Cairo-Dock

3.4.0

Generated by Doxygen 1.8.8

Fri Oct 17 2014 00:24:04

Contents

Chapter 1

Cairo-Dock's API documentation.

- How can my applet control the window of an application ?
- How can I render some numerical values on my icon ?
- How can I make my applet multi-instanciable ?
- How can I draw anywhere on the dock, not only on my icon ?

1.1 Introduction

This documentation presents the core library of Cairo-Dock: *libgldi* (GL Desktop Interface).

It is useful if you want to write a plug-in, add new features in the core, or just love C.
Note: to write applets in any language very easily, see `http://doc.glx-dock.org`.

It has a **decentralized conception** and is built of several modules: internal modules (Managers) and external modules (Modules) that can extend it.
It also has an Objects architecture.

1.2 Installation

The installation is very easy and uses *cmake*. In a terminal, copy-paste the following commands :

```
### grab the sources of the core
mkdir CD && cd CD
bzr checkout --lightweight lp:cairo-dock-core
### compil the dock and install it
cd cairo-dock-core
cmake CMakeLists.txt -DCMAKE_INSTALL_PREFIX=/usr
make
sudo make install
### grab the sources of the plug-ins
cd ..
bzr checkout --lightweight lp:cairo-dock-plug-ins
### compil the stable plug-ins and install them
cmake CMakeLists.txt -DCMAKE_INSTALL_PREFIX=/usr
make
sudo make install
```

To install unstable plug-ins, add -Denable-xxx=yes to the cmake command, where xxx is the lower-case name of the applet.

1.3 Main structures

1.3.1 Objects

Any element in *libgldi* is a _GldiObject.
An Object is created by an ObjectManager, which defines the properties and notifications of its children.
It has a reference counter, can be deleted from the current theme, and can be reloaded.
An Object can cast **notifications**; notifications are broadcasted on its ObjectManager.
An ObjectManager can inherit from another ObjectManager; in this case, all methods of the parent ObjectManagers are called recursively, and likewise all notifications on an Object are casted recursively to all parent ObjectManagers.

See _GldiObject and cairo-dock-object.h for more details.

1.3.2 Managers

The core is divided in several internal modules, called Managers.
Each Manager manages a set of parameters and objects (for instance, the Dock Manager manages the list of all Docks and their parameters).

See _GldiManager and cairo-dock-manager.h for more details.

1.3.3 Containers

Containers are generic animated windows. They can hold Icons and support cairo/OpenGL drawing.

See _GldiContainer and cairo-dock-container.h for more details.

1.3.4 Icons

Icons are elements inside a Container on which the user can interact. For instance, a Launcher is an Icon that launches a program on left-click.

See _Icon and cairo-dock-icon-factory.h for more details.

1.3.5 Dock

Docks are a kind of Container that sits on a border of the screen.

See _CairoDock and cairo-dock-dock-factory.h for more details.

1.3.6 Desklet

Desklets are a kind of Container that stays on the desktop and holds one or many icons.

See _CairoDesklet and cairo-dock-desklet-factory.h for more details.

1.3.7 Dialog

Dialogs are a kind of Container that holds no icon, but rather point to an icon, and are used to display some information or interact with the user.

See _CairoDialog and cairo-dock-dialog-factory.h for more details.

1.3.8 Modules

A Module is an Object representing a plug-in for *libgldi*.
It defines a set of properties and an interface for init/stop/reload.
A Module that adds an Icon is called an *"applet"*.

See _GldiModule and cairo-dock-module-manager.h for more details.

Note: the `cairo-dock-plug-ins` project is a set of modules in the form of loadable libraries (.so files).
the `cairo-dock-plug-ins-extra` project is a set of modules in the form of scripts (Python or any language) that interact on the core through Dbus.

1.3.9 Module-Instances

A Module-Instance is an actual instance of a Module.
It holds a set of parameters and data (amongst them the Applet-Icon if it's an applet).

A Module can have several instances.

See _GldiModuleInstance and cairo-dock-module-instance-manager.h for more details.

1.3.10 Drawing with cairo/opengl

libgldi defines _CairoDockImageBuffer, a generic Image that works for both cairo and OpenGL.
See cairo-dock-image-buffer.h for more details.

It is possible to add small images above Icons; they are called _CairoOverlay.
For instance quick-info and progress-bars are Overlays.
See cairo-dock-overlay.h for more details.

1.3.11 Windows management

libgldi keeps track of all the currently existing windows, with all their properties, and notifies everybody of any change. It is used for the Taskbar.
Each window has a corresponding _GldiWindowActor object.
See cairo-dock-windows-manager.h for more details.

1.4 External Modules

1.4.1 Create a new applet

Go to the "plug-ins" folder, and run the *generate-applet.sh* script. Answer the few questions, and you're done!
The script creates a <module-name> folder, with *src* and *data* sub-folders, which contain the following:

- data/icon.png: the default icon of your applet
- data/preview.jpg: a preview of your applet, around 200x200 pixels
- data/<module-name>.conf.in: the config file of your applet
- src/applet-init.c: contains the *init*, *stop* and *reload* methods, as well as the definition of your applet.
- src/applet-config.c: container the *get_config* and *reset_config* methods
- src/applet-notifications.c: contains the callbacks of your applet (ie, the code that is called on events, for instance on click on the icon)
- src/applet-struct.h: contains the structures (Config, Data, and any other you may need)

Note: when adding a new file, don't forget to add it in the CMakeLists.txt.
when changing something in the config file, don't forget to update the version number of the applet, in the main CMakeLists.txt.
when changing anything, don't forget to install (*sudo make install*)

1.4.2 First steps

Edit the file *src/applet-inic.c*; the macro CD_APPLET_DEFINITION is a convenient way to define an applet: just fill its name, its category, a brief description, and your name.

In the section CD_APPLET_INIT_BEGIN/CD_APPLET_INIT_END, write the code that will run on startup.

In the section CD_APPLET_STOP_BEGIN/CD_APPLET_STOP_END, write the code that will run when the applet is deactivated: remove any timer, destroy any allocated ressources, unregister notifications, etc.

In the section CD_APPLET_RELOAD_BEGIN/CD_APPLET_RELOAD_END section, write the code that will run when the applet is reloaded; this can happen in 2 cases:

- when the configuration is changed (CD_APPLET_MY_CONFIG_CHANGED is TRUE, for instance when the user edits the applet)
- when something else changed (CD_APPLET_MY_CONFIG_CHANGED is FALSE, for instance when the icon theme is changed, or the icon size is changed); in this case, most of the time you have nothing to do, except if you loaded some ressources yourself.

Edit the file *src/applet-config.c*; In the section CD_APPLET_GET_CONFIG_BEGIN/CD_APPLET_GET_CONFIG↩_END, get all your config parameters (don't forget to define them in applet-struct.h).

In the section CD_APPLET_RESET_CONFIG_BEGIN/CD_APPLET_RESET_CONFIG_END, free any config parameter that was allocated (for instance, strings).

Edit the file *src/applet-notifications.c*;

In the section CD_APPLET_ON_CLICK_BEGIN/CD_APPLET_ON_CLICK_END, write the code that will run when the user clicks on the icon (or an icon of the sub-dock).

There are other similar sections available:

- CD_APPLET_ON_MIDDLE_CLICK_BEGIN/CD_APPLET_ON_MIDDLE_CLICK_END for the actions on middle click on your icon or one of its sub-dock.
- CD_APPLET_ON_DOUBLE_CLICK_BEGIN/CD_APPLET_ON_DOUBLE_CLICK_END for the actions on double click on your icon or one of its sub-dock.
- CD_APPLET_ON_SCROLL_BEGIN/CD_APPLET_ON_SCROLL_END for the actions on scroll on your icon or one of its sub-dock.
- CD_APPLET_ON_BUILD_MENU_BEGIN/CD_APPLET_ON_BUILD_MENU_END for the building of the menu on left click on your icon or one of its sub-dock.

To register to an event, use one of the following convenient macro during the init:

- CD_APPLET_REGISTER_FOR_CLICK_EVENT
- CD_APPLET_REGISTER_FOR_MIDDLE_CLICK_EVENT
- CD_APPLET_REGISTER_FOR_DOUBLE_CLICK_EVENT
- CD_APPLET_REGISTER_FOR_SCROLL_EVENT
- CD_APPLET_REGISTER_FOR_BUILD_MENU_EVENT

Note: don't forget to unregister during the stop.

1.4.3 Go further

A lot of useful macros are provided in cairo-dock-applet-facility.h to make your life easier.

The applet instance is **myApplet**, and it holds the following:

- **myIcon** : this is your icon !
- **myContainer** : the container your icon belongs to (a Dock or a Desklet). For convenience, the following 2 parameters are available.
- **myDock** : if your container is a dock, myDock = myContainer, otherwise it is NULL.

- **myDesklet** : if your container is a desklet, myDesklet = myContainer, otherwise it is NULL.
- **myConfig** : the structure holding all the parameters you get in your config file. You have to define it in *applet-struct.h*.
- **myData** : the structure holding all the ressources loaded at run-time. You have to define it in *applet-struct.h*.
- **myDrawContext** : a cairo context, if you need to draw on the icon with the libcairo.
- To get values contained inside your **conf file**, you can use the following :
  CD_CONFIG_GET_BOOLEAN & cie
- To **build your menu**, you can use the following :
  CD_APPLET_ADD_SUB_MENU & cie
- To directly **set an image on your icon**, you can use the following :
  CD_APPLET_SET_IMAGE_ON_MY_ICON & cie
- To modify the **label** of your icon, you can use the following :
  CD_APPLET_SET_NAME_FOR_MY_ICON & cie
- To set a **quick-info** on your icon, you can use the following :
  CD_APPLET_SET_QUICK_INFO_ON_MY_ICON & cie
- To **create a surface** that fits your icon from an image, you can use the following :
  CD_APPLET_LOAD_SURFACE_FOR_MY_APPLET & cie
- To trigger the **refresh** of your icon or container after you drew something, you can use the following :
  CD_APPLET_REDRAW_MY_ICON & CAIRO_DOCK_REDRAW_MY_CONTAINER

1.4.4 How can I take advantage of the OpenGL ?

There are 3 cases :

- your applet just has a static icon; there is nothing to take into account, the common functions to set an image or a surface on an icon already handle the texture mapping.
- you draw dynamically on your icon with libcairo (using myDrawContext), but you don't want to bother with OpenGL; all you have to do is to call /ref cairo_dock_update_icon_texture to update your icon's texture after you drawn your surface. This can be done for occasional drawings, like Switcher redrawing its icon each time a window is moved.
- you draw your icon differently whether the dock is in OpenGL mode or not; in this case, you just need to put all the OpenGL commands into a CD_APPLET_START_DRAWING_MY_ICON/CD_APPLET_FINISH_DR↩
  AWING_MY_ICON section inside your code.

There are also a lot of convenient functions you can use to draw in OpenGL. See cairo-dock-draw-opengl.h for loading and drawing textures and paths, and cairo-dock-particle-system.h for an easy way to draw particle systems.

1.4.5 How can I animate my applet to make it more lively ?

If you want to animate your icon easily, to signal some action (like *Music-Player* when a new song starts), you can simply **request for one of the registered animations** with CD_APPLET_ANIMATE_MY_ICON and stop it with CD_APPLET_STOP_ANIMATING_MY_ICON. You just need to specify the name of the animation (like "rotate" or "pulse") and the number of time it will be played.

But you can also make your own animation, like *Clock* of *Cairo-Penguin*. You will have to integrate yourself into the rendering loop of your container. Don't panic, here again, Cairo-Dock helps you !

First you will register to the "update container" notification, with a simple call to CD_APPLET_REGISTER_FOR↩
_UPDATE_ICON_SLOW_EVENT or CD_APPLET_REGISTER_FOR_UPDATE_ICON_EVENT, depending on the

refresh frequency you need : ∼10Hz or ∼33Hz. A high frequency needs of course more CPU, and most of the time the slow frequancy is enough.

