is returned, that table will be used instead. The table can be empty. The images parameter can be modified and returned

## [widget]

types.lua\_widget



A widget to display in the export section of darktable's UI

## 1.6. darktable.register\_lib

```
function(  
  plugin_name : string,  
  name : string,  
  expandable : boolean,  
  resettable : boolean,  
  containers : table of types.dt_lua_view_t => [ types.dt_ui_container_t, int ],  
  widget : types.lua_widget,  
  view_enter : function,  
  view_leave : function  
)
```

Register a new lib object. A lib is a graphical element of darktable's user interface

**plugin\_name**

string

A unique name for your library

**name**

string

A user-visible name for your library

**expandable**

boolean

whether this lib should be expandable or not

**resettable**

boolean

whether this lib has a reset button or not

**containers**

table of types.dt\_lua\_view\_t => [ types.dt\_ui\_container\_t, int ]

A table associating to each view containing the lib the corresponding container and position

**widget**

types.lua\_widget

The widget to display in the lib

**view\_enter**

```
self:function(  
  old_view : types.dt_lua_view_t,
```

```
new_view : types.dt_lua_view_t
)
```

A callback called when a view displaying the lib is entered

self

```
types.dt_lua_lib_t
```

The lib on which the callback is called

old\_view

```
types.dt_lua_view_t
```

The view that we are leaving

new\_view

```
types.dt_lua_view_t
```

The view that we are entering

view\_leave

```
self:function(
  old_view : types.dt_lua_view_t,
  new_view : types.dt_lua_view_t
)
```

A callback called when leaving a view displaying the lib

self

```
types.dt_lua_lib_t
```

The lib on which the callback is called

old\_view

```
types.dt_lua_view_t
```

The view that we are leaving

new\_view

```
types.dt_lua_view_t
```

The view that we are entering

## 1.7. darktable.films

A table containing all the film objects in the database.

### 1.7.1. darktable.films.#

```
types.dt_lua_film_t
```

Each film has a numeric entry in the database.

### 1.7.2. darktable.films.new

```
function(  
    directory : string  
) : types.dt_lua_film_t
```

Creates a new empty film

see `darktable.database.import` to import a directory with all its images and to add images to a film

directory

string

The directory that the new film will represent. The directory must exist

return

types.dt\_lua\_film\_t

The newly created film, or the existing film if the directory is already imported

### 1.7.3. darktable.films.delete

see `types.dt_lua_film_t.delete`

## 1.8. darktable.new\_format

```
function(  
    type : string  
) : types.dt_imageio_module_format_t
```

Creates a new format object to export images

type

string

The type of format object to create, one of :

- copy
- exr
- j2k
- jpeg
- pdf
- pfm
- png
- ppm
- tiff
- webp

*return*

`types.dt_imageio_module_format_t`

The newly created object. Exact type depends on the type passed

## 1.9. **darktable.new\_storage**

```
function(  
  type : string  
) : types.dt_imageio_module_storage_t
```

Creates a new storage object to export images

*type*

`string`

The type of storage object to create, one of :

- disk
  - email
  - gallery
  - latex
  - piwigo
- (Other, lua-defined, storage types may appear.)

*return*

`types.dt_imageio_module_storage_t`

The newly created object. Exact type depends on the type passed

## 1.10. **darktable.new\_widget**

```
function(  
  type : string,  
  ... : variable  
) : types.lua_widget
```

Creates a new widget object to display in the UI

*type*

`string`

The type of storage object to create, one of :

- box
- button
- check\_button
- combobox

- container
- entry
- file\_chooser\_button
- label
- section\_label
- separator
- slider
- stack
- text\_view

...

variable

Extra parameters, exact value are documented with each type

*return*

`types.lua_widget`

The newly created object. Exact type depends on the type passed

## 1.11. darktable.gui

This subtable contains function and data to manipulate the darktable user interface with Lua.

Most of these function won't do anything if the GUI is not enabled (i.e you are using the command line version darktable-cli instead of darktable).

### 1.11.1. darktable.gui.action\_images

table

A table of `types.dt_lua_image_t` on which the user expects UI actions to happen.

It is based on both the hovered image and the selection and is consistent with the way darktable works.

It is recommended to use this table to implement Lua actions rather than `darktable.gui.hovered` or `darktable.gui.selection` to be consistent with darktable's GUI.

### 1.11.2. darktable.gui.hovered

`types.dt_lua_image_t`

The image under the cursor or nil if no image is hovered.

### 1.11.3. darktable.gui.selection

function(

```
[selection : table of types.dt_lua_image_t]
) : table of types.dt_lua_image_t
```

Allows to change the set of selected images.

Attributes:      • *implicit\_yield*

[selection]

table of types.dt\_lua\_image\_t

A table of images which will define the selected images. If this parameter is not given the selection will be untouched. If an empty table is given the selection will be emptied.

*return*

table of types.dt\_lua\_image\_t

A table containing the selection as it was before the function was called.

#### 1.11.4. darktable.gui.current\_view

```
function(
  [view : types.dt_lua_view_t]
) : types.dt_lua_view_t
```

Allows to change the current view.

[view]

types.dt\_lua\_view\_t

The view to switch to. If empty the current view is unchanged

*return*

types.dt\_lua\_view\_t

the current view

#### 1.11.5. darktable.gui.panel\_visible

```
function(
  panel : types.dt_ui_panel_t
) : boolean
```

Determines if the specified panel is visible.

panel

types.dt\_ui\_panel\_t

The panel to check.

*return*

boolean

true if the panel is visible, false if not

#### **1.11.6. darktable.gui.panel\_hide**

```
function(  
  panel : types.dt_ui_panel_t  
)
```

Hides the specified panel.

*panel*

`types.dt_ui_panel_t`

The panel to hide.

#### **1.11.7. darktable.gui.panel\_show**

```
function(  
  panel : types.dt_ui_panel_t  
)
```

Shows the specified panel.

*panel*

`types.dt_ui_panel_t`

The panel to show.

#### **1.11.8. darktable.gui.panel\_hide\_all**

```
function(  
)
```

Hide all panels.

#### **1.11.9. darktable.gui.panel\_show\_all**

```
function(  
)
```

Show all panels.

#### **1.11.10. darktable.gui.panel\_get\_size**

```
function(  
  panel : types.dt_ui_panel_t  
)int
```

Gets the size in pixels of the specified panel. This only works for the left, right, and bottom panels.

*panel*

`types.dt_ui_panel_t`

The panel to get the size of.

### 1.11.11. darktable.gui.panel\_set\_size

```
function(  
  panel : types.dt_ui_panel_t  
  size : int  
)
```

Sets the size in pixels of the specified panel. This only works for the left, right, and bottom panels.

panel

types.dt\_ui\_panel\_t

The panel to set the size of.

size

int>

The size to set the panel to.

### 1.11.12. darktable.gui.create\_job

```
function(  
  text : string,  
  [percentage : boolean],  
  [cancel_callback : function]  
) : types.dt_lua_backgroundjob_t
```

Create a new progress\_bar displayed in darktable.gui.libs.backgroundjobs

text

string

The text to display in the job entry

[percentage]

boolean

Should a progress bar be displayed

[cancel\_callback]

```
function(  
  job : types.dt_lua_backgroundjob_t  
)
```

A function called when the cancel button for that job is pressed

note that the job won't be destroyed automatically. You need to set types.dt\_lua\_backgroundjob\_t.valid to false for that

job

types.dt\_lua\_backgroundjob\_t



The job who is being cancelled

*return*

`types.dt_lua_backgroundjob_t`

The newly created job object

### 1.11.13. **darktable.gui.views**

The different views in darktable

#### **darktable.gui.views.map**

The map view

Attributes:      • *has\_tostring*  
                  • *parent* : `types.dt_lua_view_t`

##### **darktable.gui.views.map.latitude**

number

The latitude of the center of the map

Attributes:      • *write*

##### **darktable.gui.views.map.longitude**

number

The longitude of the center of the map

Attributes:      • *write*

##### **darktable.gui.views.map.zoom**

number

The current zoom level of the map

Attributes:      • *write*

#### **darktable.gui.views.darkroom**

The darkroom view

Attributes:      • *has\_tostring*  
                  • *parent* : `types.dt_lua_view_t`

##### **darktable.gui.views.darkroom.display\_image**

```
function(  
  [image : types.dt_lua_image_t]  
) : types.dt_lua_image_t
```

Display an image in darkroom view.

[image]

`types.dt_lua_image_t`

The image to be displayed. If the image is not given, nothing will be changed.

*return*

`types.dt_lua_image_t`

The image currently displayed.

## **darktable.gui.views.lighttable**

The lighttable view

Attributes:

- *has\_tostring*
- *parent* : `types.dt_lua_view_t`

### **darktable.gui.views.lighttable.is\_image\_visible**

```
function(  
    image : types.dt_lua_image_t  
) : types.dt_lua_image_t
```

Check if the image is visible in lighttable view. The lighttable must be in file manager or zoomable mode.

image

`types.dt_lua_image_t`

The image to be checked.

*return*

boolean

True if the image is displayed. False if the image is partially displayed or not displayed.

### **darktable.gui.views.lighttable.set\_image\_visible**

```
function(  
    image : types.dt_lua_image_t  
) : types.dt_lua_image_t
```

Set the image visible in lighttable view. The lighttable must be in file manager or zoomable mode.

image

`types.dt_lua_image_t`

The image to set visible.

*return*

int

An error is returned if no image is specified.

### **darktable.gui.views.tethering**

The tethering view

Attributes:

- *has\_tostring*
- *parent* : types.dt\_lua\_view\_t

### **darktable.gui.views.slideshow**

The slideshow view

Attributes:

- *has\_tostring*
- *parent* : types.dt\_lua\_view\_t

### **darktable.gui.views.print**

The print view

Attributes:

- *has\_tostring*
- *parent* : types.dt\_lua\_view\_t

### **1.11.14. darktable.gui.libs**

This table allows to reference all lib objects

lib are the graphical blocks within each view.

To quickly figure out what lib is what, you can use the following code which will make a given lib blink.

```
local tested_module="global_toolbox"
dt.gui.libs[tested_module].visible=false
coroutine.yield("WAIT_MS",2000)
while true do
  dt.gui.libs[tested_module].visible = not dt.gui.libs[tested_module].visible
  coroutine.yield("WAIT_MS",2000)
end
```

### **darktable.gui.libs.snapshots**

The UI element that manipulates snapshots in darkroom

Attributes:

- *has\_tostring*
- *parent* : types.dt\_lua\_lib\_t

#### **darktable.gui.libs.snapshots.ratio**

number

The place in the screen where the line separating the snapshot is. Between 0 and 1

Attributes:      • *write*

#### **darktable.gui.libs.snapshots.direction**

`types.snapshot_direction_t`

The direction of the snapshot overlay

Attributes:      • *write*

#### **darktable.gui.libs.snapshots.#**

`types.dt_lua_snapshot_t`

The different snapshots for the image

#### **darktable.gui.libs.snapshots.selected**

`types.dt_lua_snapshot_t`

The currently selected snapshot

#### **darktable.gui.libs.snapshots.take\_snapshot**

```
function(  
)
```

Take a snapshot of the current image and add it to the UI

The snapshot file will be generated at the next redraw of the main window

#### **darktable.gui.libs.snapshots.max\_snapshot**

`number`

The maximum number of snapshots

### **darktable.gui.libs.collect**

The collection UI element that allows to filter images by collection

Attributes:      • *has\_tostring*  
                 • *parent* : `types.dt_lua_lib_t`

#### **darktable.gui.libs.collect.filter**

```
function(  
  [rules : array of types.dt_lib_collect_params_rule_t]  
) : array of types.dt_lib_collect_params_rule_t
```

Allows to get or change the list of visible images

Attributes:      • *implicit\_yield*

[*rules*]

`array of types.dt_lib_collect_params_rule_t`

A table of rules describing the filter. These rules will be applied after this call

*return*

`array of types.dt_lib_collect_params_rule_t`

The rules that were applied before this call.