Then you will just put all your code in a CD_APPLET_ON_UPDATE_ICON_BEGIN/CD_APPLET_ON_UPDATE↩ _ICON_END section. That's all ! In this section, do what you want, like redrawing your icon, possibly incrementing a counter to know until where you went, etc. See the previous paragraph to draw on your icon. Inside the rendering loop, you can skip an iteration with CD_APPLET_SKIP_UPDATE_ICON, and quit the loop with CD_APPLET_↩ STOP_UPDATE_ICON or CD_APPLET_PAUSE_UPDATE_ICON (don't forget to quit the loop when you're done, otherwise your container may continue to redraw itself, which means a needless CPU load).

To know the size allocated to your icon, use the convenient CD_APPLET_GET_MY_ICON_EXTENT.

1.4.6 I have heavy treatments to do, how can I make them without slowing the dock ?

Say for instance you want to download a file on the Net, it is likely to take some amount of time, during which the dock will be frozen, waiting for you. To avoid such a situation, Cairo-Dock defines Tasks. They perform their job **asynchronously**, and can be **periodic**. See cairo-dock-task.h for a quick explanation on how a Task works.

You create a Task with cairo_dock_new_task, launch it with cairo_dock_launch_task, and either cancel it with cairo↩ _dock_discard_task or destroy it with cairo_dock_free_task.

1.4.7 Key binding

You can bind an action to a shortkey with the following macro: CD_APPLET_BIND_KEY.
For instance, the GMenu applet displays the menu on ctrl+F1.
You get a GldiShortkey that you simply destroy when the applet stops (with gldi_object_unref).

See cairo-dock-keybinder.h for more details.

1.4.8 I need more than one icon, how can I easily get more ?

In dock mode, your icon can have a sub-dock; in desklet mode, you can load a list of icons into your desklet. Cairo-Dock provides a convenient macro to **quickly load a list of icons** in both cases : CD_APPLET_LOAD_M↩ Y_ICONS_LIST to load a list of icons and CD_APPLET_DELETE_MY_ICONS_LIST to destroy it. Thus you don't need to know in which mode you are, neither to care about loading the icons, freeing them, or anything.

You can get the list of icons with CD_APPLET_MY_ICONS_LIST and to their container with CD_APPLET_MY_I↩ CONS_LIST_CONTAINER.

1.5 Advanced functionnalities

1.5.1 How can I make my own widgets in the config panel ?

Cairo-Dock can build itself the config panel of your applet from the config file. Moreover, it can do the opposite : update the conf file from the config panel. However, it is limited to the widgets it knows, and there are some cases it is not enough. Because of that, Cairo-Dock offers 2 hooks in the process of building/reading the config panel : when defining your applet in the CD_APPLET_DEFINE_BEGIN/CD_APPLET_DEFINE_END section, add to the interface the 2 functions pInterface->load_custom_widget and pInterface->save_custom_widget. They will be respectively called when the config panel of your applet is raised, and when it is validated.

If you want to modify the content of an existing widget, you can grab it with cairo_dock_gui_find_group_key_↩ widget_in_list. To add your custom widgets, insert in the conf file an empty widget (with the prefix '_'), then grab it and pack some GtkWidget inside. If you want to dynamically alter the config panel (like having a "new" button

that would make appear new widgets on click), you can add in the conf file the new widgets, and then call cairo_↩
dock_reload_current_module_widget to reload the config panel. See the AlsaMixer or Weather applets for an easy example, and Clock or Mail for a more advanced example.

1.5.2 How can my applet control the window of an application ?

Say your applet launches an external application that has its own window. It is logical to **make your applet control this application**, rather than letting the Taskbar do. All you need to do is to call the macro CD_APPLET_MANAG↩
E_APPLICATION, indicating which application you wish to manage (you need to enter the class of the application, as you can get from "xprop | grep CLASS"). Your applet will then behave like a launcher that has stolen the appli icon.

1.5.3 How can I render some numerical values on my icon ?

Cairo-Dock offers a powerful and versatile architecture for this case : _CairoDataRenderer. A DataRenderer is a generic way to render a set of values on an icon; there are several implementations of this class : Gauge, Cairo↩
DockGraph, Bar, and it is quite easy to implement a new kind of DataRenderer.

Each kind of renderer has a set of attributes that you can use to customize it; you just need to call the CD_APPLE↩
T_ADD_DATA_RENDERER_ON_MY_ICON macro with the attributes, and you're done ! Then, each time you want to render some new values, simply call CD_APPLET_RENDER_NEW_DATA_ON_MY_ICON with the new values.

When your applet is reloaded, you have to reload the DataRenderer as well, using the convenient CD_APPLE↩
T_RELOAD_MY_DATA_RENDERER macro. If you don't specify attributes to it, it will simply reload the current DataRenderer, otherwise it will load the new attributes; the previous data are not lost, which is useful in the case of Graph for instance.

You can remove it at any time with CD_APPLET_REMOVE_MY_DATA_RENDERER.

1.5.4 How can I make my applet multi-instanciable ?

Applets can be launched several times, an instance will be created each time. To ensure your applet can be instanciated several times, you just need to pass myApplet to any function that uses one of its fields (myData, myIcon, etc). Then, to indicate Cairo-Dock that your applet is multi-instanciable, you'll have to define the macro CD_APPLET_MULTI_INSTANCE in each file. A convenient way to do that is to define it in the CMakeLists.txt by adding the following line:

```
add_definitions (-DCD_APPLET_MULTI_INSTANCE="1")
```

.

1.5.5 How can I draw anywhere on the dock, not only on my icon ?

Say you want to draw directly on your container, like *CairoPenguin* or *ShowMouse* do. This can be achieved easily by registering to the NOTIFICATION_RENDER notification. You will then be notified eash time a Dock or a Desklet is drawn. Register AFTER so that you will draw after the view.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 _CairoDataRenderer Struct Reference

Generic DataRenderer. Any implementation of a DataRenderer will derive from this class.

Data Fields

- CairoDataRendererInterface interface

    interface of the Data Renderer.
- CairoDataToRenderer data

    internal data to be drawn by the renderer.
- gint iWidth

    size of the drawing area.
- CairoDataRendererFormatValueFunc format_value

    specific function to format the values as text.
- gchar cFormatBuffer [CAIRO_DOCK_DATA_FORMAT_MAX_LEN+1]

    buffer for the text.
- gpointer pFormatData

    data passed to the format fonction.
- gboolean bUpdateMinMax

    TRUE <=> the Data Renderer should dynamically update the range of the values.
- gboolean bWriteValues

    TRUE <=> the Data Renderer should write the values as text itself.
- gint iLatencyTime

    the time it will take to update to the new value, with a smooth animation (require openGL capacity)
- gint iRank

    the rank of the renderer, eg the number of values it can display at once (for exemple, 1 for a bar, 2 for a dual-gauge)
- gboolean bCanRenderValueAsText

    set to TRUE <=> the renderer can draw the values as text itself.
- gboolean bRotateWithContainer

    set to TRUE <=> the drawing will be rotated if the container is vertical.
- RendererRotateTheme iRotateTheme

    an option to rotate applet, no, automatic or always.
- gboolean bisRotate

    set to TRUE <=> the theme images are rotated 90°clockwise.
- gboolean bUseOverlay

    whether the data-renderer draws on an overlay rather than directly on the icon.

- CairoOverlayPosition iOverlayPosition

  position of the overlay, in the case the renderer uses one.
- CairoDataRendererText ∗ pLabels

  an optionnal list of labels to be displayed on the Data Renderer to indicate the nature of each value. Same size as the set of values.
- CairoDataRendererEmblem ∗ pEmblems

  an optionnal list of emblems to be displayed on the Data Renderer to indicate the nature of each value. Same size as the set of values.
- CairoDataRendererTextParam ∗ pValuesText

  an optionnal list of text zones to write the values. Same size as the set of values.
- gint iSmoothAnimationStep

  the animation counter for the smooth movement.
- gdouble fLatency

  latency due to the smooth movement (0 means the displayed value is the current one, 1 the previous)

4.1.1 Detailed Description

Generic DataRenderer. Any implementation of a DataRenderer will derive from this class.

The documentation for this struct was generated from the following file:

- cairo-dock-data-renderer.h

4.2 _CairoDataRendererAttribute Struct Reference

Generic DataRenderer attributes structure. The attributes of any implementation of a DataRenderer will derive from this class.

Data Fields

- const gchar ∗ cModelName

  name of the model ("gauge", "graph", etc) [mandatory].
- gint iNbValues

  number of values to represent (for instance 3 for (cpu, mem, swap)) [1 by default and minimum].
- gint iMemorySize

  number of values to remember over time. For instance graphs can display as much values as the icon's width [2 by default and minimum].
- gdouble ∗ pMinMaxValues

  an array of pairs of (min,max) values. [optionnal, input values will be considered between 0 and 1 if NULL].
- gboolean bUpdateMinMax

  whether to automatically update the values' range [false by default].
- gboolean bWriteValues

  whether to write the values on the icon. [false by default].
- RendererRotateTheme iRotateTheme

  an option to rotate applet, no, automatic or always.
- gint iLatencyTime

  time needed to update to the new values. The update is smooth in OpenGL mode. [0 by default]
- CairoDataRendererFormatValueFunc format_value

  a function used to format the values into a string. Only useful if you make te DataRenderer write the values [optionnal, by default the values are formatted with 2 decimals].
- gpointer pFormatData

data to be passed to the format function [optionnal].

- gchar ** cEmblems

  an optionnal list of emblems to draw on the overlay.
- gchar ** cLabels

  an optionnal list of labels to write on the overlay.

4.2.1 Detailed Description

Generic DataRenderer attributes structure. The attributes of any implementation of a DataRenderer will derive from this class.

The documentation for this struct was generated from the following file:

- cairo-dock-data-renderer.h

4.3 _CairoDataRendererInterface Struct Reference

Interface of a DataRenderer.

Data Fields

- CairoDataRendererLoadFunc load

  function that loads anything the DataRenderer will need. It also completes the DataRenderer structure (for instance the text zones).
- CairoDataRendererRenderFunc render

  function that draws the values with cairo.
- CairoDataRendererRenderOpenGLFunc render_opengl

  function that draws the values with opengl.
- CairoDataRendererReloadFunc reload

  function that reloads the DataRenderer's buffers when the icon is resized.
- CairoDataRendererUnloadFunc unload

  function that unload all the previously allocated buffers.

4.3.1 Detailed Description

Interface of a DataRenderer.

The documentation for this struct was generated from the following file:

- cairo-dock-data-renderer.h

4.4 _CairoDesklet Struct Reference

Definition of a Desklet, which derives from a Container.

4.4.1 Detailed Description

Definition of a Desklet, which derives from a Container.

The documentation for this struct was generated from the following file:

- cairo-dock-desklet-factory.h

4.5 _CairoDeskletAttr Struct Reference

Configuration attributes of a Desklet.

4.5.1 Detailed Description

Configuration attributes of a Desklet.

The documentation for this struct was generated from the following file:

- cairo-dock-desklet-factory.h

4.6 _CairoDeskletDecoration Struct Reference

Decoration of a Desklet.

4.6.1 Detailed Description

Decoration of a Desklet.

The documentation for this struct was generated from the following file:

- cairo-dock-desklet-factory.h

4.7 _CairoDeskletRenderer Struct Reference

Definition of a Desklet's renderer.

Data Fields

- CairoDeskletRenderFunc render

  rendering function with libcairo.
- CairoDeskletGLRenderFunc render_opengl

  rendering function with OpenGL.
- CairoDeskletConfigureRendererFunc configure

  get the configuration of the renderer from a set of config attributes.
- CairoDeskletLoadRendererDataFunc load_data

  load the internal data of the renderer.
- CairoDeskletFreeRendererDataFunc free_data

  free all internal data of the renderer.
- CairoDeskletCalculateIconsFunc calculate_icons

  define the icons' size and load them.
- CairoDeskletUpdateRendererDataFunc update

  function called on each iteration of the rendering loop.
- CairoDeskletGLRenderFunc render_bounding_box

  optionnal rendering function with OpenGL that only draws the bounding boxes of the icons (for picking).
- GList $*$ pPreDefinedConfigList

  An optionnal list of preset configs.

4.7.1 Detailed Description

Definition of a Desklet's renderer.

The documentation for this struct was generated from the following file:

- cairo-dock-desklet-factory.h

4.8 _CairoDialog Struct Reference

Definition of a Dialog.

Data Fields

- GldiContainer container

  container.

4.8.1 Detailed Description

Definition of a Dialog.

The documentation for this struct was generated from the following file:

- cairo-dock-dialog-factory.h

4.9 _CairoDialogDecorator Struct Reference

Definition of a Dialog/Menu decorator. It draws the frame of the Dialog/Menu.

Data Fields

- CairoDialogSetDecorationSizeFunc set_size

  defines the various margins and alignment of the dialog
- CairoDialogRenderDecorationFunc render

  draw the dialog's frame (outline and background)
- CairoMenuSetupFunc setup_menu

  defines the GldiMenuParams of the menu (radius, alignment, arrow height)
- CairoMenuRenderFunc render_menu

  draw the menu's frame (outline and background); in the end, must clip the shape of the frame on the context
- const gchar ∗ cDisplayedName

  readable name of the decorator

4.9.1 Detailed Description

Definition of a Dialog/Menu decorator. It draws the frame of the Dialog/Menu.

The documentation for this struct was generated from the following file:

- cairo-dock-dialog-factory.h

4.10 _CairoDialogRenderer Struct Reference

Definition of a Dialog renderer. It draws the inside of the Dialog.

4.10.1 Detailed Description

Definition of a Dialog renderer. It draws the inside of the Dialog.

The documentation for this struct was generated from the following file:

- cairo-dock-dialog-factory.h

4.11 _CairoDock Struct Reference

Definition of a Dock, which derives from a Container.

Data Fields

- GldiContainer container

  container.
- GList ∗ icons

  the list of icons.
- gboolean bIsMainDock

  Set to TRUE for the main dock (the first to be created, and the one containing the taskbar).
- gint iRefCount

  number of icons pointing on the dock (0 means it is a root dock, >0 a sub-dock).
- gchar ∗ cDockName

  unique name of the dock
- CairoDockVisibility iVisibility

  visibility.
- gint iNumScreen

  number of the screen the dock is placed on (-1 <=> all screen, >0 <=> num screen).
- gint iIconSize

  icon size, as specified in the config of the dock
- gboolean bGlobalIconSize

  whether the dock should use the global icons size parameters.
- gboolean bGlobalBg

  whether the dock should use the global background parameters.
- gchar ∗ cBgImagePath

  path to an image, or NULL
- gboolean bBgImageRepeat

  whether to repeat the image as a pattern, or to stretch it to fill the dock.
- GldiColor fBgColorBright

  first color of the gradation
- GldiColor fBgColorDark

  second color of the gradation
- CairoDockImageBuffer backgroundBuffer

  Background image buffer of the dock.
- gdouble fFoldingFactor

  (un)folding factor, between 0(unfolded) to 1(folded). It's up to the renderer on how to make use of it.

- gdouble fHideOffset

  counter for auto-hide.
- gdouble fPostHideOffset

  counter for the post-hiding animation for icons always visible.
- gboolean bIsBelow

  Whether the dock is in a popped up state or not.
- gint bHasModalWindow

  TRUE if the dock has a modal window (menu, dialog, etc), that will block it.
- gboolean bIsDragging

  whether the user is dragging something over the dock.
- gboolean bTemporaryHidden

  Backup of the auto-hide state before quick-hide.
- gboolean bEntranceDisabled

  whether mouse can't enter into the dock.
- gboolean bIsShrinkingDown

  whether the dock is shrinking down.
- gboolean bIsGrowingUp

  whether the dock is growing up.
- gboolean bIsHiding

  whether the dock is hiding.
- gboolean bIsShowing

  whether the dock is showing.
- gboolean bIconIsFlyingAway

  whether an icon is being dragged away from the dock
- gboolean bPreventDraggingIcons

  whether icons in the dock can be dragged with the mouse (inside and outside of the dock).
- gdouble iMaxIconHeight

  maximum height of the icons.
- gdouble fFlatDockWidth

  width of the dock, only taking into account an alignment of the icons.
- guint iSidMoveResize

  Source ID for window resizing.
- guint iSidUnhideDelayed

  Source ID for window popping down to the bottom layer.
- guint iSidLeaveDemand

  Source ID of the timer that delays the "leave" event.
- guint iSidUpdateWMIcons

  Source ID for pending update of WM icons geometry.
- guint iSidHideBack

  Source ID for hiding back the dock.
- guint iSidLoadBg

  Source ID for loading the background.
- guint iSidDestroyEmptyDock

  Source ID to destroy an empty main dock.
- guint iSidTestMouseOutside

  Source ID for shrinking down the dock after a mouse event.
- guint iSidUpdateDockSize

  Source ID for updating the dock's size and icons layout.
- CairoDockRenderer $*$ pRenderer

  current renderer, never NULL.
- gpointer pRendererData

data that can be used by the renderer.

- gboolean bCanDrop

  Set to TRUE by the renderer if one can drop between 2 icons.
- CairoDockMousePositionType iMousePositionType

  set by the view to say if the mouse is currently on icons, on the egde, or outside of icons.
- gint iMinDockWidth

  width of the dock at rest.
- gint iMinDockHeight

  height of the dock at rest.
- gint iMaxDockWidth

  maximum width of the dock.
- gint iMaxDockHeight

  maximum height of the dock.
- gint iDecorationsWidth

  width of background decorations, set by the renderer.
- gint iDecorationsHeight

  height of background decorations, set by the renderer.
- gdouble fMagnitudeMax

  maximal magnitude of the zoom, between 0 and 1.
- gint iActiveWidth

  width of the active zone of the dock.
- gint iActiveHeight

  height of the active zone of the dock.
- CairoDockInputState iInputState

  state of the input shape (active, at rest, hidden).
- cairo_region_t ∗ pShapeBitmap

  input shape of the window when the dock is at rest.
- cairo_region_t ∗ pHiddenShapeBitmap

  input shape of the window when the dock is hidden.
- cairo_region_t ∗ pActiveShapeBitmap

  input shape of the window when the dock is active (NULL to cover all dock).

4.11.1 Detailed Description

Definition of a Dock, which derives from a Container.

The documentation for this struct was generated from the following file:

- cairo-dock-dock-factory.h

4.12 _CairoDockClassAppli Struct Reference

Definition of a Class of application.

Data Fields

- gboolean bUseXIcon

  TRUE if the appli must use the icon provided by X instead the one from the theme.
- gboolean bExpand

  TRUE if the appli doesn't group togather with its class.

- GList ∗ pIconsOfClass

  List of the inhibitors of the class.
- GList ∗ pAppliOfClass

  List of the appli icons of this class.

4.12.1 Detailed Description

Definition of a Class of application.

The documentation for this struct was generated from the following file:

- cairo-dock-class-manager.h

4.13 _CairoDockDesktopEnvBackend Struct Reference

Definition of the Desktop Environment backend.

4.13.1 Detailed Description

Definition of the Desktop Environment backend.

The documentation for this struct was generated from the following file:

- cairo-dock-file-manager.h

4.14 _CairoDockGLConfig Struct Reference

This strucure summarizes the available OpenGL configuration on the system.

4.14.1 Detailed Description

This strucure summarizes the available OpenGL configuration on the system.

The documentation for this struct was generated from the following file:

- cairo-dock-opengl.h

4.15 _CairoDockGLFont Struct Reference

Structure used to load a font for OpenGL text rendering.

4.15.1 Detailed Description

Structure used to load a font for OpenGL text rendering.

The documentation for this struct was generated from the following file:

- cairo-dock-opengl-font.h

4.16 _CairoDockGLPath Struct Reference

Definition of a CairoDockGLPath.

4.16.1 Detailed Description

Definition of a CairoDockGLPath.

The documentation for this struct was generated from the following file:

- cairo-dock-opengl-path.h

4.17 _CairoDockGroupKeyWidget Struct Reference

Definition of a widget corresponding to a given (group;key) pair.

4.17.1 Detailed Description

Definition of a widget corresponding to a given (group;key) pair.

The documentation for this struct was generated from the following file:

- cairo-dock-gui-factory.h

4.18 _CairoDockGuiBackend Struct Reference

Definition of the GUI interface for modules.

Data Fields

- void(∗ set_status_message_on_gui)(const gchar ∗cMessage)

  display a message on the GUI.
- void(∗ reload_current_widget)(GldiModuleInstance ∗pModuleInstance, int iShowPage)

  Reload the current config window from the conf file. iShowPage is the page that should be displayed in case the module has several pages, -1 means to keep the current page.
- CairoDockGroupKeyWidget ∗(∗ get_widget_from_name)(GldiModuleInstance ∗pModuleInstance, const gchar ∗cGroupName, const gchar ∗cKeyName)

  retrieve the widgets in the current module window, corresponding to the (group,key) pair in its conf file.

4.18.1 Detailed Description

Definition of the GUI interface for modules.

The documentation for this struct was generated from the following file:

- cairo-dock-gui-manager.h

4.19 _CairoDockHidingEffect Struct Reference

Definition of a Hiding Effect backend (used to provide an animation when the docks hides/shows itself).

Data Fields

- const gchar ∗ cDisplayedName

  translated name of the effect
- gboolean bCanDisplayHiddenDock

  whether the backend can display the dock even when it's hidden
- void(∗ pre_render)(CairoDock ∗pDock, double fOffset, cairo_t ∗pCairoContext)

  function called before the icons are drawn (cairo)
- void(∗ pre_render_opengl)(CairoDock ∗pDock, double fOffset)

  function called before the icons are drawn (opengl)
- void(∗ post_render)(CairoDock ∗pDock, double fOffset, cairo_t ∗pCairoContext)

  function called afer the icons are drawn (cairo)
- void(∗ post_render_opengl)(CairoDock ∗pDock, double fOffset)

  function called afer the icons are drawn (opengl)
- void(∗ init)(CairoDock ∗pDock)

  function called when the animation is started.

4.19.1 Detailed Description

Definition of a Hiding Effect backend (used to provide an animation when the docks hides/shows itself).

The documentation for this struct was generated from the following file:

- cairo-dock-animations.h

4.20 _CairoDockImageBuffer Struct Reference

Definition of an Image Buffer. It provides an unified interface for a cairo/opengl image buffer.

4.20.1 Detailed Description

Definition of an Image Buffer. It provides an unified interface for a cairo/opengl image buffer.

The documentation for this struct was generated from the following file:

- cairo-dock-image-buffer.h

4.21 _CairoDockPackage Struct Reference

Definition of a generic package.

Data Fields

- gchar ∗ cPackagePath

  complete path of the package.
- gdouble fSize

  size in Mo
- gchar ∗ cAuthor

  author(s)
- gchar ∗ cDisplayedName

name of the package

- CairoDockPackageType iType

    type of package : installed, user, distant.
- gint iRating

    rating of the package.
- gint iSobriety

    sobriety/simplicity of the package.
- gchar ∗ cHint

    hint of the package, for instance "sound" or "battery" for a gauge, "internet" or "desktop" for a third-party applet.
- gint iCreationDate

    date of creation of the package.
- gint iLastModifDate

    date of latest changes in the package.

4.21.1 Detailed Description

Definition of a generic package.

The documentation for this struct was generated from the following file:

- cairo-dock-packages.h

4.22 _CairoDockRenderer Struct Reference

Dock’s renderer, also known as ’view’.

Data Fields

- CairoDockComputeSizeFunc compute_size

    function that computes the sizes of a dock.
- CairoDockCalculateIconsFunc calculate_icons

    function that computes all the icons’ parameters.
- CairoDockRenderFunc render

    rendering function (cairo)
- CairoDockRenderOptimizedFunc render_optimized

    optimized rendering function (cairo) that only redraw a part of the dock.
- CairoDockGLRenderFunc render_opengl

    rendering function (OpenGL, optionnal).
- CairoDockSetSubDockPositionFunc set_subdock_position

    function that computes the position of the dock when it’s a sub-dock.
- CairoDockRenderFreeDataFunc free_data

    function called when the renderer is unset from the dock.
- CairoDockSetInputShapeFunc update_input_shape

    function called when the input zones are defined.
- CairoDockSetIconSizeFunc set_icon_size

    function called to define the size of an icon, or NULL to let the container handles that.
- gboolean bUseStencil

    TRUE if the view uses the OpenGL stencil buffer.
- gboolean bUseReflect

    TRUE is the view uses reflects.

- const gchar ∗ cDisplayedName

    name displayed in the GUI (translated).
- gchar ∗ cReadmeFilePath

    path to a readme file that gives a short description of the view.
- gchar ∗ cPreviewFilePath

    path to a preview image.