#### **darktable.gui.libs.collect.new\_rule**

```
function(  
  ) : types.dt_lib_collect_params_rule_t
```

Returns a newly created rule object

*return*

`types.dt_lib_collect_params_rule_t`

The newly created rule

#### **darktable.gui.libs.import**

The buttons to start importing images

Attributes:

- *has\_tostring*
- *parent* : `types.dt_lua_lib_t`

#### **darktable.gui.libs.import.register\_widget**

```
function(  
  widget : types.lua_widget  
)
```

Add a widget in the option expander of the import dialog

*widget*

`types.lua_widget`

The widget to add to the dialog. The reset callback of the widget will be called whenever the dialog is opened

#### **darktable.gui.libs.styles**

The style selection menu

Attributes:

- *has\_tostring*
- *parent* : `types.dt_lua_lib_t`

#### **darktable.gui.libs.metadata\_view**

The widget displaying metadata about the current image

Attributes:

- *has\_tostring*
- *parent* : `types.dt_lua_lib_t`

### **darktable.gui.libs.metadata\_view.register\_info**

```
function(  
  name : string,  
  callback : function  
)
```

Register a function providing extra info to display in the widget

**name**

string

The name displayed for the new information

**callback**

```
function(  
  image : types.dt_lua_image_t  
) : string
```

The function providing the info

**image**

types.dt\_lua\_image\_t

The image to analyze

*return*

string

The extra information to display

### **darktable.gui.libs.metadata**

The widget allowing modification of metadata fields on the current image

Attributes:

- *has\_tostring*
- *parent* : types.dt\_lua\_lib\_t

### **darktable.gui.libs.hinter**

The small line of text at the top of the UI showing the number of selected images

Attributes:

- *has\_tostring*
- *parent* : types.dt\_lua\_lib\_t

### **darktable.gui.libs.filmstrip**

The filmstrip at the bottom of some views

Attributes:

- *has\_tostring*
- *parent* : types.dt\_lua\_lib\_t

### **darktable.gui.libs.viewswitcher**

The labels allowing to switch view

- Attributes:
- *has\_tostring*
  - *parent* : types.dt\_lua\_lib\_t

### **darktable.gui.libs.darktable\_label**

The darktable logo in the upper left corner

- Attributes:
- *has\_tostring*
  - *parent* : types.dt\_lua\_lib\_t

### **darktable.gui.libs.tagging**

The tag manipulation UI

- Attributes:
- *has\_tostring*
  - *parent* : types.dt\_lua\_lib\_t

### **darktable.gui.libs.geotagging**

The geotagging time synchronisation UI

- Attributes:
- *has\_tostring*
  - *parent* : types.dt\_lua\_lib\_t

### **darktable.gui.libs.recentcollect**

The recent collection UI element

- Attributes:
- *has\_tostring*
  - *parent* : types.dt\_lua\_lib\_t

### **darktable.gui.libs.global\_toolbox**

The common tools to all view (settings, grouping...)

- Attributes:
- *has\_tostring*
  - *parent* : types.dt\_lua\_lib\_t

#### **darktable.gui.libs.global\_toolbox.grouping**

boolean

The current status of the image grouping option

- Attributes:
- *write*

#### **darktable.gui.libs.global\_toolbox.show\_overlays**

boolean

the current status of the image overlays option

Attributes:      • *write*

### **darktable.gui.libs.filter**

The image-filter menus at the top of the UI

Attributes:      • *has\_tostring*  
                 • *parent* : types.dt\_lua\_lib\_t

#### **darktable.gui.libs.filter.sort**

```
function(  
  [sort : types.dt_collection_sort_t]  
) : types.dt_collection_sort_t
```

Change the collection sort field.

[sort]

types.dt\_collection\_sort\_t

The new field to sort by. If empty the current sort field is unchanged

*return*

types.dt\_collection\_sort\_t

The current sort field.

#### **darktable.gui.libs.filter.sort\_order**

```
function(  
  [order : types.dt_collection_sort_order_t]  
) : types.dt_collection_sort_order_t
```

Change the collection sort order.

[order]

types.dt\_collection\_sort\_order\_t

The order to sort by. If empty the current sort order is unchanged.

*return*

types.dt\_collection\_sort\_order\_t

The current sort order.

#### **darktable.gui.libs.filter.rating**

```
function(  
  [rating : types.dt_collection_filter_t]  
) : types.dt_collection_filter_t
```



Change the collection rating filter.

[rating]

`types.dt_collection_filter_t`

The new rating field to filter by. If empty the current rating field is unchanged.

*return*

`types.dt_collection_filter_t`

The current rating field.

#### **darktable.gui.libs.filter.rating\_comparator**

```
function(  
  [comparator : types.dt_collection_rating_comperator_t]  
) : types.dt_collection_rating_comperator_t
```

Change the collection filter comparison field.

[comparator]

`types.dt_collection_rating_comperator_t`

The new comparison field to filter the rating by. If empty the current rating comparison field is unchanged

*return*

`types.dt_collection_rating_comperator_t`

The current rating comparison field.

#### **darktable.gui.libs.ratings**

The stars to set the rating of an image

Attributes:

- *has\_tostring*
- *parent* : `types.dt_lua_lib_t`

#### **darktable.gui.libs.select**

The buttons that allow to quickly change the selection

Attributes:

- *has\_tostring*
- *parent* : `types.dt_lua_lib_t`

#### **darktable.gui.libs.select.register\_selection**

```
function(  
  label : string,  
  callback : function,  
  [tooltip : string]  
)
```

Add a new button and call a callback when it is clicked

label

string

The label to display on the button

callback

```
function(  
    event : string,  
    images : table of types.dt_lua_image_t  
) : table of types.dt_lua_image_t
```

The function to call when the button is pressed

event

string

The name of the button that was pressed

images

table of types.dt\_lua\_image\_t

The images in the current collection. This is the same content as `darktable.collection`

return

table of types.dt\_lua\_image\_t

The images to set the selection to

[tooltip]

string

The tooltip to use on the new button

## **darktable.gui.libs.colorlabels**

The color buttons that allow to set labels on an image

Attributes:

- *has\_tostring*
- *parent* : types.dt\_lua\_lib\_t

## **darktable.gui.libs.lighttable\_mode**

The navigation and zoom level UI in lighttable

Attributes:

- *has\_tostring*
- *parent* : types.dt\_lua\_lib\_t

## **darktable.gui.libs.lighttable\_mode.layout**

```
function(
  [layout : types.dt_lighttable_layout_t]
) : types.dt_lighttable_layout_t
```

Allows to change the lighttable layout.

[layout]

types.dt\_lighttable\_layout\_t

The layout to switch to. If empty the current layout is unchanged

*return*

types.dt\_lighttable\_layout\_t

the current layout

#### **darktable.gui.libs.lighttable\_mode.zoom\_level**

```
function(
  [level : number]
) : number
```

Change the lighttable zoom level.

[level]

number

The zoom level to switch to. If empty the current zoom level is unchanged

*return*

number

the current zoom level

#### **darktable.gui.libs.copy\_history**

The UI element that manipulates history

Attributes:

- *has\_tostring*
- *parent* : types.dt\_lua\_lib\_t

#### **darktable.gui.libs.image**

The UI element that manipulates the current images

Attributes:

- *has\_tostring*
- *parent* : types.dt\_lua\_lib\_t

#### **darktable.gui.libs.image.register\_action**

```
function(
  label : string,
```

```

    callback : function,
    [tooltip : string]
)

```

Add a new button and call a callback when it is clicked

label

```
string
```

The label to display on the button

callback

```

function(
    event : string,
    images : table of types.dt_lua_image_t
)

```

The function to call when the button is pressed

event

```
string
```

The name of the button that was pressed

images

```
table of types.dt_lua_image_t
```

The images to act on when the button was clicked

[tooltip]

```
string
```

The tooltip to use on the new button

### **darktable.gui.libs.modulegroups**

The icons describing the different iop groups

Attributes:

- *has\_tostring*
- *parent* : types.dt\_lua\_lib\_t

### **darktable.gui.libs.module\_toolbox**

The tools on the bottom line of the UI (overexposure)

Attributes:

- *has\_tostring*
- *parent* : types.dt\_lua\_lib\_t

### **darktable.gui.libs.session**

The session UI when tethering

- Attributes:
- *has\_tostring*
  - *parent* : types.dt\_lua\_lib\_t

### **darktable.gui.libs.histogram**

The histogram widget

- Attributes:
- *has\_tostring*
  - *parent* : types.dt\_lua\_lib\_t

### **darktable.gui.libs.export**

The export menu

- Attributes:
- *has\_tostring*
  - *parent* : types.dt\_lua\_lib\_t

### **darktable.gui.libs.history**

The history manipulation menu

- Attributes:
- *has\_tostring*
  - *parent* : types.dt\_lua\_lib\_t

### **darktable.gui.libs.colorpicker**

The colorpicker menu

- Attributes:
- *has\_tostring*
  - *parent* : types.dt\_lua\_lib\_t

### **darktable.gui.libs.navigation**

The full image preview to allow navigation

- Attributes:
- *has\_tostring*
  - *parent* : types.dt\_lua\_lib\_t

### **darktable.gui.libs.masks**

The masks window

- Attributes:
- *has\_tostring*
  - *parent* : types.dt\_lua\_lib\_t

### **darktable.gui.libs.view\_toolbox**

- Attributes:
- *has\_tostring*

- *parent* : types.dt\_lua\_lib\_t

### **darktable.gui.libs.live\_view**

The liveview window

- Attributes:
- *has\_tostring*
  - *parent* : types.dt\_lua\_lib\_t

### **darktable.gui.libs.map\_settings**

The map setting window

- Attributes:
- *has\_tostring*
  - *parent* : types.dt\_lua\_lib\_t

### **darktable.gui.libs.camera**

The camera selection UI

- Attributes:
- *has\_tostring*
  - *parent* : types.dt\_lua\_lib\_t

### **darktable.gui.libs.location**

The location ui

- Attributes:
- *has\_tostring*
  - *parent* : types.dt\_lua\_lib\_t

### **darktable.gui.libs.backgroundjobs**

The window displaying the currently running jobs

- Attributes:
- *has\_tostring*
  - *parent* : types.dt\_lua\_lib\_t

### **darktable.gui.libs.print\_settings**

The settings window in the print view

- Attributes:
- *has\_tostring*
  - *parent* : types.dt\_lua\_lib\_t

## **1.12. darktable.guides**

table

Guide lines to overlay over an image in crop and rotate.

All guides are clipped to the drawing area.

### 1.12.1. darktable.guides.register\_guide

```
function(  
  name : string,  
  draw_callback : function,  
  [gui_callback : function]  
)
```

Register a new guide.

**name**

string

The name of the guide to show in the GUI.

**draw\_callback**

```
function(  
  cr : types.dt_lua_cairo_t,  
  x : float,  
  y : float,  
  width : float,  
  height : float,  
  zoom_scale : float  
)
```

The function to call to draw the guide lines. The drawn lines will be stroked by darktable.

THIS IS RUNNING IN THE GUI THREAD AND HAS TO BE FAST!

**cr**

types.dt\_lua\_cairo\_t

The cairo object used for drawing.

**x**

float

The x coordinate of the top left corner of the drawing area.

**y**

float

The y coordinate of the top left corner of the drawing area.

**width**

float

The width of the drawing area.

**height**

float

The height of the drawing area.

zoom\_scale

float

The current zoom\_scale. Only needed when setting the line thickness.

[gui\_callback]

function

A function returning a widget to show when the guide is selected. It takes no arguments.

## 1.13. darktable.tags

Allows access to all existing tags.

### 1.13.1. darktable.tags.#

types.dt\_lua\_tag\_t

Each existing tag has a numeric entry in the tags table - use ipairs to iterate over them.

### 1.13.2. darktable.tags.create

```
function(  
  name : string  
)
```

Creates a new tag and return it. If the tag exists return the existing tag.

name

string

The name of the new tag.