4.22.1 Detailed Description

Dock's renderer, also known as 'view'.

The documentation for this struct was generated from the following file:

- cairo-dock-dock-factory.h

4.23 _CairoDockTransition Struct Reference

Transitions are an easy way to set an animation on an Icon to make it change from a state to another.

Data Fields

- CairoDockTransitionRenderFunc render

    the cairo rendering function.
- CairoDockTransitionGLRenderFunc render_opengl

    the openGL rendering function (can be NULL, in which case the texture mapping from the cairo drawing is done automatically).
- gpointer pUserData

    data passed to the rendering functions.
- GFreeFunc pFreeUserDataFunc

    function called to destroy the data when the transition is deleted.
- gboolean bFastPace

    TRUE <=> the transition will be in the fast loop (high frequency refresh).
- gboolean bRemoveWhenFinished

    TRUE <=> the transition will be destroyed and removed from the icon when finished.
- gint iDuration

    duration if the transition, in ms. Can be 0 for an endless transition.
- gint iElapsedTime

    elapsed time since the beginning of the transition, in ms.
- gint iCount

    number of setps since the beginning of the transition, in ms.
- GldiContainer ∗ pContainer

    Container of the Icon.

4.23.1 Detailed Description

Transitions are an easy way to set an animation on an Icon to make it change from a state to another.

The documentation for this struct was generated from the following file:

- cairo-dock-animations.h

4.24 _CairoGraphAttribute Struct Reference

Attributes of a Graph.

Data Fields

- CairoDataRendererAttribute rendererAttribute

    General attributes of any DataRenderer.
- CairoDockTypeGraph iType

    type of graph
- gdouble ∗ fHighColor

    color of the high values. it's a table of nb_values triplets, each of them representing an rgb color.
- gdouble ∗ fLowColor

    color of the low values. same as fHighColor.
- gdouble fBackGroundColor [4]

    color of the background.
- gboolean bMixGraphs

    TRUE to draw all the values on the same graph.

4.24.1 Detailed Description

Attributes of a Graph.

The documentation for this struct was generated from the following file:

- cairo-dock-graph.h

4.25 _CairoIconContainerRenderer Struct Reference

Definition of an Icon container (= an icon holding a sub-dock) renderer.

4.25.1 Detailed Description

Definition of an Icon container (= an icon holding a sub-dock) renderer.

The documentation for this struct was generated from the following file:

- cairo-dock-icon-factory.h

4.26 _CairoOverlay Struct Reference

Definition of an Icon Overlay.

Data Fields

- GldiObject object

    object
- CairoDockImageBuffer image

    image buffer

- CairoOverlayPosition iPosition

  position on the icon
- gdouble fScale

  scale at which to draw the overlay, relatively to the icon (0.5 by default, 1 will cover the whole icon, 0 means to draw at the actual buffer size).
- Icon ∗ pIcon

  icon it belongs to.
- gpointer data

  data used to identify an overlay

4.26.1 Detailed Description

Definition of an Icon Overlay.

The documentation for this struct was generated from the following file:

- cairo-dock-overlay.h

4.27 _CairoParticle Struct Reference

A particle of a particle system.

Data Fields

- GLfloat x

  horizontal position, in fraction of the particle system's width, and relatively to the center of the particle system. So it is comprised between -1 and 1.
- GLfloat y

  vertical position, in fraction of the particle system's height, and relatively to the bottom of the particle system. So it is comprised between 0 and 1.
- GLfloat z

  depth of the particle, negative to be "behind". 0 means it is at the same depth as icons.
- GLfloat vx

  horizontal speed
- GLfloat vy

  vertical speed
- GLfloat fWidth

  size
- GLfloat color [4]

  color r,g,b,a
- GLfloat fOscillation

  phase of the oscillations.
- GLfloat fOmega

  oscillation variation speed.
- GLfloat fSizeFactor

  current size factor
- GLfloat fResizeSpeed

  size variation speed.
- gint iLife

  current life time, decreased by 1 at each step.
- gint iInitialLife

  total life time.

4.27.1 Detailed Description

A particle of a particle system.

The documentation for this struct was generated from the following file:

- cairo-dock-particle-system.h

4.28 _CairoParticleSystem Struct Reference

A particle system.

4.28.1 Detailed Description

A particle system.

The documentation for this struct was generated from the following file:

- cairo-dock-particle-system.h

4.29 _CairoProgressBarAttribute Struct Reference

Attributes of a PgrogressBar.

Data Fields

- CairoDataRendererAttribute rendererAttribute

  General attributes of any DataRenderer.
- gchar ∗ cImageGradation

  image or NULL
- gdouble ∗ fColorGradation

  color gradation of the bar (an array of 8 doubles, representing 2 RGBA values) or NULL
- gboolean bUseCustomPosition

  TRUE to define a custom position (by default it is placed at the middle bottom)
- CairoOverlayPosition iCustomPosition

  custom position
- gboolean bInverted

  invert default colors

4.29.1 Detailed Description

Attributes of a PgrogressBar.

The documentation for this struct was generated from the following file:

- cairo-dock-progressbar.h

4.30 _GldiContainer Struct Reference

Definition of a Container, whom derive Dock, Desklet, Dialog and FlyingContainer.

Data Fields

- GldiObject object

  object.
- gpointer pDataSlot [CAIRO_DOCK_NB_DATA_SLOT]

  External data.
- GtkWidget ∗ pWidget

  window of the container.
- gint iWidth

  size of the container.
- gint iWindowPositionX

  position of the container.
- gboolean bInside

  TURE is the mouse is inside the container (including the possible sub-widgets).
- CairoDockTypeHorizontality bIsHorizontal

  TRUE if the container is horizontal, FALSE if vertical.
- gboolean bDirectionUp

  TRUE if the container is oriented upwards, FALSE if downwards.
- guint iSidGLAnimation

  Source ID of the animation loop.
- gint iAnimationDeltaT

  interval of time between 2 animation steps.
- gint iMouseX

  X position of the mouse in the container's system of reference.
- gint iMouseY

  Y position of the mouse in the container's system of reference.
- gdouble fRatio

  zoom applied to the container's elements.
- gboolean bUseReflect

  TRUE if the container has a reflection power.
- GLXContext glContext

  OpenGL context.
- gboolean bPerspectiveView

  whether the GL context is an ortho or a perspective view.
- gboolean bKeepSlowAnimation

  TRUE if a slow animation is running.
- gint iAnimationStep

  counter for the animation loop.

4.30.1 Detailed Description

Definition of a Container, whom derive Dock, Desklet, Dialog and FlyingContainer.

The documentation for this struct was generated from the following file:

- cairo-dock-container.h

4.31 _GldiContainerManagerBackend Struct Reference

Definition of the Container backend. It defines some operations that should be, but are not, provided by GTK.

4.31.1 Detailed Description

Definition of the Container backend. It defines some operations that should be, but are not, provided by GTK.

The documentation for this struct was generated from the following file:

- cairo-dock-container.h

4.32 _GldiDesktopBackground Struct Reference

Definition of a Desktop Background Buffer. It has a reference count so that it can be shared across all the lib.

4.32.1 Detailed Description

Definition of a Desktop Background Buffer. It has a reference count so that it can be shared across all the lib.

The documentation for this struct was generated from the following file:

- cairo-dock-desktop-manager.h

4.33 _GldiDesktopManagerBackend Struct Reference

Definition of the Desktop Manager backend.

4.33.1 Detailed Description

Definition of the Desktop Manager backend.

The documentation for this struct was generated from the following file:

- cairo-dock-desktop-manager.h

4.34 _GldiManager Struct Reference

Definition of a Manager.

Data Fields

- GldiObject object

  object
- GldiManagerInitFunc init

  function called once and for all at the init of the core.
- GldiManagerLoadFunc load

  function called when loading the current theme, after getting the config
- GldiManagerUnloadFunc unload

  function called when unloading the current theme, before resetting the config.
- GldiManagerReloadFunc reload

  function called when reloading a part of the current theme.
- GldiManagerGetConfigFunc get_config

  function called when getting the config of the current theme, or a part of it.
- GldiManagerResetConfigFunc reset_config

  function called when resetting the current theme, or a part of it.

4.34.1 Detailed Description

Definition of a Manager.

The documentation for this struct was generated from the following file:

- cairo-dock-manager.h

4.35 _GldiModule Struct Reference

Definition of an external module.

Data Fields

- GldiObject object

  object
- GldiModuleInterface ∗ pInterface

  interface of the module.
- GldiVisitCard ∗ pVisitCard

  visit card of the module.
- gchar ∗ cConfFilePath

  conf file of the module.
- gpointer handle

  if the module interface is provided by a dynamic library, handle to this library.
- GList ∗ pInstancesList

  list of instances of the module.

4.35.1 Detailed Description

Definition of an external module.

The documentation for this struct was generated from the following file:

- cairo-dock-module-manager.h

4.36 _GldiModuleInstance Struct Reference

Definition of an instance of a module. A module can be instanciated several times.

Data Fields

- GldiObject object

  object
- GldiModule ∗ pModule

  the module this instance represents.
- gchar ∗ cConfFilePath

  conf file of the instance.
- gboolean bCanDetach

  TRUE if the instance can be detached from docks (desklet mode).
- Icon ∗ pIcon

the icon holding the instance.
- GldiContainer ∗ pContainer

    container of the icon.
- CairoDock ∗ pDock

    this field repeats the 'pContainer' field if the container is a dock, and is NULL otherwise.
- CairoDesklet ∗ pDesklet

    this field repeats the 'pContainer' field if the container is a desklet, and is NULL otherwise.
- cairo_t ∗ pDrawContext

    a drawing context on the icon.
- gint iSlotID

    a unique ID to insert external data on icons and containers.
- gpointer pConfig

    pointer to a structure containing the config parameters of the applet.
- gpointer pData

    pointer to a structure containing the data of the applet.

4.36.1 Detailed Description

Definition of an instance of a module. A module can be instanciated several times.

The documentation for this struct was generated from the following file:

- cairo-dock-module-instance-manager.h

4.37 _GldiModuleInterface Struct Reference

Definition of the interface of a module.

4.37.1 Detailed Description

Definition of the interface of a module.

The documentation for this struct was generated from the following file:

- cairo-dock-module-manager.h

4.38 _GldiObject Struct Reference

Definition of an Object.

4.38.1 Detailed Description

Definition of an Object.

The documentation for this struct was generated from the following file:

- cairo-dock-object.h

4.39 _GldiObjectManager Struct Reference

Definition of an ObjectManager.

4.39.1 Detailed Description

Definition of an ObjectManager.

The documentation for this struct was generated from the following file:

- cairo-dock-object.h

4.40 _GldiTask Struct Reference

Definition of a periodic and/or asynchronous Task.

Data Fields

- guint iPeriod

  interval of time in seconds, 0 if the Task is to run once.
- gpointer pSharedMemory

  structure passed as parameter of the 'get_data' and 'update' functions. Must not be accessed outside of these 2 functions !
- gboolean bDiscard

  TRUE when the task has been discarded.

4.40.1 Detailed Description

Definition of a periodic and/or asynchronous Task.

The documentation for this struct was generated from the following file:

- cairo-dock-task.h

4.41 _GldiTextDescription Struct Reference

Description of the rendering of a text.

Data Fields

- gchar ∗ cFont

  font.
- PangoFontDescription ∗ fd

  pango font
- gint iSize

  size in pixels
- gboolean bNoDecorations

  whether to draw the decorations (frame and outline) or not
- gboolean bUseDefaultColors

  whether to use the default colors or the colors defined below
- GldiColor fColorStart

  text color
- GldiColor fBackgroundColor

  background color

- GldiColor fLineColor

    outline color

- gboolean bOutlined

    TRUE to stroke the outline of the characters (in black).

- gint iMargin

    margin around the text, it is also the dimension of the frame if available.

- gboolean bUseMarkup

    *whether to use Pango markups or not (markups are html-like marks, like **...**; using markups force you to escape some characters like "&" -> "&amp;")*

- gdouble fMaxRelativeWidth

    maximum width allowed, in ratio of the screen's width. Carriage returns will be inserted if necessary. 0 means no limit.

4.41.1 Detailed Description

Description of the rendering of a text.

The documentation for this struct was generated from the following file:

- cairo-dock-style-facility.h

4.42 _GldiVisitCard Struct Reference

Definition of the visit card of a module. Contains everything that is statically defined for a module.

4.42.1 Detailed Description

Definition of the visit card of a module. Contains everything that is statically defined for a module.

The documentation for this struct was generated from the following file:

- cairo-dock-module-manager.h

4.43 _GldiWindowActor Struct Reference

Definition of a window actor.

Data Fields

- gboolean bIsHidden

    not used yet...

4.43.1 Detailed Description

Definition of a window actor.

The documentation for this struct was generated from the following file:

- cairo-dock-windows-manager.h

4.44 _GldiWindowManagerBackend Struct Reference

Definition of the Windows Manager backend.

4.44.1 Detailed Description

Definition of the Windows Manager backend.

The documentation for this struct was generated from the following file:

- cairo-dock-windows-manager.h

4.45 _Icon Struct Reference

Definition of an Icon.

Data Fields

- GldiObject object
  object
- CairoDockIconGroup iGroup
  group of the icon.
- IconInterface iface
  interface
- gchar ∗ cName
  Name of the icon.
- gchar ∗ cQuickInfo
  Short info displayed on the icon (few characters).
- gchar ∗ cFileName
  name or path of an image displayed on the icon.
- gchar ∗ cClass
  Class of application the icon will be bound to.
- gchar ∗ cParentDockName
  name of the dock the icon belongs to (NULL means it's not currently inside a dock).
- CairoDock ∗ pSubDock
  Sub-dock the icon is pointing to.
- gdouble fOrder
  Order of the icon amongst the other icons of its group.
- gboolean bStatic
  a hint to indicate the icon should be kept static (no animation like bouncing).
- gboolean bAlwaysVisible
  a flag that allows the icon to be always visible, even when the dock is hidden.
- gboolean bPointed
  Whether the icon is currently pointed or not.

4.45.1 Detailed Description

Definition of an Icon.

The documentation for this struct was generated from the following file:

- cairo-dock-icon-factory.h

4.46 _IconInterface Struct Reference

Icon's interface.

Data Fields

- void(∗ load_image)(Icon ∗icon)

  function that loads the icon surface (and optionnally texture).
- void(∗ action_on_drag_hover)(Icon ∗icon)

  function called when the user drag something over the icon for more than 500ms.

4.46.1 Detailed Description

Icon's interface.

The documentation for this struct was generated from the following file:

- cairo-dock-icon-factory.h

Chapter 5

File Documentation

5.1 cairo-dock-animations.h File Reference

Data Structures

- struct _CairoDockTransition

    Transitions are an easy way to set an animation on an Icon to make it change from a state to another.
- struct _CairoDockHidingEffect

    Definition of a Hiding Effect backend (used to provide an animation when the docks hides/shows itself).

Macros

- #define cairo_dock_container_is_animating(pContainer)
- #define cairo_dock_animation_will_be_visible(pDock)
- #define gldi_icon_stop_animation(pIcon)
- #define cairo_dock_get_animation_delta_t(pContainer)
- #define cairo_dock_get_slow_animation_delta_t(pContainer)
- #define cairo_dock_has_transition(pIcon)
- #define cairo_dock_get_transition_count(pIcon)
- #define cairo_dock_get_transition_elapsed_time(pIcon)
- #define cairo_dock_get_transition_fraction(pIcon)

Typedefs

- typedef gboolean(∗ CairoDockTransitionRenderFunc)(Icon ∗pIcon, gpointer pUserData)

    callback to render the icon with libcairo at each step of the Transition.
- typedef gboolean(∗ CairoDockTransitionGLRenderFunc)(Icon ∗pIcon, gpointer pUserData)

    callback to render the icon with OpenGL at each step of the Transition.

Functions

- void cairo_dock_pop_up (CairoDock ∗pDock)
- void cairo_dock_pop_down (CairoDock ∗pDock)
- void cairo_dock_launch_animation (GldiContainer ∗pContainer)
- void gldi_icon_start_animation (Icon ∗icon)
- void gldi_icon_request_animation (Icon ∗pIcon, const gchar ∗cAnimation, int iNbRounds)
- void gldi_icon_request_attention (Icon ∗pIcon, const gchar ∗cAnimation, int iNbRounds)
- void gldi_icon_stop_attention (Icon ∗pIcon)

- void cairo_dock_trigger_icon_removal_from_dock (Icon ∗pIcon)
- void cairo_dock_set_transition_on_icon (Icon ∗pIcon, GldiContainer ∗pContainer, CairoDockTransition↩ RenderFunc render_step_cairo, CairoDockTransitionGLRenderFunc render_step_opengl, gboolean bFast↩ Pace, gint iDuration, gboolean bRemoveWhenFinished, gpointer pUserData, GFreeFunc pFreeUserData↩ Func)
- void cairo_dock_remove_transition_on_icon (Icon ∗pIcon)

5.1.1 Detailed Description

This class handles the icons and containers animations. Each container has a rendering loop. An iteration of this loop is separated in 2 phases : the update of each element of the container and of the container itself, and the redraw of each element and of the container itself. The loop has 2 possible frequencies : fast (∼33Hz) and slow (∼10Hz), to optimize the CPU load according to the needs of the animation. To be called on each iteration of the loop, you register to the CAIRO_DOCK_UPDATE_X or CAIRO_DOCK_UPDATE_X_SLOW, where X is either ICON, DOCK, DESKLET, DIALOG or FLYING_CONTAINER. If you need to draw things directly on the container, you register to CAIRO_DOCK_RENDER_X, where X is either ICON, DOCK, DESKLET, DIALOG or FLYING_CONTAINER.

5.1.2 Macro Definition Documentation

5.1.2.1 #define cairo_dock_container_is_animating(*pContainer*)

Say if a container is currently animated.

Parameters

pContainer	a Container

5.1.2.2 #define cairo_dock_animation_will_be_visible(*pDock*)

Say if it's usefull to launch an animation on a Dock (indeed, it's useless to launch it if it will be invisible).

Parameters

pDock	the Dock to animate.

5.1.2.3 #define gldi_icon_stop_animation(*pIcon*)

Stop any animation on an Icon, except the disappearance/appearance animation.

Parameters

pIcon	the icon

5.1.2.4 #define cairo_dock_get_animation_delta_t(*pContainer*)

Get the interval of time between 2 iterations of the fast loop (in ms).

Parameters

pContainer	the container.

5.1.2.5 #define cairo_dock_get_slow_animation_delta_t(*pContainer*)

Get the interval of time between 2 iterations of the slow loop (in ms).

Parameters

pContainer	the container.

5.1.2.6 #define cairo_dock_has_transition(*pIcon*)

Say if an Icon has a Transition.

Parameters

pIcon	the icon.

Returns

TRUE if the icon has a Transition.

5.1.2.7 #define cairo_dock_get_transition_count(*pIcon*)

Get the the elpased number of steps since the beginning of the transition.

Parameters

pIcon	the icon.

Returns

the elpased number of steps.

5.1.2.8 #define cairo_dock_get_transition_elapsed_time(*pIcon*)

Get the elapsed time (in ms) since the beginning of the transition.

Parameters

pIcon	the icon.

Returns

the elapsed time.

5.1.2.9 #define cairo_dock_get_transition_fraction(*pIcon*)

Get the percentage of the elapsed time (between 0 and 1) since the beginning of the transition, if the transition has a fixed duration (otherwise 0).

Parameters

pIcon	the icon.

Returns

the elapsed time in [0,1].

5.1.3 Function Documentation

5.1.3.1 void cairo_dock_pop_up (CairoDock ∗ *pDock*)

Pop up a Dock above other windows, if it is in mode "keep below other windows"; otherwise do nothing.

Parameters

pDock	the dock.

5.1.3.2 void cairo_dock_pop_down (CairoDock ∗ *pDock*)

Pop down a Dock below other windows, if it is in mode "keep below other windows"; otherwise do nothing.

Parameters

pDock	the dock.

5.1.3.3 void cairo_dock_launch_animation (GldiContainer ∗ *pContainer*)

Launch the animation of a Container.

Parameters

pContainer	the container to animate.

5.1.3.4 void gldi_icon_start_animation (Icon ∗ *icon*)

Start the animation of an Icon. Do nothing if the icon is at rest or if the animation won't be visible.

Parameters

icon	the icon to animate.

5.1.3.5 void gldi_icon_request_animation (Icon ∗ *pIcon,* const gchar ∗ *cAnimation,* int *iNbRounds*)

Launch a given animation on an Icon. Do nothing if the icon will not be animated or if the animation doesn't exist.

Parameters

pIcon	the icon to animate.
cAnimation	name of the animation.
iNbRounds	number of rounds the animation will be played.

5.1.3.6 void gldi_icon_request_attention (Icon ∗ *pIcon,* const gchar ∗ *cAnimation,* int *iNbRounds*)

Launch an animation that will draw the user's attention (ie, the icon will be visible even if the dock is hidden or even if it's in a sub-dock).

Parameters

pIcon	the icon
cAnimation	an animation name, or NULL or "default" to use the default attention animation
iNbRounds	number of rounds, or <= 0 for an endles animation

5.1.3.7 void gldi_icon_stop_attention (Icon ∗ *pIcon*)

Stop the icon from drawing the attention. If the icon is not drawing the attention, do nothing.

Parameters

pIcon	the icon

5.1.3.8 void cairo_dock_trigger_icon_removal_from_dock (Icon ∗ *pIcon*)

Trigger the removal of an Icon from its Dock. The icon will effectively be removed at the end of the animation. If the icon is not inside a dock, nothing happens.

Parameters

pIcon	the icon to remove

5.1.3.9 void cairo_dock_set_transition_on_icon (Icon ∗ *pIcon*, GldiContainer ∗ *pContainer*, CairoDockTransitionRenderFunc *render_step_cairo*, CairoDockTransitionGLRenderFunc *render_step_opengl*, gboolean *bFastPace*, gint *iDuration*, gboolean *bRemoveWhenFinished*, gpointer *pUserData*, GFreeFunc *pFreeUserDataFunc*)

Set a Transition on an Icon.

Parameters

pIcon	the icon.
pContainer	the Container of the Icon. It will be shared with the transition.
render_step_↩ cairo	the cairo rendering function.
render_step_↩ opengl	the openGL rendering function (can be NULL, in which case the texture mapping from the cairo drawing is done automatically).
bFastPace	TRUE for a high frequency refresh (this uses of course more CPU).
iDuration	duration if the transition, in ms. Can be 0 for an endless transition, in which case you can stop the transition with cairo_dock_remove_transition_on_icon.
bRemove↩ WhenFinished	TRUE to destroy and remove the transition when it is finished.
pUserData	data passed to the rendering functions.
pFreeUser↩ DataFunc	function called to free the user data when the transition is destroyed (optionnal).

5.1.3.10 void cairo_dock_remove_transition_on_icon (Icon ∗ *pIcon*)

Stop and remove the Transition of an Icon.

Parameters

pIcon	the icon.