### 1.13.3. darktable.tags.find

```
function(  
  name : string  
) : types.dt_lua_tag_t
```

Returns the tag object or nil if the tag doesn't exist.

name

string

The name of the tag to find.

*return*

types.dt\_lua\_tag\_t

The tag object or nil.



#### 1.13.4. darktable.tags.delete

```
function(  
  tag : types.dt_lua_tag_t  
)
```

Deletes the tag object, detaching it from all images.

tag

types.dt\_lua\_tag\_t

The tag to be deleted.

#### 1.13.5. darktable.tags.attach

```
function(  
  tag : types.dt_lua_tag_t,  
  image : types.dt_lua_image_t  
)
```

Attach a tag to an image; the order of the parameters can be reversed.

tag

types.dt\_lua\_tag\_t

The tag to be attached.

image

types.dt\_lua\_image\_t

The image to attach the tag to.

#### 1.13.6. darktable.tags.detach

```
function(  
  tag : types.dt_lua_tag_t,  
  image : types.dt_lua_image_t  
)
```

Detach a tag from an image; the order of the parameters can be reversed.

tag

types.dt\_lua\_tag\_t

The tag to be detached.

image

types.dt\_lua\_image\_t

The image to detach the tag from.

#### 1.13.7. darktable.tags.get\_tags

```
function(  
  tag : types.dt_lua_tag_t,  
  image : types.dt_lua_image_t  
)
```

```
image : types.dt_lua_image_t  
) : table of types.dt_lua_tag_t
```

Gets all tags attached to an image.

image

```
types.dt_lua_image_t
```

The image to get the tags from.

*return*

```
table of types.dt_lua_tag_t
```

A table of tags that are attached to the image.

### 1.13.8. darktable.tags.get\_tagged\_images

```
function(  
  tag : types.dt_lua_tag_t  
) : table of types.dt_lua_image_t
```

Gets all images with the attached tag.

tag

```
types.dt_lua_tag_t
```

The tag to search images for.

*return*

```
table of types.dt_lua_image_t
```

A table of images that have the tag attached.

## 1.14. darktable.configuration

table

This table regroups values that describe details of the configuration of darktable.

### 1.14.1. darktable.configuration.version

string

The version number of darktable.

### 1.14.2. darktable.configuration.has\_gui

boolean

True if darktable has a GUI (launched through the main darktable command, not darktable-cli).

### 1.14.3. darktable.configuration.verbose

boolean

True if the Lua logdomain is enabled.

#### **1.14.4. darktable.configuration.tmp\_dir**

string

The name of the directory where darktable will store temporary files.

#### **1.14.5. darktable.configuration.config\_dir**

string

The name of the directory where darktable will find its global configuration objects (modules).

#### **1.14.6. darktable.configuration.cache\_dir**

string

The name of the directory where darktable will store its mipmaps.

#### **1.14.7. darktable.configuration.api\_version\_major**

number

The major version number of the lua API.

#### **1.14.8. darktable.configuration.api\_version\_minor**

number

The minor version number of the lua API.

#### **1.14.9. darktable.configuration.api\_version\_patch**

number

The patch version number of the lua API.

#### **1.14.10. darktable.configuration.api\_version\_suffix**

string

The version suffix of the lua API.

#### **1.14.11. darktable.configuration.api\_version\_string**

string

The version description of the lua API. This is a string compatible with the semantic versioning convention

#### **1.14.12. darktable.configuration.running\_os**

string

The name of the Operating system darktable is currently running on

#### **1.14.13. darktable.configuration.check\_version**

```
function(
  module_name : string,
  ... : table...
)
```

Check that a module is compatible with the running version of darktable

Add the following line at the top of your module :

```
darktable.configuration.check( ... , {M,m,p} , {M2,m2,p2} )
```

To document that your module has been tested with API version M.m.p and M2.m2.p2.

This will raise an error if the user is running a released version of DT and a warning if he is running a development version

(the ... here will automatically expand to your module name if used at the top of your script

module\_name

string

The name of the module to report on error

...

table...

Tables of API versions that are known to work with the script

## 1.15. darktable.preferences

table

Lua allows you to manipulate preferences. Lua has its own namespace for preferences and you can't access nor write normal darktable preferences.

Preference handling functions take a `_script_` parameter. This is a string used to avoid name collision in preferences (i.e namespace). Set it to something unique, usually the name of the script handling the preference.

Preference handling functions can't guess the type of a parameter. You must pass the type of the preference you are handling.

Note that the directory, enum, lua and file type preferences are stored internally as string. The user can only select valid values, but a lua script can set it to any string

### 1.15.1. darktable.preferences.register

```
function(
  script : string,
  name : string,
  type : types.lua_pref_type,
  label : string,
  tooltip : string,
  [default : depends on type],
  [min : int or float],
  [max : int or float],
```

```

[step : float],
[values : string...],
[widget : types.lua_widget],
[set_callback : function]
)

```

Creates a new preference entry in the Lua tab of the preference screen. If this function is not called the preference can't be set by the user (you can still read and write invisible preferences).

**script**

string

Invisible prefix to guarantee unicity of preferences.

**name**

string

A unique name used with the script part to identify the preference.

**type**

types.lua\_pref\_type

The type of the preference - one of the string values described above.

**label**

string

The label displayed in the preference screen.

**tooltip**

string

The tooltip to display in the preference menu.

**[default]**

depends on type

Default value to use when not set explicitly or by the user.

For the enum type of pref, this is mandatory

**[min]**

int or float

Minimum value (integer and float preferences only).

**[max]**

int or float

Maximum value (integer and float preferences only).

[step]

float

Step of the spinner (float preferences only).

[values]

string...

Other allowed values (enum preferences only)

[widget]

types.lua\_widget

The widget to use in preference(lua preferences only)

[set\_callback]

```
function(  
  widget : types.lua_widget  
)
```

A function called when the widget needs to be updated from the preference

widget

types.lua\_widget

The widget to update

### 1.15.2. darktable.preferences.read

```
function(  
  script : string,  
  name : string,  
  type : types.lua_pref_type  
) : depends on type
```

Reads a value from a Lua preference.

script

string

Invisible prefix to guarantee unicity of preferences.

name

string

The name of the preference displayed in the preference screen.

type

types.lua\_pref\_type

The type of the preference.

*return*

depends on type

The value of the preference.

### 1.15.3. darktable.preferences.write

```
function(  
  script : string,  
  name : string,  
  type : types.lua_pref_type,  
  value : depends on type  
)
```

Writes a value to a Lua preference.

*script*

string

Invisible prefix to guarantee unicity of preferences.

*name*

string

The name of the preference displayed in the preference screen.

*type*

types.lua\_pref\_type

The type of the preference.

*value*

depends on type

The value to set the preference to.

## 1.16. darktable.styles

This pseudo table allows you to access and manipulate styles.

### 1.16.1. darktable.styles.#

types.dt\_style\_t

Each existing style has a numeric index; you can iterate them using ipairs.

### 1.16.2. darktable.styles.create

```
function(  
  image : types.dt_lua_image_t,  
  name : string,  
  description : string  
) : types.dt_style_t
```

Create a new style based on an image.

image

`types.dt_lua_image_t`

The image to create the style from.

name

`string`

The name to give to the new style.

description

`string`

The description of the new style.

*return*

`types.dt_style_t`

The new style object.

### 1.16.3. **darktable.styles.delete**

```
function(  
  style : types.dt_style_t  
)
```

Deletes an existing style.

style

`types.dt_style_t`

the style to delete

### 1.16.4. **darktable.styles.duplicate**

```
function(  
  style : types.dt_style_t,  
  name : string,  
  description : string  
) : types.dt_style_t
```

Create a new style based on an existing style.

style

`types.dt_style_t`

The style to base the new style on.

name

`string`



The new style's name.

description

string

The new style's description.

*return*

types.dt\_style\_t

The new style object.

### 1.16.5. darktable.styles.apply

```
function(  
  style : types.dt_style_t,  
  image : types.dt_lua_image_t  
)
```

Apply a style to an image. The order of parameters can be inverted.

style

types.dt\_style\_t

The style to use.

image

types.dt\_lua\_image\_t

The image to apply the style to.

### 1.16.6. darktable.styles.import

```
function(  
  filename : string  
)
```

Import a style from an external .dtstyle file

filename

string

The file to import

### 1.16.7. darktable.styles.export

```
function(  
  style : types.dt_style_t,  
  directory : string,  
  overwrite : boolean  
)
```

Export a style to an external .dtstyle file

style

`types.dt_style_t`

The style to export

directory

`string`

The directory to export to

overwrite

`boolean`

Is overwriting an existing file allowed

## 1.17. darktable.database

Allows to access the database of images. Note that duplicate images (images with the same RAW but different XMP) will appear multiple times with different duplicate indexes. Also note that all images are here. This table is not influenced by any GUI filtering (collections, stars etc...).

### 1.17.1. darktable.database.#

`types.dt_lua_image_t`

Each image in the database appears with a numerical index; you can iterate them using `ipairs`.

### 1.17.2. darktable.database.duplicate

```
function(  
  image : types.dt_lua_image_t  
) : types.dt_lua_image_t
```

Creates a duplicate of an image and returns it.

image

`types.dt_lua_image_t`

the image to duplicate

*return*

`types.dt_lua_image_t`

The new image object.

### 1.17.3. darktable.database.import

```
function(  
  location : string  
) : types.dt_lua_image_t
```

Imports new images into the database.

location

string

The filename or directory to import images from. NOTE: If the images are set to be imported recursively in preferences only the toplevel film is returned (the one whose path was given as a parameter). NOTE2: If the parameter is a directory the call is non-blocking; the film object will not have the newly imported images yet. Use a post-import-film filtering on that film to react when images are actually imported.

*return*

types.dt\_lua\_image\_t

The created image if an image is imported or the toplevel film object if a film was imported.

#### 1.17.4. darktable.database.move\_image

```
function(  
  image : types.dt_lua_image_t,  
  film : types.dt_lua_film_t,  
  [newname : string]  
)
```

Physically moves an image (and all its duplicates) to another film, and/or renames the image file.

This will move the image file, the related XMP and all XMP for the duplicates to the directory of the new film and rename them as specified.

Note that the order of the two required parameters is not relevant.

image

types.dt\_lua\_image\_t

The image to move

film

types.dt\_lua\_film\_t

The film to move to. To rename a file within the same film, set this to the image's current film.

newname

string

(Optional) If provided, rename the file with this new name. Must not contain any path separator characters.

#### 1.17.5. darktable.database.copy\_image

```
function(  
  image : types.dt_lua_image_t,  
  film : types.dt_lua_film_t,
```

```
[newname : string]  
) : types.dt_lua_image_t
```

Physically copies an image to another film.

This will copy the image file and the related XMP to the directory of the new film and rename them as specified.

If there is already a file with the same name as the image file, it will create a duplicate from that file instead.

Note that the order of the two required parameters is not relevant.

image

```
types.dt_lua_image_t
```

The image to copy

film

```
types.dt_lua_film_t
```

The film to copy to. To make a copy of a file within the same film, set this to the image's current film.

newname

```
string
```

*(Optional)* If provided, give the copy this new filename. Must not contain any path separator characters.

*return*

```
types.dt_lua_image_t
```

The new image

#### 1.17.6. darktable.database.delete

see `types.dt_lua_image_t.delete`

### 1.18. darktable.collection

Allows to access the currently worked on images, i.e the ones selected by the collection lib. Filtering (rating etc) does not change that collection.

#### 1.18.1. darktable.collection.#

```
types.dt_lua_image_t
```

Each image in the collection appears with a numerical index; you can iterate them using `ipairs`.