5.2 cairo-dock-applet-canvas.h File Reference

Macros

- #define CD_APPLET_DEFINE_ALL_BEGIN(_cName, _iMajorVersion, _iMinorVersion, _iMicroVersion, _i↩ AppletCategory, _cDescription, _cAuthor)
- #define CD_APPLET_DEFINE_END
- #define CD_APPLET_DEFINITION(cName, iMajorVersion, iMinorVersion, iMicroVersion, iAppletCategory, cDescription, cAuthor)

- #define CD_APPLET_INIT_ALL_BEGIN(pApplet)
- #define CD_APPLET_INIT_END
- #define CD_APPLET_STOP_BEGIN
- #define CD_APPLET_STOP_END
- #define CD_APPLET_RELOAD_ALL_BEGIN
- #define CD_APPLET_RELOAD_END
- #define CD_APPLET_GET_CONFIG_ALL_BEGIN
- #define CD_APPLET_GET_CONFIG_END
- #define CD_APPLET_RESET_CONFIG_ALL_BEGIN
- #define CD_APPLET_RESET_CONFIG_ALL_END
- #define CD_APPLET_RESET_DATA_BEGIN
- #define CD_APPLET_RESET_DATA_ALL_END
- #define CD_APPLET_ON_CLICK_BEGIN
- #define CD_APPLET_ON_CLICK_END
- #define CD_APPLET_ON_BUILD_MENU_BEGIN
- #define CD_APPLET_ON_BUILD_MENU_END
- #define CD_APPLET_ON_MIDDLE_CLICK_BEGIN
- #define CD_APPLET_ON_MIDDLE_CLICK_END
- #define CD_APPLET_ON_DOUBLE_CLICK_BEGIN
- #define CD_APPLET_ON_DOUBLE_CLICK_END
- #define CD_APPLET_ON_DROP_DATA_BEGIN
- #define CD_APPLET_ON_DROP_DATA_END
- #define CD_APPLET_ON_SCROLL_BEGIN
- #define CD_APPLET_ON_SCROLL_END
- #define CD_APPLET_ON_UPDATE_ICON_BEGIN
- #define CD_APPLET_ON_UPDATE_ICON_END
- #define CD_APPLET_SKIP_UPDATE_ICON
- #define CD_APPLET_STOP_UPDATE_ICON
- #define CD_APPLET_PAUSE_UPDATE_ICON
- #define CD_APPLET_REGISTER_FOR_CLICK_EVENT
- #define CD_APPLET_UNREGISTER_FOR_CLICK_EVENT
- #define CD_APPLET_REGISTER_FOR_BUILD_MENU_EVENT
- #define CD_APPLET_UNREGISTER_FOR_BUILD_MENU_EVENT
- #define CD_APPLET_REGISTER_FOR_MIDDLE_CLICK_EVENT
- #define CD_APPLET_UNREGISTER_FOR_MIDDLE_CLICK_EVENT
- #define CD_APPLET_REGISTER_FOR_DOUBLE_CLICK_EVENT
- #define CD_APPLET_UNREGISTER_FOR_DOUBLE_CLICK_EVENT
- #define CD_APPLET_REGISTER_FOR_DROP_DATA_EVENT
- #define CD_APPLET_UNREGISTER_FOR_DROP_DATA_EVENT
- #define CD_APPLET_REGISTER_FOR_SCROLL_EVENT
- #define CD_APPLET_UNREGISTER_FOR_SCROLL_EVENT
- #define CD_APPLET_REGISTER_FOR_UPDATE_ICON_SLOW_EVENT
- #define CD_APPLET_UNREGISTER_FOR_UPDATE_ICON_SLOW_EVENT
- #define CD_APPLET_REGISTER_FOR_UPDATE_ICON_EVENT
- #define CD_APPLET_UNREGISTER_FOR_UPDATE_ICON_EVENT

5.2.1 Detailed Description

This file defines numerous macros, that form a canvas for all the applets.

You probably won't need to dig into this file, since you can generate an applet with the 'generate-new-applet.↩
sh' script, that will build the whole canvas for you. Moreover, you can have a look at an applet that has a similar functioning to yours.

5.2.2 Macro Definition Documentation

5.2.2.1 #define CD_APPLET_DEFINE_ALL_BEGIN(*_cName, _iMajorVersion, _iMinorVersion, _iMicroVersion, _iAppletCategory, _cDescription, _cAuthor*)

Debut de la fonction de pre-initialisation de l'applet (celle qui est appele a l'enregistrement de tous les plug-ins). Definit egalement les variables globales suivantes : myIcon, myDock, myDesklet, myContainer, et myDrawContext.

Parameters

_cName	nom de sous lequel l'applet sera enregistree par Cairo-Dock.
_iMajorVersion	version majeure du dock necessaire au bon fonctionnement de l'applet.
_iMinorVersion	version mineure du dock necessaire au bon fonctionnement de l'applet.
_iMicroVersion	version micro du dock necessaire au bon fonctionnement de l'applet.
_iApplet↩ Category	Categorie de l'applet (CAIRO_DOCK_CATEGORY_ACCESSORY, CAIRO_DOCK_CATE↩ GORY_DESKTOP, CAIRO_DOCK_CATEGORY_CONTROLER)
_cDescription	description et mode d'emploi succint de l'applet.
_cAuthor	nom de l'auteur et eventuellement adresse mail.

5.2.2.2 #define CD_APPLET_DEFINE_END

Fin de la fonction de pre-initialisation de l'applet.

5.2.2.3 #define CD_APPLET_DEFINITION(*cName, iMajorVersion, iMinorVersion, iMicroVersion, iAppletCategory, cDescription, cAuthor*)

Fonction de pre-initialisation generique. Ne fais que definir l'applet (en appelant les 2 macros precedentes), la plupart du temps cela est suffisant.

5.2.2.4 #define CD_APPLET_INIT_ALL_BEGIN(*pApplet*)

Debut de la fonction d'initialisation de l'applet (celle qui est appelee a chaque chargement de l'applet). Lis le fichier de conf de l'applet, et cree son icone ainsi que son contexte de dessin.

Parameters

pApplet	une instance du module.

5.2.2.5 #define CD_APPLET_INIT_END

Fin de la fonction d'initialisation de l'applet.

5.2.2.6 #define CD_APPLET_STOP_BEGIN

Debut de la fonction d'arret de l'applet.

5.2.2.7 #define CD_APPLET_STOP_END

Fin de la fonction d'arret de l'applet.

5.2.2.8 #define CD_APPLET_RELOAD_ALL_BEGIN

Debut de la fonction de rechargement de l'applet.

5.2.2.9 #define CD_APPLET_RELOAD_END

Fin de la fonction de rechargement de l'applet.

5.2.2.10 #define CD_APPLET_GET_CONFIG_ALL_BEGIN

Debut de la fonction de configuration de l’applet (celle qui est appelee au debut de l’init).

5.2.2.11 #define CD_APPLET_GET_CONFIG_END

Fin de la fonction de configuration de l’applet.

5.2.2.12 #define CD_APPLET_RESET_CONFIG_ALL_BEGIN

Debut de la fonction de liberation des donnees de la config.

5.2.2.13 #define CD_APPLET_RESET_CONFIG_ALL_END

Fin de la fonction de liberation des donnees de la config.

5.2.2.14 #define CD_APPLET_RESET_DATA_BEGIN

Debut de la fonction de liberation des donnees internes.

5.2.2.15 #define CD_APPLET_RESET_DATA_ALL_END

Fin de la fonction de liberation des donnees internes.

5.2.2.16 #define CD_APPLET_ON_CLICK_BEGIN

Debut de la fonction de notification au clic gauche.

5.2.2.17 #define CD_APPLET_ON_CLICK_END

Fin de la fonction de notification au clic gauche. Par defaut elle intercepte la notification si elle l’a recue.

5.2.2.18 #define CD_APPLET_ON_BUILD_MENU_BEGIN

Debut de la fonction de notification de construction du menu.

5.2.2.19 #define CD_APPLET_ON_BUILD_MENU_END

Fin de la fonction de notification de construction du menu. Par defaut elle intercepte la notification si elle l’a recue.

5.2.2.20 #define CD_APPLET_ON_MIDDLE_CLICK_BEGIN

Debut de la fonction de notification du clic du milieu.

5.2.2.21 #define CD_APPLET_ON_MIDDLE_CLICK_END

Fin de la fonction de notification du clic du milieu. Par defaut elle intercepte la notification si elle l’a recue.

5.2.2.22 #define CD_APPLET_ON_DOUBLE_CLICK_BEGIN

Debut de la fonction de notification du clic du milieu.

5.2.2.23 #define CD_APPLET_ON_DOUBLE_CLICK_END

Fin de la fonction de notification du clic du milieu. Par defaut elle intercepte la notification si elle l'a recue.

5.2.2.24 #define CD_APPLET_ON_DROP_DATA_BEGIN

Debut de la fonction de notification du glisse-depose.

5.2.2.25 #define CD_APPLET_ON_DROP_DATA_END

Fin de la fonction de notification du glisse-depose. Par defaut elle intercepte la notification si elle l'a recue.

5.2.2.26 #define CD_APPLET_ON_SCROLL_BEGIN

Debut de la fonction de notification au scroll.

5.2.2.27 #define CD_APPLET_ON_SCROLL_END

Fin de la fonction de notification au scroll. Par defaut elle intercepte la notification si elle l'a recue.

5.2.2.28 #define CD_APPLET_ON_UPDATE_ICON_BEGIN

Debut de la fonction de notification d'update icon.

5.2.2.29 #define CD_APPLET_ON_UPDATE_ICON_END

Fin de la fonction de notification d'update icon.

5.2.2.30 #define CD_APPLET_SKIP_UPDATE_ICON

Quit the update function immediately and wait for the next update.

5.2.2.31 #define CD_APPLET_STOP_UPDATE_ICON

Quit the update function immediately with no more updates.

5.2.2.32 #define CD_APPLET_PAUSE_UPDATE_ICON

Quit the update function immediately with no more updates after redrawing the icon.

5.2.2.33 #define CD_APPLET_REGISTER_FOR_CLICK_EVENT

Abonne l'applet aux notifications du clic gauche. A effectuer lors de l'init de l'applet.

5.2.2.34 #define CD_APPLET_UNREGISTER_FOR_CLICK_EVENT

Desabonne l’applet aux notifications du clic gauche. A effectuer lors de l’arret de l’applet.

5.2.2.35 #define CD_APPLET_REGISTER_FOR_BUILD_MENU_EVENT

Abonne l’applet aux notifications de construction du menu. A effectuer lors de l’init de l’applet.

5.2.2.36 #define CD_APPLET_UNREGISTER_FOR_BUILD_MENU_EVENT

Desabonne l’applet aux notifications de construction du menu. A effectuer lors de l’arret de l’applet.

5.2.2.37 #define CD_APPLET_REGISTER_FOR_MIDDLE_CLICK_EVENT

Abonne l’applet aux notifications du clic du milieu. A effectuer lors de l’init de l’applet.

5.2.2.38 #define CD_APPLET_UNREGISTER_FOR_MIDDLE_CLICK_EVENT

Desabonne l’applet aux notifications du clic du milieu. A effectuer lors de l’arret de l’applet.

5.2.2.39 #define CD_APPLET_REGISTER_FOR_DOUBLE_CLICK_EVENT

Abonne l’applet aux notifications du double clic. A effectuer lors de l’init de l’applet.

5.2.2.40 #define CD_APPLET_UNREGISTER_FOR_DOUBLE_CLICK_EVENT

Desabonne l’applet aux notifications du double clic. A effectuer lors de l’arret de l’applet.

5.2.2.41 #define CD_APPLET_REGISTER_FOR_DROP_DATA_EVENT

Abonne l’applet aux notifications du glisse-depose. A effectuer lors de l’init de l’applet.

5.2.2.42 #define CD_APPLET_UNREGISTER_FOR_DROP_DATA_EVENT

Desabonne l’applet aux notifications du glisse-depose. A effectuer lors de l’arret de l’applet.

5.2.2.43 #define CD_APPLET_REGISTER_FOR_SCROLL_EVENT

Abonne l’applet aux notifications du clic gauche. A effectuer lors de l’init de l’applet.

5.2.2.44 #define CD_APPLET_UNREGISTER_FOR_SCROLL_EVENT

Desabonne l’applet aux notifications du clic gauche. A effectuer lors de l’arret de l’applet.

5.2.2.45 #define CD_APPLET_REGISTER_FOR_UPDATE_ICON_SLOW_EVENT

Register the applet to the ’update icon’ notifications of the slow rendering loop.

5.2.2.46 #define CD_APPLET_UNREGISTER_FOR_UPDATE_ICON_SLOW_EVENT

Unregister the applet from the slow rendering loop.

5.2.2.47 #define CD_APPLET_REGISTER_FOR_UPDATE_ICON_EVENT

Register the applet to the 'update icon' notifications of the fast rendering loop.

5.2.2.48 #define CD_APPLET_UNREGISTER_FOR_UPDATE_ICON_EVENT

Unregister the applet from the fast rendering loop.

5.3 cairo-dock-applet-facility.h File Reference

Macros

- #define cairo_dock_set_icon_surface(pIconContext, pSurface, pIcon)
- #define CD_CONFIG_GET_BOOLEAN_WITH_DEFAULT(cGroupName, cKeyName, bDefaultValue)
- #define CD_CONFIG_GET_BOOLEAN(cGroupName, cKeyName)
- #define CD_CONFIG_GET_INTEGER_WITH_DEFAULT(cGroupName, cKeyName, iDefaultValue)
- #define CD_CONFIG_GET_INTEGER(cGroupName, cKeyName)
- #define CD_CONFIG_GET_DOUBLE_WITH_DEFAULT(cGroupName, cKeyName, fDefaultValue)
- #define CD_CONFIG_GET_DOUBLE(cGroupName, cKeyName)
- #define CD_CONFIG_GET_INTEGER_LIST(cGroupName, cKeyName, iNbElements, iValueBuffer)
- #define CD_CONFIG_GET_STRING_WITH_DEFAULT(cGroupName, cKeyName, cDefaultValue)
- #define CD_CONFIG_GET_STRING(cGroupName, cKeyName)
- #define CD_CONFIG_GET_FILE_PATH(cGroupName, cKeyName, cDefaultFileName)
- #define CD_CONFIG_GET_STRING_LIST_WITH_DEFAULT(cGroupName, cKeyName, length, cDefault↩ Values)
- #define CD_CONFIG_GET_STRING_LIST(cGroupName, cKeyName, length)
- #define CD_CONFIG_GET_COLOR_RGBA_WITH_DEFAULT(cGroupName, cKeyName, pColorBuffer, p↩ DefaultColor)
- #define CD_CONFIG_GET_COLOR_RGBA(cGroupName, cKeyName, pColorBuffer)
- #define CD_CONFIG_GET_COLOR_RGB_WITH_DEFAULT(cGroupName, cKeyName, pColorBuffer, p↩ DefaultColor)
- #define CD_CONFIG_GET_COLOR_RGB(cGroupName, cKeyName, pColorBuffer)
- #define CD_CONFIG_GET_COLOR(cGroupName, cKeyName, pColor)
- #define CD_CONFIG_GET_THEME_PATH(cGroupName, cKeyName, cThemeDirName, cDefaultTheme↩ Name)
- #define CD_CONFIG_GET_GAUGE_THEME(cGroupName, cKeyName)
- #define CD_CONFIG_RENAME_GROUP(cGroupName, cNewGroupName)
- #define CD_APPLET_ADD_SUB_MENU_WITH_IMAGE(cLabel, pMenu, cImage)
- #define CD_APPLET_ADD_SUB_MENU(cLabel, pMenu)
- #define CD_APPLET_ADD_IN_MENU_WITH_STOCK_AND_DATA(cLabel, gtkStock, pCallBack, pMenu, pData)
- #define CD_APPLET_ADD_IN_MENU_WITH_DATA(cLabel, pCallBack, pMenu, pData)
- #define CD_APPLET_ADD_IN_MENU(cLabel, pCallBack, pMenu)
- #define CD_APPLET_ADD_IN_MENU_WITH_STOCK(cLabel, gtkStock, pCallBack, pMenu)
- #define CD_APPLET_ADD_SEPARATOR_IN_MENU(pMenu)
- #define CD_APPLET_POPUP_MENU_ON_MY_ICON(pMenu)
- #define CD_APPLET_RELOAD_CONFIG_PANEL
- #define CD_APPLET_RELOAD_CONFIG_PANEL_WITH_PAGE(iNumPage)
- #define CD_APPLET_MY_CONF_FILE

- #define CD_APPLET_MY_KEY_FILE
- #define CD_APPLET_MY_CONFIG_CHANGED
- #define CD_APPLET_MY_CONTAINER_TYPE_CHANGED
- #define CD_APPLET_MY_OLD_CONTAINER
- #define CD_APPLET_CLICKED_ICON
- #define CD_APPLET_CLICKED_CONTAINER
- #define CD_APPLET_SHIFT_CLICK
- #define CD_APPLET_CTRL_CLICK
- #define CD_APPLET_ALT_CLICK
- #define CD_APPLET_MY_MENU
- #define CD_APPLET_RECEIVED_DATA
- #define CD_APPLET_SCROLL_UP
- #define CD_APPLET_SCROLL_DOWN
- #define CD_APPLET_BIND_KEY(cShortKey, cDescription, cGroupName, cKeyName, handler)
- #define CD_APPLET_REDRAW_MY_ICON
- #define CAIRO_DOCK_REDRAW_MY_CONTAINER
- #define CD_APPLET_LOAD_SURFACE_FOR_MY_APPLET(cImagePath)
- #define CD_APPLET_LOAD_SURFACE_FOR_MY_APPLET_WITH_DEFAULT(cUserImageName, c↩ DefaultLocalImageName)
- #define CD_APPLET_SET_SURFACE_ON_MY_ICON(pSurface)
- #define CD_APPLET_SET_IMAGE_ON_MY_ICON(cIconName)
- #define CD_APPLET_SET_USER_IMAGE_ON_MY_ICON(cIconName, cDefaultLocalImageName)
- #define CD_APPLET_SET_DEFAULT_IMAGE_ON_MY_ICON_IF_NONE
- #define CD_APPLET_SET_NAME_FOR_MY_ICON(cIconName)
- #define CD_APPLET_SET_NAME_FOR_MY_ICON_PRINTF(cIconNameFormat,...)
- #define CD_APPLET_SET_QUICK_INFO_ON_MY_ICON(cQuickInfo)
- #define CD_APPLET_SET_QUICK_INFO_ON_MY_ICON_PRINTF(cQuickInfoFormat,...)
- #define CD_APPLET_SET_HOURS_MINUTES_AS_QUICK_INFO(iTimeInSeconds)
- #define CD_APPLET_SET_MINUTES_SECONDES_AS_QUICK_INFO(iTimeInSeconds)
- #define CD_APPLET_SET_SIZE_AS_QUICK_INFO(iSizeInBytes)
- #define CD_APPLET_SET_STATIC_ICON
- #define CD_APPLET_UNSET_STATIC_ICON
- #define CD_APPLET_SET_ALWAYS_VISIBLE_ICON(bAlwaysVisible)
- #define CD_APPLET_ANIMATE_MY_ICON(cAnimationName, iAnimationLength)
- #define CD_APPLET_STOP_ANIMATING_MY_ICON
- #define CD_APPLET_DEMANDS_ATTENTION(cAnimationName, iAnimationLength)
- #define CD_APPLET_STOP_DEMANDING_ATTENTION
- #define CD_APPLET_GET_MY_ICON_EXTENT(iWidthPtr, iHeightPtr)
- #define CD_APPLET_START_DRAWING_MY_ICON
- #define CD_APPLET_START_DRAWING_MY_ICON_CAIRO
- #define CD_APPLET_START_DRAWING_MY_ICON_OR_RETURN(...)
- #define CD_APPLET_START_DRAWING_MY_ICON_OR_RETURN_CAIRO(...)
- #define CD_APPLET_FINISH_DRAWING_MY_ICON
- #define CD_APPLET_FINISH_DRAWING_MY_ICON_CAIRO
- #define CD_APPLET_ADD_OVERLAY_ON_MY_ICON(cImageFile, iPosition)
- #define CD_APPLET_PRINT_OVERLAY_ON_MY_ICON(cImageFile, iPosition)
- #define CD_APPLET_REMOVE_OVERLAY_ON_MY_ICON(iPosition)
- #define CD_APPLET_ADD_DATA_RENDERER_ON_MY_ICON(pAttr)
- #define CD_APPLET_RELOAD_MY_DATA_RENDERER(...)
- #define CD_APPLET_RENDER_NEW_DATA_ON_MY_ICON(pValues)
- #define CD_APPLET_REMOVE_MY_DATA_RENDERER
- #define CD_APPLET_SET_MY_DATA_RENDERER_HISTORY_TO_MAX
- #define CD_APPLET_MY_CONTAINER_IS_OPENGL
- #define CD_APPLET_SET_DESKLET_RENDERER_WITH_DATA(cRendererName, pConfig)
- #define CD_APPLET_SET_DESKLET_RENDERER(cRendererName)

- #define CD_APPLET_SET_STATIC_DESKLET
- #define CD_APPLET_ALLOW_NO_CLICKABLE_DESKLET
- #define CD_APPLET_DELETE_MY_ICONS_LIST
- #define CD_APPLET_REMOVE_ICON_FROM_MY_ICONS_LIST(pIcon)
- #define CD_APPLET_DETACH_ICON_FROM_MY_ICONS_LIST(pIcon)
- #define CD_APPLET_LOAD_MY_ICONS_LIST(pIconList, cDockRendererName, cDeskletRendererName, pDeskletRendererConfig)
- #define CD_APPLET_ADD_ICON_IN_MY_ICONS_LIST(pIcon)
- #define CD_APPLET_MY_ICONS_LIST
- #define CD_APPLET_MY_ICONS_LIST_CONTAINER
- #define CD_APPLET_MANAGE_APPLICATION(cApplicationClass)
- #define D_(message)

Enumerations

- enum CairoDockInfoDisplay {
  CAIRO_DOCK_INFO_NONE,
  CAIRO_DOCK_INFO_ON_ICON,
  CAIRO_DOCK_INFO_ON_LABEL }

  type of possible display on a Icon.

Functions

- void cairo_dock_set_icon_surface_full (cairo_t ∗pIconContext, cairo_surface_t ∗pSurface, double fScale, double fAlpha, Icon ∗pIcon)
- gboolean cairo_dock_set_image_on_icon (cairo_t ∗pIconContext, const gchar ∗cIconName, Icon ∗pIcon, GldiContainer ∗pContainer)
- void cairo_dock_set_image_on_icon_with_default (cairo_t ∗pIconContext, const gchar ∗cImage, Icon ∗pIcon, GldiContainer ∗pContainer, const gchar ∗cDefaultImagePath)
- gchar ∗ cairo_dock_get_human_readable_size (long long int iSizeInBytes)
- void cairo_dock_play_sound (const gchar ∗cSoundPath)

5.3.1 Detailed Description

A collection of useful macros for applets. Macros provides a normalized API that will :

- lets you perform complex operations with a minimum amount of code
- ensures a bug-free functioning
- masks the internal complexity
- allows a normalized and easy-to-maintain code amongst all the applets.

5.3.2 Macro Definition Documentation

5.3.2.1 #define cairo_dock_set_icon_surface(*pIconContext, pSurface, pIcon*)

Apply a surface on a context. The context is cleared beforehand with the default icon background..

Parameters

pIconContext	the drawing context; is not altered by the function.
pSurface	the surface to apply.
pIcon	the icon.

5.3.2.2 #define CD_CONFIG_GET_BOOLEAN_WITH_DEFAULT(*cGroupName, cKeyName, bDefaultValue*)

Get the value of a 'boolean' from the conf file.

Parameters

cGroupName	name of the group in the conf file.
cKeyName	name of the key in the conf file.
bDefaultValue	default value if the group/key is not found (typically if the key is new).

Returns

a gboolean.

5.3.2.3 #define CD_CONFIG_GET_BOOLEAN(*cGroupName, cKeyName*)

Get the value of a 'boolean' from the conf file, with TRUE as default value.

Parameters

cGroupName	name of the group in the conf file.
cKeyName	name of the key in the conf file.

Returns

a gboolean.

5.3.2.4 #define CD_CONFIG_GET_INTEGER_WITH_DEFAULT(*cGroupName, cKeyName, iDefaultValue*)

Get the value of an 'integer' from the conf file.

Parameters

cGroupName	name of the group in the conf file.
cKeyName	name of the key in the conf file.
iDefaultValue	default value if the group/key is not found (typically if the key is new).

Returns

an integer.

5.3.2.5 #define CD_CONFIG_GET_INTEGER(*cGroupName, cKeyName*)

Get the value of a 'entier' from the conf file, with 0 as default value.

Parameters

cGroupName	name of the group in the conf file.
cKeyName	name of the key in the conf file.

Returns

an integer.

5.3.2.6 #define CD_CONFIG_GET_DOUBLE_WITH_DEFAULT(*cGroupName*, *cKeyName*, *fDefaultValue*)

Get the value of a 'double' from the conf file.

Parameters

cGroupName	name of the group in the conf file.
cKeyName	name of the key in the conf file.
fDefaultValue	default value if the group/key is not found (typically if the key is new).

Returns

a double.

5.3.2.7 #define CD_CONFIG_GET_DOUBLE(*cGroupName*, *cKeyName*)

Get the value of a 'double' from the conf file, with 0. as default value.