### 1.19. darktable.control

This table contain function to manipulate the control flow of lua programs. It provides ways to do background jobs and other related functions

### 1.19.1. darktable.control.ending

boolean

TRUE when darktable is terminating

Use this variable to detect when you should finish long running jobs

### 1.19.2. darktable.control.dispatch

```
function(  
  function : function,  
  ... : anything  
)
```

Runs a function in the background. This function will be run at a later point, after luarc has finished running. If you do a loop in such a function, please check darktable.control.ending in your loop to finish the function when DT exits

function

function

The call to dispatch

...

anything

extra parameters to pass to the function

### 1.19.3. darktable.control.sleep

```
function(  
  delay : int  
)
```

Suspends execution while not blocking darktable

delay

int

The delay in millisecond to sleep

### 1.19.4. darktable.control.execute

```
function(  
  command : string  
) : int
```

Run a command in a shell while not blocking darktable

command

string

The command to run, as in 'sh -c'

*return*

int

The result of the system call

### **1.19.5. darktable.control.read**

```
function(  
  file : file  
)
```

Block until a file is readable while not blocking darktable

*This function is not available on Windows builds*

file

file

The file object to wait for

## **1.20. darktable.gettext**

table

This table contains functions related to translating lua scripts

### **1.20.1. darktable.gettext.gettext**

```
function(  
  msgid : string  
) : string
```

Translate a string using the darktable textdomain

msgid

string

The string to translate

*return*

string

The translated string

### **1.20.2. darktable.gettext.dgettext**

```
function(  
  domainname : string,  
  msgid : string  
) : string
```

Translate a string using the specified textdomain

domainname

string

The domain to use for that translation

*msgid*

string

The string to translate

*return*

string

The translated string

### 1.20.3. **darktablegettext.ngettext**

```
function(  
  msgid : string,  
  msgid_plural : string,  
  n : int  
) : string
```

Translate a string depending on the number of objects using the darktable textdomain

*msgid*

string

The string to translate

*msgid\_plural*

string

The string to translate in plural form

*n*

int

The number of objects

*return*

string

The translated string

### 1.20.4. **darktablegettext.dgettext**

```
function(  
  domainname : string,  
  msgid : string,  
  msgid_plural : string,  
  n : int  
) : string
```

Translate a string depending on the number of objects using the specified textdomain

domainname

string

The domain to use for that translation

msgid

string

The string to translate

msgid\_plural

string

The string to translate in plural form

n

int

The number of objects

*return*

string

The translated string

#### 1.20.5. darktable.gettext.bindtextdomain

```
function(  
  domainname : string,  
  dirname : string  
)
```

Tell gettext where to find the .mo file translating messages for a particular domain

domainname

string

The domain to use for that translation

dirname

string

The base directory to look for the file. The file should be placed in *dirname/locale name/LC\_MESSAGES/domain.mo*

#### 1.21. darktable.debug

table

This section must be activated separately by calling `require "darktable.debug"`

##### 1.21.1. darktable.debug.dump



```
function(
    object : anything,
    [name : string],
    [known : table]
) : string
```

This will return a string describing everything Lua knows about an object, used to know what an object is. This function is recursion-safe and can be used to dump \_G if needed.

**object**

anything

The object to dump.

**[name]**

string

A name to use for the object.

**[known]**

table

A table of object,string pairs. Any object in that table will not be dumped, the string will be printed instead.

defaults to darktable.debug.known if not set

**return**

string

A string containing a text description of the object - can be very long.

### 1.21.2. darktable.debug.debug

boolean

Initialized to false; set it to true to also dump information about metatables.

### 1.21.3. darktable.debug.max\_depth

number

Initialized to 10; The maximum depth to recursively dump content.

### 1.21.4. darktable.debug.known

table

A table containing the default value of darktable.debug.dump.known

### 1.21.5. darktable.debug.type

```
function(
    object : anything
) : string
```

Similar to the system function `type()` but it will return the real type instead of "userdata" for darktable specific objects.

**object**

`anything`

The object whose type must be reported.

***return***

`string`

A string describing the type of the object.

## 2. types

This section documents types that are specific to darktable's Lua API.

### 2.1. `types.lua_os_type`

enum

The type of OS we darktable can run on

Attributes:

- *values* :
- windows
- macos
- linux
- unix

### 2.2. `types.dt_lua_image_t`

dt\_type

Image objects represent an image in the database. This is slightly different from a file on disk since a file can have multiple developments. Note that this is the real image object; changing the value of a field will immediately change it in darktable and will be reflected on any copy of that image object you may have kept.

Attributes:

- *has\_tostring*

#### 2.2.1. `types.dt_lua_image_t.attach_tag`

see `darktable.tags.attach`

#### 2.2.2. `types.dt_lua_image_t.detach_tag`

see `darktable.tags.detach`

#### 2.2.3. `types.dt_lua_image_t.get_tags`

see `darktable.tags.get_tags`

#### 2.2.4. `types.dt_lua_image_t.create_style`

see `darktable.styles.create`

#### 2.2.5. `types.dt_lua_image_t.apply_style`

see `darktable.styles.apply`

#### 2.2.6. `types.dt_lua_image_t.duplicate`

see `darktable.database.duplicate`

#### 2.2.7. `types.dt_lua_image_t.move`

see `darktable.database.move_image`

### 2.2.8. `types.dt_lua_image_t.copy`

see `darktable.database.copy_image`

### 2.2.9. `types.dt_lua_image_t.id`

number

A unique id identifying the image in the database.

### 2.2.10. `types.dt_lua_image_t.path`

string

The file the directory containing the image.

### 2.2.11. `types.dt_lua_image_t.film`

`types.dt_lua_film_t`

The film object that contains this image.

### 2.2.12. `types.dt_lua_image_t.filename`

string

The filename of the image.

### 2.2.13. `types.dt_lua_image_t.sidecar`

string

The filename of the image's sidecar file.

### 2.2.14. `types.dt_lua_image_t.duplicate_index`

number

If there are multiple images based on a same file, each will have a unique number, starting from 0.

### 2.2.15. `types.dt_lua_image_t.publisher`

string

The publisher field of the image.

Attributes:      • *write*

### 2.2.16. `types.dt_lua_image_t.title`

string

The title field of the image.

Attributes:      • *write*

### 2.2.17. `types.dt_lua_image_t.creator`

string

The creator field of the image.

Attributes:      • *write*

#### **2.2.18. types.dt\_lua\_image\_t.rights**

string

The rights field of the image.

Attributes:      • *write*

#### **2.2.19. types.dt\_lua\_image\_t.description**

string

The description field for the image.

Attributes:      • *write*

#### **2.2.20. types.dt\_lua\_image\_t.notes**

string

The notes field for the image.

Attributes:      • *write*

#### **2.2.21. types.dt\_lua\_image\_t.version\_name**

string

The version\_name field for the image.

Attributes:      • *write*

#### **2.2.22. types.dt\_lua\_image\_t.exif\_maker**

string

The maker exif data.

Attributes:      • *write*

#### **2.2.23. types.dt\_lua\_image\_t.exif\_model**

string

The camera model used.

Attributes:      • *write*

#### **2.2.24. types.dt\_lua\_image\_t.exif\_lens**

string

The id string of the lens used.

Attributes:      • *write*

#### **2.2.25. types.dt\_lua\_image\_t.exif\_aperture**

number

The aperture saved in the exif data.

Attributes:      • *write*

#### **2.2.26. types.dt\_lua\_image\_t.exif\_exposure**

number

The exposure time of the image.

Attributes:      • *write*

#### **2.2.27. types.dt\_lua\_image\_t.exif\_focal\_length**

number

The focal length of the image.

Attributes:      • *write*

#### **2.2.28. types.dt\_lua\_image\_t.exif\_iso**

number

The iso used on the image.

Attributes:      • *write*

#### **2.2.29. types.dt\_lua\_image\_t.exif\_datetime\_taken**

string

The date and time of the image.

Attributes:      • *write*

#### **2.2.30. types.dt\_lua\_image\_t.exif\_focus\_distance**

number

The distance of the subject.

Attributes:      • *write*

#### **2.2.31. types.dt\_lua\_image\_t.exif\_crop**

number

The exif crop data.

Attributes:      • *write*

### 2.2.32. **types.dt\_lua\_image\_t.latitude**

float or nil

GPS latitude data of the image, nil if not set.

Attributes:      • *write*

### 2.2.33. **types.dt\_lua\_image\_t.longitude**

float or nil

GPS longitude data of the image, nil if not set.

Attributes:      • *write*

### 2.2.34. **types.dt\_lua\_image\_t.elevation**

float or nil

GPS altitude data of the image, nil if not set.

Attributes:      • *write*

### 2.2.35. **types.dt\_lua\_image\_t.is\_raw**

boolean

True if the image is a RAW file.

WARNING: This is a flag that gets set the first time the image is opened, either by an edit or thumbnail generation. If images are imported using `darktable.database.import()`, then `dt_lua_image_t.is_raw` is not guaranteed to be correct.

### 2.2.36. **types.dt\_lua\_image\_t.is\_ldr**

boolean

True if the image is a ldr image.

### 2.2.37. **types.dt\_lua\_image\_t.is\_hdr**

boolean

True if the image is a hdr image.

### 2.2.38. **types.dt\_lua\_image\_t.has\_txt**

boolean

True if the image has a txt sidecar file.

Attributes:      • *write*

### 2.2.39. `types.dt_lua_image_t.width`

`number`

The width of the image.

### 2.2.40. `types.dt_lua_image_t.height`

`number`

The height of the image.

### 2.2.41. `types.dt_lua_image_t.rating`

`number`

The rating of the image (-1 for rejected).

Attributes:      • *write*

### 2.2.42. `types.dt_lua_image_t.red`

`boolean`

True if the image has the corresponding colorlabel.

Attributes:      • *write*

### 2.2.43. `types.dt_lua_image_t.blue`

see `types.dt_lua_image_t.red`

### 2.2.44. `types.dt_lua_image_t.green`

see `types.dt_lua_image_t.red`

### 2.2.45. `types.dt_lua_image_t.yellow`

see `types.dt_lua_image_t.red`

### 2.2.46. `types.dt_lua_image_t.purple`

see `types.dt_lua_image_t.red`

### 2.2.47. `types.dt_lua_image_t.reset`

```
self: function(  
)
```

Removes all processing from the image, resetting it back to its original state

`self`

`types.dt_lua_image_t`

The image whose history will be deleted

### 2.2.48. `types.dt_lua_image_t.delete`



```
self:function(  
)
```

Removes an image from the database

self

types.dt\_lua\_image\_t

The image to remove

### 2.2.49. types.dt\_lua\_image\_t.group\_with

```
self:function(  
  [image : types.dt_lua_image_t]  
)
```

Puts the first image in the same group as the second image. If no second image is provided the image will be in its own group.

self

types.dt\_lua\_image\_t

The image whose group must be changed.

[image]

types.dt\_lua\_image\_t

The image we want to group with.

### 2.2.50. types.dt\_lua\_image\_t.make\_group\_leader

```
self:function(  
)
```

Makes the image the leader of its group.

self

types.dt\_lua\_image\_t

The image we want as the leader.

### 2.2.51. types.dt\_lua\_image\_t.get\_group\_members

```
self:function(  
) : table of types.dt_lua_image_t
```

Returns a table containing all types.dt\_lua\_image\_t of the group. The group leader is both at a numeric key and at the "leader" special key (so you probably want to use ipairs to iterate through that table).

self

types.dt\_lua\_image\_t

The image whose group we are querying.

*return*

table of `types.dt_lua_image_t`

A table of image objects containing all images that are in the same group as the image.

### 2.2.52. `types.dt_lua_image_t.group_leader`

`types.dt_lua_image_t`

The image which is the leader of the group this image is a member of.

### 2.2.53. `types.dt_lua_image_t.local_copy`

boolean

True if the image has a copy in the local cache

Attributes:      • *write*

### 2.2.54. `types.dt_lua_image_t.drop_cache`

```
self:function(  
)
```

drops the cached version of this image.

This function should be called if an image is modified out of darktable to force DT to re-generate the thumbnail

darktable will regenerate the thumbnail by itself when it is needed

self

`types.dt_lua_image_t`

The image whose cache must be dropped.

## 2.3. `types.dt_imageio_module_format_t`

`dt_type`

A virtual type representing all format types.

### 2.3.1. `types.dt_imageio_module_format_t.plugin_name`

string

A unique name for the plugin.

### 2.3.2. `types.dt_imageio_module_format_t.name`

string

A human readable name for the plugin.

### 2.3.3. `types.dt_imageio_module_format_t.extension`

string

The typical filename extension for that format.

#### 2.3.4. `types.dt_imageio_module_format_t.mime`

string

The mime type associated with the format.

#### 2.3.5. `types.dt_imageio_module_format_t.max_width`

number

The max width allowed for the format (0 = unlimited).

Attributes:      • *write*

#### 2.3.6. `types.dt_imageio_module_format_t.max_height`

number

The max height allowed for the format (0 = unlimited).

Attributes:      • *write*

#### 2.3.7. `types.dt_imageio_module_format_t.write_image`

```
self:function(  
  image : types.dt_lua_image_t,  
  filename : string,  
  [allow_upscale : boolean]  
) : boolean
```

Exports an image to a file. This is a blocking operation that will not return until the image is exported.

Attributes:      • *implicit\_yield*

**self**

`types.dt_imageio_module_format_t`

The format that will be used to export.

**image**

`types.dt_lua_image_t`

The image object to export.

**filename**

string

The filename to export to.

**[allow\_upscale]**

boolean

Set to true to allow upscaling of the image.

*return*

boolean

Returns true on success.

## 2.4. **types.dt\_imageio\_module\_format\_data\_png**

dt\_type

Type object describing parameters to export to png.

Attributes:      • *parent* : types.dt\_imageio\_module\_format\_t

### 2.4.1. **types.dt\_imageio\_module\_format\_data\_png.bpp**

number

The bpp parameter to use when exporting.

Attributes:      • *write*

## 2.5. **types.dt\_imageio\_module\_format\_data\_tiff**

dt\_type

Type object describing parameters to export to tiff.

Attributes:      • *parent* : types.dt\_imageio\_module\_format\_t

### 2.5.1. **types.dt\_imageio\_module\_format\_data\_tiff.bpp**

number

The bpp parameter to use when exporting.

Attributes:      • *write*

## 2.6. **types.dt\_imageio\_module\_format\_data\_exr**

dt\_type

Type object describing parameters to export to exr.

Attributes:      • *parent* : types.dt\_imageio\_module\_format\_t

### 2.6.1. **types.dt\_imageio\_module\_format\_data\_exr.compression**

string

The compression parameter to use when exporting.

Attributes:      • *write*

## 2.7. `types.dt_imageio_module_format_data_copy`

`dt_type`

Type object describing parameters to export to copy.

Attributes:      • *parent* : `types.dt_imageio_module_format_t`

## 2.8. `types.dt_imageio_module_format_data_pfm`

`dt_type`

Type object describing parameters to export to pfm.

Attributes:      • *parent* : `types.dt_imageio_module_format_t`

## 2.9. `types.dt_imageio_module_format_data_jpeg`

`dt_type`

Type object describing parameters to export to jpeg.

Attributes:      • *parent* : `types.dt_imageio_module_format_t`

### 2.9.1. `types.dt_imageio_module_format_data_jpeg.quality`

`number`

The quality to use at export time.

Attributes:      • *write*

## 2.10. `types.dt_imageio_module_format_data_ppm`

`dt_type`

Type object describing parameters to export to ppm.

Attributes:      • *parent* : `types.dt_imageio_module_format_t`

## 2.11. `types.dt_imageio_module_format_data_webp`

`dt_type`

Type object describing parameters to export to webp.

Attributes:      • *parent* : `types.dt_imageio_module_format_t`

### 2.11.1. `types.dt_imageio_module_format_data_webp.quality`

`number`

The quality to use at export time.

Attributes:      • *write*

### 2.11.2. `types.dt_imageio_module_format_data_webp.comp_type`

`types.comp_type_t`

The overall quality to use; can be one of "webp\_lossy" or "webp\_lossless".

Attributes:      • *write*

### 2.11.3. `types.dt_imageio_module_format_data_webp.hint`

`types.hint_t`

A hint on the overall content of the image.

Attributes:      • *write*

## 2.12. `types.dt_imageio_module_format_data_j2k`

`dt_type`

Type object describing parameters to export to jpeg2000.

Attributes:      • *parent* : `types.dt_imageio_module_format_t`

### 2.12.1. `types.dt_imageio_module_format_data_j2k.quality`

`number`

The quality to use at export time.

Attributes:      • *write*

### 2.12.2. `types.dt_imageio_module_format_data_j2k.bpp`

`number`

The bpp parameter to use when exporting.

Attributes:      • *write*

### 2.12.3. `types.dt_imageio_module_format_data_j2k.format`

`types.dt_imageio_j2k_format_t`

The format to use.

Attributes:      • *write*

### 2.12.4. `types.dt_imageio_module_format_data_j2k.preset`

`types.dt_imageio_j2k_preset_t`

The preset to use.

Attributes:      • *write*

## 2.13. `types.dt_imageio_module_format_data_pdf`

`dt_type`

Type object describing parameters to export to pdf.

Attributes:      • *parent*: `types.dt_imageio_module_format_t`

### 2.13.1. `types.dt_imageio_module_format_data_pdf.dpi`

`number`

The dot per inch value to use at export

Attributes:      • *write*

### 2.13.2. `types.dt_imageio_module_format_data_pdf.icc`

`boolean`

Should the images be tagged with their embedded profile

Attributes:      • *write*

### 2.13.3. `types.dt_imageio_module_format_data_pdf.border`

`string`

Empty space around the PDF images

Attributes:      • *write*

### 2.13.4. `types.dt_imageio_module_format_data_pdf.orientation`

`string`

Orientation of the pages in the document

Attributes:      • *write*

### 2.13.5. `types.dt_imageio_module_format_data_pdf.title`

`string`

The title for the document `types.dt_imageio_module_format_data_pdf.rotate:set_text([[Should the images be rotated to match the PDF orientation`

Attributes:      • *write*

### 2.13.6. `types.dt_imageio_module_format_data_pdf.mode`

`string`

The image mode to use at export time

Attributes:      • *write*

### 2.13.7. `types.dt_imageio_module_format_data_pdf.size`

string

The paper size to use

Attributes:      • *write*

### 2.13.8. `types.dt_imageio_module_format_data_pdf.compression`

string

Compression mode to use for images

Attributes:      • *write*

### 2.13.9. `types.dt_imageio_module_format_data_pdf.pages`

string

The page type to use

Attributes:      • *write*

### 2.13.10. `types.dt_imageio_module_format_data_pdf.rotate`

boolean

Should the images be rotated in the resulting PDF

Attributes:      • *write*

## 2.14. `types._pdf_mode_t`

enum

The export mode to use for PDF document

Attributes:      • *values* :

- normal
- draft
- debug

## 2.15. `types._pdf_pages_t`

enum

The different page types for PDF export

Attributes:      • *values* :

- all



- single
- contact

## 2.16. `types.dt_pdf_stream_encoder_t`

enum

The compression mode for PDF document

- Attributes:
- *values* :
  - uncompressed
  - deflate

## 2.17. `types.dt_imageio_module_storage_t`

dt\_type

A virtual type representing all storage types.

### 2.17.1. `types.dt_imageio_module_storage_t.plugin_name`

string

A unique name for the plugin.

- Attributes:
- *write*

### 2.17.2. `types.dt_imageio_module_storage_t.name`

string

A human readable name for the plugin.

- Attributes:
- *write*

### 2.17.3. `types.dt_imageio_module_storage_t.width`

number

The currently selected width for the plugin.

- Attributes:
- *write*

### 2.17.4. `types.dt_imageio_module_storage_t.height`

number

The currently selected height for the plugin.

- Attributes:
- *write*

### 2.17.5. `types.dt_imageio_module_storage_t.recommended_width`

number

The recommended width for the plugin.

Attributes:      • *write*

#### **2.17.6. `types.dt_imageio_module_storage_t.recommended_height`**

number

The recommended height for the plugin.

Attributes:      • *write*

#### **2.17.7. `types.dt_imageio_module_storage_t.supports_format`**

```
self: function(  
    format : types.dt_imageio_module_format_t  
    ) : boolean
```

Checks if a format is supported by this storage.

self

`types.dt_imageio_module_storage_t`

The storage type to check against.

format

`types.dt_imageio_module_format_t`

The format type to check.

*return*

boolean

True if the format is supported by the storage.

#### **2.18. `types.dt_imageio_module_storage_data_email`**

dt\_type

An object containing parameters to export to email.

Attributes:      • *parent* : `types.dt_imageio_module_storage_t`

#### **2.19. `types.dt_imageio_module_storage_data_latex`**

dt\_type

An object containing parameters to export to latex.

Attributes:      • *parent* : `types.dt_imageio_module_storage_t`

##### **2.19.1. `types.dt_imageio_module_storage_data_latex.filename`**

string

The filename to export to.

Attributes:      • *write*

### **2.19.2. types.dt\_imageio\_module\_storage\_data\_latex.title**

string

The title to use for export.

Attributes:      • *write*

## **2.20. types.dt\_imageio\_module\_storage\_data\_piwigo**

dt\_type

An object containing parameters to export to piwigo.

Attributes:      • *parent* : types.dt\_imageio\_module\_storage\_t

## **2.21. types.dt\_imageio\_module\_storage\_data\_gallery**

dt\_type

An object containing parameters to export to gallery.

Attributes:      • *parent* : types.dt\_imageio\_module\_storage\_t

### **2.21.1. types.dt\_imageio\_module\_storage\_data\_gallery.filename**

string

The filename to export to.

Attributes:      • *write*

### **2.21.2. types.dt\_imageio\_module\_storage\_data\_gallery.title**

string

The title to use for export.

Attributes:      • *write*

## **2.22. types.dt\_imageio\_module\_storage\_data\_disk**

dt\_type

An object containing parameters to export to disk.

Attributes:      • *parent* : types.dt\_imageio\_module\_storage\_t

### **2.22.1. types.dt\_imageio\_module\_storage\_data\_disk.filename**

string

The filename to export to.

Attributes:      • *write*

## 2.23. types.dt\_lua\_film\_t

dt\_type

A film in darktable; this represents a directory containing imported images.

Attributes:      • *has\_tostring*

### 2.23.1. types.dt\_lua\_film\_t.move\_image

see darktable.database.move\_image

### 2.23.2. types.dt\_lua\_film\_t.copy\_image

see darktable.database.copy\_image

### 2.23.3. types.dt\_lua\_film\_t.#

types.dt\_lua\_image\_t

The different images within the film.

### 2.23.4. types.dt\_lua\_film\_t.id

number

A unique numeric id used by this film.

Attributes:      • *write*

### 2.23.5. types.dt\_lua\_film\_t.path

string

The path represented by this film.

Attributes:      • *write*

### 2.23.6. types.dt\_lua\_film\_t.delete

```
self:function(  
  [force : Boolean]  
)
```

Removes the film from the database.

self

types.dt\_lua\_film\_t

The film to remove.

[force]

Boolean

Force removal, even if the film is not empty.

## 2.24. **types.dt\_style\_t**

dt\_type

A style that can be applied to an image.

Attributes:      • *has\_tostring*

### 2.24.1. **types.dt\_style\_t.delete**

see darktable.styles.delete

### 2.24.2. **types.dt\_style\_t.duplicate**

see darktable.styles.duplicate

### 2.24.3. **types.dt\_style\_t.apply**

see darktable.styles.apply

### 2.24.4. **types.dt\_style\_t.export**

see darktable.styles.export

### 2.24.5. **types.dt\_style\_t.name**

string

The name of the style.

### 2.24.6. **types.dt\_style\_t.description**

string

The description of the style.

### 2.24.7. **types.dt\_style\_t.#**

types.dt\_style\_item\_t

The different items that make the style.

## 2.25. **types.dt\_style\_item\_t**

dt\_type

An element that is part of a style.

Attributes:      • *has\_tostring*

### 2.25.1. **types.dt\_style\_item\_t.name**

string

The name of the style item.

### 2.25.2. **types.dt\_style\_item\_t.num**

number

The position of the style item within its style.

## 2.26. **types.dt\_lua\_tag\_t**

dt\_type

A tag that can be attached to an image.

Attributes:      • *has\_tostring*

### 2.26.1. **types.dt\_lua\_tag\_t.delete**

see darktable.tags.delete

### 2.26.2. **types.dt\_lua\_tag\_t.attach**

see darktable.tags.attach

### 2.26.3. **types.dt\_lua\_tag\_t.detach**

see darktable.tags.detach

### 2.26.4. **types.dt\_lua\_tag\_t.name**

string

The name of the tag.

### 2.26.5. **types.dt\_lua\_tag\_t.#**

types.dt\_lua\_image\_t

The images that have that tag attached to them.

## 2.27. **types.dt\_lua\_lib\_t**

dt\_type

The type of a UI lib

### 2.27.1. **types.dt\_lua\_lib\_t.id**

string

A unit string identifying the lib

### 2.27.2. **types.dt\_lua\_lib\_t.name**

string

The translated title of the UI element

### 2.27.3. `types.dt_lua_lib_t.version`

number

The version of the internal data of this lib

### 2.27.4. `types.dt_lua_lib_t.visible`

boolean

Allow to make a lib module completely invisible to the user.

Note that if the module is invisible the user will have no way to restore it without lua

Attributes:      • *implicit\_yield*  
                  • *write*

### 2.27.5. `types.dt_lua_lib_t.container`

`types.dt_ui_container_t`

The location of the lib in the darktable UI

### 2.27.6. `types.dt_lua_lib_t.expandable`

boolean

True if the lib can be expanded/retracted

### 2.27.7. `types.dt_lua_lib_t.expanded`

boolean

True if the lib is expanded

Attributes:      • *write*

### 2.27.8. `types.dt_lua_lib_t.position`

number

A value deciding the position of the lib within its container

### 2.27.9. `types.dt_lua_lib_t.views`

table

A table of all the views that display this widget

### 2.27.10. `types.dt_lua_lib_t.reset`

```
self: function(  
)
```

A function to reset the lib to its default values

This function will do nothing if the lib is not visible or can't be reset

`self`

`types.dt_lua_lib_t`

The lib to reset

#### **2.27.11. `types.dt_lua_lib_t.on_screen`**

`boolean`

True if the lib is currently visible on the screen

#### **2.28. `types.dt_lua_view_t`**

`dt_type`

A darktable view

##### **2.28.1. `types.dt_lua_view_t.id`**

`string`

A unique string identifying the view

##### **2.28.2. `types.dt_lua_view_t.name`**

`string`

The name of the view

#### **2.29. `types.dt_lua_backgroundjob_t`**

`dt_type`

A lua-managed entry in the backgroundjob lib

##### **2.29.1. `types.dt_lua_backgroundjob_t.percent`**

`number`

The value of the progress bar, between 0 and 1. will return nil if there is no progress bar, will raise an error if read or written on an invalid job

Attributes:      • *write*

##### **2.29.2. `types.dt_lua_backgroundjob_t.valid`**

`boolean`

True if the job is displayed, set it to false to destroy the entry

An invalid job cannot be made valid again

Attributes:      • *write*

#### **2.30. `types.dt_lua_snapshot_t`**

`dt_type`



The description of a snapshot in the snapshot lib

Attributes:      • *has\_tostring*

### 2.30.1. **types.dt\_lua\_snapshot\_t.filename**

string

The filename of an image containing the snapshot

### 2.30.2. **types.dt\_lua\_snapshot\_t.select**

```
self:function(  
)
```

Activates this snapshot on the display. To deactivate all snapshot you need to call this function on the active snapshot

self

```
types.dt_lua_snapshot_t
```

The snapshot to activate

### 2.30.3. **types.dt\_lua\_snapshot\_t.name**

string

The name of the snapshot, as seen in the UI

## 2.31. **types.hint\_t**

enum

a hint on the way to encode a webp image

Attributes:      • *values* :

- hint\_default
- hint\_picture
- hint\_photo
- hint\_graphic

## 2.32. **types.dt\_ui\_container\_t**

enum

A place in the darktable UI where a lib can be placed

Attributes:      • *values* :

- DT\_UI\_CONTAINER\_PANEL\_LEFT\_TOP
- DT\_UI\_CONTAINER\_PANEL\_LEFT\_CENTER
- DT\_UI\_CONTAINER\_PANEL\_LEFT\_BOTTOM

- DT\_UI\_CONTAINER\_PANEL\_RIGHT\_TOP
- DT\_UI\_CONTAINER\_PANEL\_RIGHT\_CENTER
- DT\_UI\_CONTAINER\_PANEL\_RIGHT\_BOTTOM
- DT\_UI\_CONTAINER\_PANEL\_TOP\_LEFT
- DT\_UI\_CONTAINER\_PANEL\_TOP\_CENTER
- DT\_UI\_CONTAINER\_PANEL\_TOP\_RIGHT
- DT\_UI\_CONTAINER\_PANEL\_CENTER\_TOP\_LEFT
- DT\_UI\_CONTAINER\_PANEL\_CENTER\_TOP\_CENTER
- DT\_UI\_CONTAINER\_PANEL\_CENTER\_TOP\_RIGHT
- DT\_UI\_CONTAINER\_PANEL\_CENTER\_BOTTOM\_LEFT
- DT\_UI\_CONTAINER\_PANEL\_CENTER\_BOTTOM\_CENTER
- DT\_UI\_CONTAINER\_PANEL\_CENTER\_BOTTOM\_RIGHT
- DT\_UI\_CONTAINER\_PANEL\_BOTTOM

### 2.33. **types.snapshot\_direction\_t**

enum

Which part of the main window is occupied by a snapshot

- Attributes:
- *values* :
    - left
    - right
    - top
    - bottom

### 2.34. **types.dt\_imageio\_j2k\_format\_t**

enum

J2K format type

- Attributes:
- *values* :
    - j2k
    - jp2

### 2.35. **types.dt\_imageio\_j2k\_preset\_t**

enum

J2K preset type

- Attributes:
- *values* :
    - off
    - cinema2k\_24
    - cinema2k\_48
    - cinema4k\_24

## 2.36. types.comp\_type\_t

enum

Type of compression for webp

- Attributes:
- *values* :
    - webp\_lossy
    - webp\_lossless

## 2.37. types.lua\_pref\_type

enum

The type of value to save in a preference

- Attributes:
- *values* :
    - enum
    - directory
    - file
    - string
    - bool
    - integer
    - float
    - lua

## 2.38. types.dt\_imageio\_exr\_compression\_t

enum

The type of compression to use for the EXR image

- Attributes:
- *values* :
    - off

- `rle`
- `zips`
- `zip`
- `piz`
- `pxr24`
- `b44`
- `b44a`

## 2.39. `types.dt_lib_collect_params_rule_t`

`dt_type`

A single rule for filtering a collection

### 2.39.1. `types.dt_lib_collect_params_rule_t.mode`

`types.dt_lib_collect_mode_t`

How this rule is applied after the previous one. Unused for the first rule

Attributes:      • *write*

### 2.39.2. `types.dt_lib_collect_params_rule_t.data`

`string`

The text segment of the rule. Exact content depends on the type of rule

Attributes:      • *write*

### 2.39.3. `types.dt_lib_collect_params_rule_t.item`

`types.dt_collection_properties_t`

The item on which this rule filter. i.e the type of the rule

Attributes:      • *write*

## 2.40. `types.dt_lib_collect_mode_t`

`enum`

The logical operators to apply between rules

Attributes:      • *values :*

- `DT_LIB_COLLECT_MODE_AND`
- `DT_LIB_COLLECT_MODE_OR`
- `DT_LIB_COLLECT_MODE_AND_NOT`

## 2.41. types.dt\_collection\_properties\_t

enum

The different elements on which a collection can be filtered

- Attributes:
- *values* :
    - DT\_COLLECTION\_PROP\_FILMROLL
    - DT\_COLLECTION\_PROP\_FOLDERS
    - DT\_COLLECTION\_PROP\_CAMERA
    - DT\_COLLECTION\_PROP\_TAG
    - DT\_COLLECTION\_PROP\_DAY
    - DT\_COLLECTION\_PROP\_TIME
    - DT\_COLLECTION\_PROP\_IMPORT\_TIMESTAMP
    - DT\_COLLECTION\_PROP\_CHANGE\_TIMESTAMP
    - DT\_COLLECTION\_PROP\_EXPORT\_TIMESTAMP
    - DT\_COLLECTION\_PROP\_PRINT\_TIMESTAMP
    - DT\_COLLECTION\_PROP\_HISTORY
    - DT\_COLLECTION\_PROP\_COLORLABEL
    - DT\_COLLECTION\_PROP\_TITLE
    - DT\_COLLECTION\_PROP\_DESCRIPTION
    - DT\_COLLECTION\_PROP\_CREATOR
    - DT\_COLLECTION\_PROP\_PUBLISHER
    - DT\_COLLECTION\_PROP\_RIGHTS
    - DT\_COLLECTION\_PROP\_LENS
    - DT\_COLLECTION\_PROP\_FOCAL\_LENGTH
    - DT\_COLLECTION\_PROP\_ISO
    - DT\_COLLECTION\_PROP\_APERTURE
    - DT\_COLLECTION\_PROP\_FILENAME
    - DT\_COLLECTION\_PROP\_GEOTAGGING

## 2.42. types.dt\_collection\_sort\_t

enum

The different elements on which a collection can be sorted

- Attributes:
- *values* :
    - DT\_COLLECTION\_SORT\_NONE
    - DT\_COLLECTION\_SORT\_FILENAME
    - DT\_COLLECTION\_SORT\_DATETIME
    - DT\_COLLECTION\_SORT\_RATING
    - DT\_COLLECTION\_SORT\_ID
    - DT\_COLLECTION\_SORT\_COLOR
    - DT\_COLLECTION\_SORT\_GROUP
    - DT\_COLLECTION\_SORT\_PATH
    - DT\_COLLECTION\_SORT\_CUSTOM\_ORDER
    - DT\_COLLECTION\_SORT\_TITLE
    - DT\_COLLECTION\_SORT\_DESCRIPTION
    - DT\_COLLECTION\_SORT\_ASPECT\_RATIO
    - DT\_COLLECTION\_SORT\_SHUFFLE

## 2.43. types.dt\_collection\_sort\_order\_t

enum

The different orders that a collection can be sorted in

- Attributes:
- *values* :
    - DT\_COLLECTION\_SORT\_ORDER\_ASCENDING
    - DT\_COLLECTION\_SORT\_ORDER\_DESCENDING

## 2.44. types.dt\_collection\_filter\_t

enum

The different elements on which a collection can be filtered

- Attributes:
- *values* :
    - DT\_COLLECTION\_FILTER\_ALL
    - DT\_COLLECTION\_FILTER\_STAR\_NO
    - DT\_COLLECTION\_FILTER\_STAR\_1
    - DT\_COLLECTION\_FILTER\_STAR\_2
    - DT\_COLLECTION\_FILTER\_STAR\_3
    - DT\_COLLECTION\_FILTER\_STAR\_4

- DT\_COLLECTION\_FILTER\_STAR\_5
- DT\_COLLECTION\_FILTER\_REJECT
- DT\_COLLECTION\_FILTER\_NOT\_REJECT

## 2.45. types.dt\_collection\_rating\_comperator\_t

enum

The different ways in which a collection filter can be compared

- Attributes:
- *values* :
    - DT\_COLLECTION\_RATING\_COMP\_LT
    - DT\_COLLECTION\_RATING\_COMP\_LEQ
    - DT\_COLLECTION\_RATING\_COMP\_EQ
    - DT\_COLLECTION\_RATING\_COMP\_GEQ
    - DT\_COLLECTION\_RATING\_COMP\_GT
    - DT\_COLLECTION\_RATING\_COMP\_NE
    - DT\_COLLECTION\_RATING\_N\_COMPS

## 2.46. types.dt\_lua\_orientation\_t

enum

A possible orientation for a widget

- Attributes:
- *values* :
    - horizontal
    - vertical

## 2.47. types.dt\_lua\_align\_t

enum

The alignment of a label

- Attributes:
- *values* :
    - fill
    - start
    - end
    - center
    - baseline

## 2.48. `types.dt_lua_ellipsize_mode_t`

enum

The ellipsize mode of a label

Attributes:

- *values:*
  - none
  - start
  - middle
  - end

## 2.49. `types.dt_lua_cairo_t`

dt\_type

A wrapper around a cairo drawing context.

You probably shouldn't use this after the callback that got it passed returned.

For more details of the member functions have a look at the cairo documentation for the drawing context [<http://www.cairographics.org/manual/cairo-cairo-t.html>], transformations [<http://www.cairographics.org/manual/cairo-Transformations.html>] and paths [<http://www.cairographics.org/manual/cairo-Paths.html>].

### 2.49.1. `types.dt_lua_cairo_t.save`

```
self:function(  
)
```

Save the state of the drawing context.

self

`types.dt_lua_cairo_t`

The context to modify.

### 2.49.2. `types.dt_lua_cairo_t.restore`

```
self:function(  
)
```

Restore a previously saved state.

self

`types.dt_lua_cairo_t`

The context to modify.

### 2.49.3. `types.dt_lua_cairo_t.move_to`

```
self:function(  
  x : float,
```



```
y : float
)
```

Begin a new sub-path.

self

```
types.dt_lua_cairo_t
```

The context to modify

x

```
float
```

The x coordinate of the new position.

y

```
float
```

The y coordinate of the new position.

#### 2.49.4. `types.dt_lua_cairo_t.line_to`

```
self:function(
  x : float,
  y : float
)
```

Add a line to the path.

self

```
types.dt_lua_cairo_t
```

The context to modify.

x

```
float
```

The x coordinate of the end of the new line.

y

```
float
```

The y coordinate of the end of the new line.

#### 2.49.5. `types.dt_lua_cairo_t.rectangle`

```
self:function(
  x : float,
  y : float,
  width : float,
  height : float
)
```

Add a closed sub-path rectangle.

**self**

`types.dt_lua_cairo_t`

The context to modify.

**x**

`float`

The x coordinate of the top left corner of the rectangle.

**y**

`float`

The y coordinate of the top left corner of the rectangle.

**width**

`float`

The width of the rectangle.

**height**

`float`

The height of the rectangle.

#### **2.49.6. `types.dt_lua_cairo_t.arc`**

```
self:function(  
  x : float,  
  y : float,  
  radius : float,  
  angle1 : float,  
  angle2 : float  
)
```

Add a circular arc.

**self**

`types.dt_lua_cairo_t`

The context to modify.

**x**

`float`

The x position of the center of the arc.

**y**

`float`

The y position of the center of the arc.

radius

float

The radius of the arc.

angle1

float

The start angle, in radians.

angle2

float

The end angle, in radians.

#### 2.49.7. `types.dt_lua_cairo_t.arc_negative`

```
self:function(  
  x : float,  
  y : float,  
  radius : float,  
  angle1 : float,  
  angle2 : float  
)
```

Add a circular arc. It only differs in the direction from `types.dt_lua_cairo_t.arc`.

self

`types.dt_lua_cairo_t`

The context to modify.

x

float

The x position of the center of the arc.

y

float

The y position of the center of the arc.

radius

float

The radius of the arc.

angle1

float

The start angle, in radians.

angle2

float

The end angle, in radians.

#### 2.49.8. `types.dt_lua_cairo_t.rotate`

```
self:function(  
  angle : float  
)
```

Add a rotation to the transformation matrix.

self

`types.dt_lua_cairo_t`

The context to modify.

angle

float

The angle (in radians) by which the user-space axes will be rotated.

#### 2.49.9. `types.dt_lua_cairo_t.scale`

```
self:function(  
  x : float,  
  y : float  
)
```

Add a scaling to the transformation matrix.

self

`types.dt_lua_cairo_t`

The context to modify.

x

float

The scale factor for the x dimension.

y

float

The scale factor for the y dimension.

#### 2.49.10. `types.dt_lua_cairo_t.translate`

```
self:function(  
  x : float,  
  y : float  
)
```

Add a translation to the transformation matrix.

**self**

`types.dt_lua_cairo_t`

The context to modify.

**x**

`float`

Amount to translate in the x direction

**y**

`float`

Amount to translate in the y direction

#### **2.49.11. `types.dt_lua_cairo_t.new_sub_path`**

```
self:function(  
)
```

Begin a new sub-path.

**self**

`types.dt_lua_cairo_t`

The context to modify.

#### **2.49.12. `types.dt_lua_cairo_t.draw_line`**

```
self:function(  
  x_start : float,  
  y_start : float,  
  x_end   : float,  
  y_end   : float  
)
```

Helper function to draw a line with a given start and end.

**self**

`types.dt_lua_cairo_t`

The context to modify.

**x\_start**

`float`

The x coordinate of the start of the new line.

**y\_start**

`float`

The y coordinate of the start of the new line.

x\_end

float

The x coordinate of the end of the new line.

y\_end

float

The y coordinate of the end of the new line.

## 2.50. types.dt\_ui\_panel\_t

enum

The different user interface panels

- Attributes:
- *values* :
    - DT\_UI\_PANEL\_TOP
    - DT\_UI\_PANEL\_CENTER\_TOP
    - DT\_UI\_PANEL\_CENTER\_BOTTOM
    - DT\_UI\_PANEL\_LEFT
    - DT\_UI\_PANEL\_RIGHT
    - DT\_UI\_PANEL\_BOTTOM
    - DT\_UI\_PANEL\_SIZE

## 2.51. types.dt\_lighttable\_layout\_t

enum

The different lighttable layouts

- Attributes:
- *values* :
    - DT\_LIGHTTABLE\_LAYOUT\_FIRST
    - DT\_LIGHTTABLE\_LAYOUT\_ZOOMABLE
    - DT\_LIGHTTABLE\_LAYOUT\_FILEMANAGER
    - DT\_LIGHTTABLE\_LAYOUT\_CULLING
    - DT\_LIGHTTABLE\_LAYOUT\_LAST

## 2.52. types.lua\_widget

dt\_type

Common parent type for all lua-handled widgets

Attributes:      • *has\_tostring*

### 2.52.1. **types.lua\_widget.extra registration parameters**

This widget has no extra registration parameters

### 2.52.2. **types.lua\_widget.sensitive**

boolean

Set if the widget is enabled/disabled

Attributes:      • *write*

### 2.52.3. **types.lua\_widget.tooltip**

string or nil

Tooltip to display for the widget

Attributes:      • *write*

### 2.52.4. **types.lua\_widget.reset\_callback**

```
function(  
  widget : types.lua_widget  
)
```

A function to call when the widget needs to reset itself

Note that some widgets have a default implementation that can be overridden, (containers in particular will recursively reset their children). If you replace that default implementation you need to reimplement that functionality or call the original function within your callback

Attributes:      • *write*

widget

types.lua\_widget

The widget that triggered the callback

### 2.52.5. **types.lua\_widget.As a function**

```
function(  
  attributes : table  
) : types.lua_widget
```

Using a lua widget as a function Allows to set multiple attributes of that widget at once. This is mainly used to create UI elements in a more readable way

For example:

```
local widget = dt.new_widget("button"){  
  label = "my label",  
  clicked_callback = function() print "hello world" end
```

}

attributes

table

A table of attributes => value to set

*return*

`types.lua_widget`

The object called itself, to allow chaining

## 2.53. `types.lua_container`

dt\_type

A widget containing other widgets

Attributes:

- *has\_tostring*
- *parent* : `types.lua_widget`

### 2.53.1. `types.lua_container.__call`

see `types.lua_widget`.As a function

### 2.53.2. `types.lua_container.extra registration parameters`

This widget has no extra registration parameters

### 2.53.3. `types.lua_container.#`

`types.lua_widget`

The widgets contained by the box

You can append widgets by adding them at the end of the list

You can remove widgets by setting them to nil

Attributes:

- *write*

## 2.54. `types.lua_check_button`

dt\_type

A checkable button with a label next to it

Attributes:

- *has\_tostring*
- *parent* : `types.lua_widget`

### 2.54.1. `types.lua_check_button.__call`

see `types.lua_widget`.As a function



### 2.54.2. `types.lua_check_button.extra` registration parameters

This widget has no extra registration parameters

### 2.54.3. `types.lua_check_button.label`

`string`

The label displayed next to the button

Attributes:      • *write*

### 2.54.4. `types.lua_check_button.value`

`boolean`

If the widget is checked or not

Attributes:      • *write*

### 2.54.5. `types.lua_check_button.clicked_callback`

```
function(  
  widget : types.lua_widget  
)
```

A function to call on button click

Attributes:      • *write*

`widget`

`types.lua_widget`

The widget that triggered the callback

## 2.55. `types.lua_label`

`dt_type`

A label containing some text

Attributes:      • *has\_tostring*  
                  • *parent* : `types.lua_widget`

### 2.55.1. `types.lua_label.__call`

see `types.lua_widget`.As a function

### 2.55.2. `types.lua_label.extra` registration parameters

This widget has no extra registration parameters

### 2.55.3. `types.lua_label.label`

`string`

The label displayed

Attributes:      • *write*

#### **2.55.4. `types.lua_label.selectable`**

boolean

True if the label content should be selectable

Attributes:      • *write*

#### **2.55.5. `types.lua_label.halign`**

`types.dt_lua_align_t`

The horizontal alignment of the label

Attributes:      • *write*

#### **2.55.6. `types.lua_label.ellipsize`**

`types.dt_lua_ellipsize_mode_t`

The ellipsize mode of the label

Attributes:      • *write*

### **2.56. `types.lua_button`**

`dt_type`

A clickable button

Attributes:      • *has\_tostring*  
                 • *parent* : `types.lua_widget`

#### **2.56.1. `types.lua_button.__call`**

see `types.lua_widget`.As a function

#### **2.56.2. `types.lua_button.extra registration parameters`**

This widget has no extra registration parameters

#### **2.56.3. `types.lua_button.label`**

string

The label displayed on the button

Attributes:      • *write*

#### **2.56.4. `types.lua_button.ellipsize`**

`types.dt_lua_ellipsize_mode_t`

The ellipsize mode of the button label

Attributes:      • *write*

#### 2.56.5. **types.lua\_button.clicked\_callback**

```
function(  
  widget : types.lua_widget  
)
```

A function to call on button click

Attributes:      • *write*

widget

`types.lua_widget`

The widget that triggered the callback

### 2.57. **types.lua\_box**

dt\_type

A container for widget in a horizontal or vertical list

Attributes:      • *has\_tostring*  
                  • *parent* : types.lua\_container

#### 2.57.1. **types.lua\_box.\_\_call**

see types.lua\_widget.As a function

#### 2.57.2. **types.lua\_box.extra registration parameters**

This widget has no extra registration parameters

#### 2.57.3. **types.lua\_box.orientation**

`types.dt_lua_orientation_t`

The orientation of the box.

Attributes:      • *write*

### 2.58. **types.lua\_entry**

dt\_type

A widget in which the user can input text

Attributes:      • *has\_tostring*  
                  • *parent* : types.lua\_widget

#### 2.58.1. **types.lua\_entry.\_\_call**

see `types.lua_widget`.As a function

### 2.58.2. `types.lua_entry.extra` registration parameters

This widget has no extra registration parameters

### 2.58.3. `types.lua_entry.text`

`string`

The content of the entry

Attributes:      • *write*

### 2.58.4. `types.lua_entry.placeholder`

`string`

The text to display when the entry is empty

Attributes:      • *write*

### 2.58.5. `types.lua_entry.is_password`

`boolean`

True if the text content should be hidden

Attributes:      • *write*

### 2.58.6. `types.lua_entry.editable`

`boolean`

False if the entry should be read-only

Attributes:      • *write*

## 2.59. `types.lua_separator`

`dt_type`

A widget providing a separation in the UI.

Attributes:      • *has\_tostring*  
                  • *parent* : `types.lua_widget`

### 2.59.1. `types.lua_separator.__call`

see `types.lua_widget`.As a function

### 2.59.2. `types.lua_separator.extra` registration parameters

This widget has no extra registration parameters

### 2.59.3. `types.lua_separator.orientation`

string

The orientation of the separator.

Attributes:      • *write*

## 2.60. types.lua\_combobox

dt\_type

A widget with multiple text entries in a menu

This widget can be set as editable at construction time.

If it is editable the user can type a value and is not constrained by the values in the menu

Attributes:      • *has\_tostring*  
                 • *parent* : types.lua\_widget

### 2.60.1. types.lua\_combobox.\_\_call

see types.lua\_widget.As a function

### 2.60.2. types.lua\_combobox.extra registration parameters

This widget has no extra registration parameters

### 2.60.3. types.lua\_combobox.value

string

The text content of the selected entry, can be nil

You can set it to a number to select the corresponding entry from the menu

If the combo box is editable, you can set it to any string

You can set it to nil to deselect all entries

Attributes:      • *write*

### 2.60.4. types.lua\_combobox.selected

integer

The index of the selected entry, or 0 if nothing is selected

You can set it to a number to select the corresponding entry from the menu, or to 0 to select nothing

You can set it to nil to deselect all entries

Attributes:      • *write*

### 2.60.5. types.lua\_combobox.#

string

The various menu entries.

You can add new entries by writing to the first element beyond the end

You can removes entries by setting them to nil

Attributes:      • *write*

#### **2.60.6. types.lua\_combobox.changed\_callback**

```
function(  
  widget : types.lua_widget  
)
```

A function to call when the value field changes (character entered or value selected)

Attributes:      • *write*

widget

types.lua\_widget

The widget that triggered the callback

#### **2.60.7. types.lua\_combobox.editable**

boolean

True is the user is allowed to type a string in the combobox

Attributes:      • *write*

#### **2.60.8. types.lua\_combobox.label**

string

The label displayed on the combobox

Attributes:      • *write*

#### **2.61. types.lua\_file\_chooser\_button**

dt\_type

A button that allows the user to select an existing file

Attributes:      • *has\_tostring*  
                  • *parent* : types.lua\_widget

##### **2.61.1. types.lua\_file\_chooser\_button.\_\_call**

see types.lua\_widget.As a function

##### **2.61.2. types.lua\_file\_chooser\_button.extra registration parameters**

This widget has no extra registration parameters

### 2.61.3. `types.lua_file_chooser_button.title`

string

The title of the window when choosing a file

Attributes:      • *write*

### 2.61.4. `types.lua_file_chooser_button.value`

string

The currently selected file

Attributes:      • *write*

### 2.61.5. `types.lua_file_chooser_button.changed_callback`

```
function(  
  widget : types.lua_widget  
)
```

A function to call when the value field changes (character entered or value selected)

Attributes:      • *write*

widget

`types.lua_widget`

The widget that triggered the callback

### 2.61.6. `types.lua_file_chooser_button.is_directory`

boolean

True if the file chooser button only allows directories to be selected

Attributes:      • *write*

## 2.62. `types.lua_stack`

dt\_type

A container that will only show one of its child at a time

Attributes:      • *has\_tostring*  
                  • *parent* : `types.lua_container`

### 2.62.1. `types.lua_stack.__call`

see `types.lua_widget`.As a function

### 2.62.2. `types.lua_stack.extra registration parameters`

This widget has no extra registration parameters

### 2.62.3. `types.lua_stack.active`

`types.lua_widget` or `nil`

The currently selected child, can be `nil` if the container has no child, can be set to one of the child widget or to an index in the child table

Attributes:      • *write*

### 2.62.4. `types.lua_stack.h_size_fixed`

boolean

True if horizontal size is fixed, false if stack can be resized horizontally.

Attributes:      • *write*

### 2.62.5. `types.lua_stack.v_size_fixed`

boolean

True if vertical size is fixed, false if stack can be resized vertically.

Attributes:      • *write*

## 2.63. `types.lua_slider`

`dt_type`

A slider that can be set by the user

Attributes:      • *has\_tostring*  
                 • *parent* : `types.lua_widget`

### 2.63.1. `types.lua_slider.__call`

see `types.lua_widget`.As a function

### 2.63.2. `types.lua_slider.extra` registration parameters

This widget has no extra registration parameters

### 2.63.3. `types.lua_slider.soft_min`

number

The soft minimum value for the slider, the slider can't go beyond this point

Attributes:      • *write*

### 2.63.4. `types.lua_slider.soft_max`

number

The soft maximum value for the slider, the slider can't go beyond this point



Attributes:      • *write*

#### **2.63.5. types.lua\_slider.hard\_min**

number

The hard minimum value for the slider, the user can't manually enter a value beyond this point

Attributes:      • *write*

#### **2.63.6. types.lua\_slider.hard\_max**

number

The hard maximum value for the slider, the user can't manually enter a value beyond this point

Attributes:      • *write*

#### **2.63.7. types.lua\_slider.step**

number

The step width of the slider

Attributes:      • *write*

#### **2.63.8. types.lua\_slider.digits**

integer

The number of decimal digits shown on the slider

Attributes:      • *write*

#### **2.63.9. types.lua\_slider.value**

number

The current value of the slider

Attributes:      • *write*

#### **2.63.10. types.lua\_slider.label**

string

The label next to the slider

Attributes:      • *write*

### **2.64. types.lua\_text\_view**

dt\_type

A multiline text input widget

Attributes:      • *has\_tostring*  
                  • *parent* : types.lua\_widget

#### **2.64.1. types.lua\_text\_view.\_\_call**

see types.lua\_widget.As a function

#### **2.64.2. types.lua\_text\_view.extra registration parameters**

This widget has no extra registration parameters

#### **2.64.3. types.lua\_text\_view.text**

string

The text in the widget

Attributes:      • *write*

#### **2.64.4. types.lua\_text\_view.editable**

boolean

False if the entry should be read-only

Attributes:      • *write*

### **2.65. types.lua\_section\_label**

dt\_type

A section label

Attributes:      • *has\_tostring*  
                  • *parent* : types.lua\_widget

#### **2.65.1. types.lua\_section\_label.\_\_call**

see types.lua\_widget.As a function

#### **2.65.2. types.lua\_section\_label.extra registration parameters**

This widget has no extra registration parameters

#### **2.65.3. types.lua\_section\_label.label**

string

The section name

Attributes:      • *write*

## 3. events

This section documents events that can be used to trigger Lua callbacks.

### 3.1. events.intermediate-export-image

event

This event is called each time an image is exported, once for each image after the image has been processed to an image format but before the storage has moved the image to its final destination. The call is blocking.

#### 3.1.1. events.intermediate-export-image.callback

```
function(  
    event : string,  
    image : types.dt_lua_image_t,  
    filename : string,  
    format : types.dt_imageio_module_format_t,  
    storage : types.dt_imageio_module_storage_t  
)
```

event

string

The name of the event that triggered the callback.

image

types.dt\_lua\_image\_t

The image object that has been exported.

filename

string

The name of the file that is the result of the image being processed.

format

types.dt\_imageio\_module\_format\_t

The format used to export the image.

storage

types.dt\_imageio\_module\_storage\_t

The storage used to export the image (can be nil).

#### 3.1.2. events.intermediate-export-image.extra registration parameters

This event has no extra registration parameters.

### 3.2. events.post-import-image

event

This event is triggered whenever a new image is imported into the database. This event can be registered multiple times, all callbacks will be called. The call is blocking.

### 3.2.1. events.post-import-image.callback

```
function(  
    event : string,  
    image : types.dt_lua_image_t  
)
```

event

string

The name of the event that triggered the callback.

image

types.dt\_lua\_image\_t

The image object that has been imported.

### 3.2.2. events.post-import-image.extra registration parameters

This event has no extra registration parameters.

## 3.3. events.shortcut

event

This event registers a new keyboard shortcut. The shortcut isn't bound to any key until the users does so in the preference panel. The event is triggered whenever the shortcut is triggered. This event can only be registered once per value of shortcut.

### 3.3.1. events.shortcut.callback

```
function(  
    event : string,  
    shortcut : string  
)
```

event

string

The name of the event that triggered the callback.

shortcut

string

The tooltip string that was given at registration time.

### 3.3.2. events.shortcut.extra registration parameters

tooltip

string

The string that will be displayed on the shortcut preference panel describing the shortcut.

### 3.4. events.post-import-film

event

This event is triggered when an film import is finished (all post-import-image callbacks have already been triggered). This event can be registered multiple times.

#### 3.4.1. events.post-import-film.callback

```
function(  
    event : string,  
    film : types.dt_lua_film_t  
)
```

event

string

The name of the event that triggered the callback.

film

types.dt\_lua\_film\_t

The new film that has been added. If multiple films were added recursively only the top level film is reported.

#### 3.4.2. events.post-import-film.extra registration parameters

This event has no extra registration parameters.

### 3.5. events.view-changed

event

This event is triggered after the user changed the active view

#### 3.5.1. events.view-changed.callback

```
function(  
    event : string,  
    old_view : types.dt_lua_view_t,  
    new_view : types.dt_lua_view_t  
)
```

event

string

The name of the event that triggered the callback.

old\_view

`types.dt_lua_view_t`

The view that we just left

new\_view

`types.dt_lua_view_t`

The view we are now in

### **3.5.2. events.view-changed.extra registration parameters**

This event has no extra registration parameters.

## **3.6. events.global\_toolbox-grouping\_toggle**

event

This event is triggered after the user toggled the grouping button.

### **3.6.1. events.global\_toolbox-grouping\_toggle.callback**

```
function(  
  toggle : boolean  
)
```

toggle

boolean

the new grouping status.

### **3.6.2. events.global\_toolbox-grouping\_toggle.extra registration parameters**

This event has no extra registration parameters.

## **3.7. events.global\_toolbox-overlay\_toggle**

event

This event is triggered after the user toggled the overlay button.

### **3.7.1. events.global\_toolbox-overlay\_toggle.callback**

```
function(  
  toggle : boolean  
)
```

toggle

boolean

the new overlay status.

### **3.7.2. events.global\_toolbox-overlay\_toggle.extra registration parameters**

This event has no extra registration parameters.

### **3.8. events.mouse-over-image-changed**

event

This event is triggered whenever the image under the mouse changes

#### **3.8.1. events.mouse-over-image-changed.callback**

```
function(  
    event : string,  
    image : types.dt_lua_image_t  
)
```

event

string

The name of the event that triggered the callback.

image

types.dt\_lua\_image\_t

The new image under the mouse, can be nil if there is no image under the mouse

#### **3.8.2. events.mouse-over-image-changed.extra registration parameters**

This event has no extra registration parameters.

### **3.9. events.exit**

event

This event is triggered when darktable exits, it allows lua scripts to do cleanup jobs

#### **3.9.1. events.exit.callback**

```
function(  
)
```

#### **3.9.2. events.exit.extra registration parameters**

This event has no extra registration parameters.

### **3.10. events.pre-import**

event

This event is trigger before any import action

#### **3.10.1. events.pre-import.callback**

```
function(  
    event : string,
```

```
images : table of string  
)
```

**event**

string

The name of the event that triggered the callback.

**images**

table of string

The files that will be imported. Modifying this table will change the list of files that will be imported"

### **3.10.2. events.pre-import.extra registration parameters**

This event has no extra registration parameters.



## **4. attributes**

This section documents various attributes used throughout the documentation.

### **4.1. attributes.write**

This object is a variable that can be written to.

### **4.2. attributes.has\_tostring**

This object has a specific reimplementation of the "tostring" method that allows pretty-printing it.

### **4.3. attributes.implicit\_yield**

This call will release the Lua lock while executing, thus allowing other Lua callbacks to run.

### **4.4. attributes.parent**

This object inherits some methods from another object. You can call the methods from the parent on the child object