Parameters

cGroupName	name of the group in the conf file.
cKeyName	name of the key in the conf file.

Returns

a double.

5.3.2.8 #define CD_CONFIG_GET_INTEGER_LIST(*cGroupName*, *cKeyName*, *iNbElements*, *iValueBuffer*)

Get the value of an 'integers list' from the conf file.

Parameters

cGroupName	name of the group in the conf file.
cKeyName	name of the key in the conf file.
iNbElements	number of elements to get from the conf file.
iValueBuffer	buffer to fill with the values.

5.3.2.9 #define CD_CONFIG_GET_STRING_WITH_DEFAULT(*cGroupName*, *cKeyName*, *cDefaultValue*)

Get the value of a 'string' from the conf file.

Parameters

cGroupName	name of the group in the conf file.
cKeyName	name of the key in the conf file.
cDefaultValue	default value if the group/key is not found (typically if the key is new). can be NULL.

Returns

a newly allocated string.

5.3.2.10 #define CD_CONFIG_GET_STRING(*cGroupName, cKeyName*)

Get the value of a 'string' from the conf file, with NULL as default value.

Parameters

cGroupName	name of the group in the conf file.
cKeyName	name of the key in the conf file.

Returns

a newly allocated string.

5.3.2.11 #define CD_CONFIG_GET_FILE_PATH(*cGroupName, cKeyName, cDefaultFileName*)

Get the value of a 'file' from the conf file, with NULL as default value. If the value is a file name (not a path), it is supposed to be in the Cairo-Dock's current theme folder. If the value is NULL, the default file is used, taken at the applet's data folder, but the conf file is not updated with this value.

Parameters

cGroupName	name of the group in the conf file.
cKeyName	name of the key in the conf file.
cDefaultFile↩ Name	defaul tfile if none is specified in the conf file.

Returns

a newly allocated string giving the complete path of the file.

5.3.2.12 #define CD_CONFIG_GET_STRING_LIST_WITH_DEFAULT(*cGroupName, cKeyName, length, cDefaultValues*)

Get the value of a 'strings list' from the conf file.

Parameters

cGroupName	name of the group in the conf file.
cKeyName	name of the key in the conf file.
length	pointer to the number of strings that were extracted from the conf file.
cDefaultValues	default value if the group/key is not found (typically if the key is new). It is a string with words separated by ';'. It can be NULL.

Returns

a table of strings, to be freeed with 'g_strfreev'.

5.3.2.13 #define CD_CONFIG_GET_STRING_LIST(*cGroupName, cKeyName, length*)

Get the value of a 'strings list' from the conf file, with NULL as default value.

Parameters

cGroupName	name of the group in the conf file.
cKeyName	name of the key in the conf file.
length	pointer to the number of strings that were extracted from the conf file.

Returns

a table of strings, to be freeed with 'g_strfreev'.

5.3.2.14 #define CD_CONFIG_GET_COLOR_RGBA_WITH_DEFAULT(*cGroupName,* *cKeyName,* *pColorBuffer,* *pDefaultColor*)

Get the value of a 'color' in the RGBA format from the conf file.

Parameters

cGroupName	name of the group in the conf file.
cKeyName	name of the key in the conf file.
pColorBuffer	a table of 4 'double' already allocated, that will be filled with the color components.
pDefaultColor	default value if the group/key is not found (typically if the key is new). It is a table of 4 'double'. It can be NULL.

5.3.2.15 #define CD_CONFIG_GET_COLOR_RGBA(*cGroupName,* *cKeyName,* *pColorBuffer*)

Get the value of a 'color' in the RGBA format from the conf file, with NULL as default value.

Parameters

cGroupName	name of the group in the conf file.
cKeyName	name of the key in the conf file.
pColorBuffer	a table of 4 'double' already allocated, that will be filled with the color components.

5.3.2.16 #define CD_CONFIG_GET_COLOR_RGB_WITH_DEFAULT(*cGroupName,* *cKeyName,* *pColorBuffer,* *pDefaultColor*)

Get the value of a 'color' in the RGB format from the conf file.

Parameters

cGroupName	name of the group in the conf file.
cKeyName	name of the key in the conf file.
pColorBuffer	a table of 3 'double' already allocated, that will be filled with the color components.
pDefaultColor	default value if the group/key is not found (typically if the key is new). It is a table of 3 'double'. It can be NULL.

5.3.2.17 #define CD_CONFIG_GET_COLOR_RGB(*cGroupName,* *cKeyName,* *pColorBuffer*)

Get the value of a 'color' in the RGB format from the conf file, with NULL as default value.

Parameters

cGroupName	name of the group in the conf file.

cKeyName	name of the key in the conf file.
pColorBuffer	a table of 3 'double' already allocated, that will be filled with the color components.

5.3.2.18 #define CD_CONFIG_GET_COLOR(*cGroupName, cKeyName, pColor*)

Get the value of a 'color' in a GldiColor from the conf file, with NULL as default value.

Parameters

cGroupName	name of the group in the conf file.
cKeyName	name of the key in the conf file.
pColor	a GldiColor already allocated, that will be filled with the color components.

5.3.2.19 #define CD_CONFIG_GET_THEME_PATH(*cGroupName, cKeyName, cThemeDirName, cDefaultThemeName*)

Get the complete path of a theme in the conf file.

Parameters

cGroupName	name of the group (in the conf file).
cKeyName	name of the key (in the conf file).
cThemeDirName	name of the folder containing the local, user, and distant themes.
cDefault↩ ThemeName	default value, if the key/group/theme doesn't exist.

Returns

Path to the folder of the theme, in a newly allocated string.

5.3.2.20 #define CD_CONFIG_GET_GAUGE_THEME(*cGroupName, cKeyName*)

Get the complete path of a Gauge theme in the conf file.

Parameters

cGroupName	name of the group (in the conf file).
cKeyName	name of the key (in the conf file).

Returns

Path to the theme, in a newly allocated string.

5.3.2.21 #define CD_CONFIG_RENAME_GROUP(*cGroupName, cNewGroupName*)

Rename a group in the conf file, in case you had to change it. Do nothing if the old group no more exists in the conf file.

Parameters

cGroupName	name of the group.
cNewGroup↩ Name	new name of the group.

5.3.2.22 #define CD_APPLET_ADD_SUB_MENU_WITH_IMAGE(*cLabel, pMenu, cImage*)

Create and add a sub-menu to a given menu.

Parameters

cLabel	name of the sub-menu.
pMenu	GtkWidget of the menu we will add the sub-menu to..
cImage	name of an image (can be a path or a GtkStock).

Returns

the sub-menu, newly created and attached to the menu.

5.3.2.23 #define CD_APPLET_ADD_SUB_MENU(*cLabel, pMenu*)

Create and add a sub-menu to a given menu.

Parameters

cLabel	name of the sub-menu.
pMenu	GtkWidget of the menu we will add the sub-menu to..

Returns

the sub-menu, newly created and attached to the menu.

5.3.2.24 #define CD_APPLET_ADD_IN_MENU_WITH_STOCK_AND_DATA(*cLabel, gtkStock, pCallBack, pMenu, pData*)

Create and add an entry to a menu, with an icon.

Parameters

cLabel	name of the entry.
gtkStock	name of a GTK icon or path to an image.
pCallBack	function called when the user selects this entry.
pMenu	menu to add the entry to.
pData	data passed as parameter of the callback.

5.3.2.25 #define CD_APPLET_ADD_IN_MENU_WITH_DATA(*cLabel, pCallBack, pMenu, pData*)

Create and add an entry to a menu.

Parameters

cLabel	name of the entry.
pCallBack	function called when the user selects this entry.
pMenu	menu to add the entry to.
pData	data passed as parameter of the callback.

5.3.2.26 #define CD_APPLET_ADD_IN_MENU(*cLabel, pCallBack, pMenu*)

Create and add an entry to a menu. 'myApplet' will be passed to the callback.

Parameters

cLabel	name of the entry.
pCallBack	function called when the user selects this entry.
pMenu	menu to add the entry to.

5.3.2.27 #define CD_APPLET_ADD_IN_MENU_WITH_STOCK(*cLabel*, *gtkStock*, *pCallBack*, *pMenu*)

Create and add an entry to a menu, with an icon. 'myApplet' will be passed to the callback.

Parameters

cLabel	name of the entry.
gtkStock	name of a GTK icon or path to an image.
pCallBack	function called when the user selects this entry.
pMenu	menu to add the entry to.

5.3.2.28 #define CD_APPLET_ADD_SEPARATOR_IN_MENU(*pMenu*)

Create and add a separator to a menu.

5.3.2.29 #define CD_APPLET_POPUP_MENU_ON_MY_ICON(*pMenu*)

Pop-up a menu on the applet's icon.

Parameters

pMenu	menu to show

5.3.2.30 #define CD_APPLET_RELOAD_CONFIG_PANEL

Reload the config panel of the applet. This is useful if you have custom widgets inside your conf file, and need to reload them.

5.3.2.31 #define CD_APPLET_RELOAD_CONFIG_PANEL_WITH_PAGE(*iNumPage*)

Reload the config panel of the applet and jump to the given page. This is useful if you have custom widgets inside your conf file, and need to reload them.

5.3.2.32 #define CD_APPLET_MY_CONF_FILE

Path of the applet's instance's conf file.

5.3.2.33 #define CD_APPLET_MY_KEY_FILE

Key file of the applet instance, availale during the init, config, and reload.

5.3.2.34 #define CD_APPLET_MY_CONFIG_CHANGED

TRUE if the conf file has changed before the reload.

5.3.2.35 #define CD_APPLET_MY_CONTAINER_TYPE_CHANGED

TRUE if the container type has changed (which can only happen if the config has changed).

5.3.2.36 #define CD_APPLET_MY_OLD_CONTAINER

The previous Container.

5.3.2.37 #define CD_APPLET_CLICKED_ICON

The clicked Icon.

5.3.2.38 #define CD_APPLET_CLICKED_CONTAINER

The clicked Container.

5.3.2.39 #define CD_APPLET_SHIFT_CLICK

TRUE if the 'SHIFT' key was pressed during the click.

5.3.2.40 #define CD_APPLET_CTRL_CLICK

TRUE if the 'CTRL' key was pressed during the click.

5.3.2.41 #define CD_APPLET_ALT_CLICK

TRUE if the 'ALT' key was pressed during the click.

5.3.2.42 #define CD_APPLET_MY_MENU

Main menu of the applet.

5.3.2.43 #define CD_APPLET_RECEIVED_DATA

Data received after a drop occured (string).

5.3.2.44 #define CD_APPLET_SCROLL_UP

TRUE if the user scrolled up.

5.3.2.45 #define CD_APPLET_SCROLL_DOWN

TRUE if the user scrolled down.

5.3.2.46 #define CD_APPLET_BIND_KEY(*cShortKey*, *cDescription*, *cGroupName*, *cKeyName*, *handler*)

Bind a shortkey to an action. Unref it when you don't want it anymore. 'myApplet' is passed as the callback data.

Parameters

cShortKey	a keyboard shortcut.
cDescription	a short description of the action
cGroupName	group name where it's stored in the applet's conf file
cKeyName	key name where it's stored in the applet's conf file
handler	function called when the shortkey is pressed by the user

Returns

the shortkey.

5.3.2.47 #define CD_APPLET_REDRAW_MY_ICON

Redraw the applet's icon (as soon as the main loop is available).

5.3.2.48 #define CAIRO_DOCK_REDRAW_MY_CONTAINER

Redraw the applet's container (as soon as the main loop is available).

5.3.2.49 #define CD_APPLET_LOAD_SURFACE_FOR_MY_APPLET(*cImagePath*)

Load an image into a surface, at the same size as the applet's icon. If the image is given by its sole name, it is searched inside the current theme root folder.

Parameters

cImagePath	path or name of an image.

Returns

the newly allocated surface.

5.3.2.50 #define CD_APPLET_LOAD_SURFACE_FOR_MY_APPLET_WITH_DEFAULT(*cUserImageName, cDefaultLocalImageName*)

Load a user image into a surface, at the same size as the applet's icon, or a default image taken in the installed folder of the applet if the first one is NULL. If the user image is given by its sole name, it is searched inside the current theme root folder.

Parameters

cUserImage↩ Name	name or path of an user image.
cDefaultLocal↩ ImageName	default image

Returns

the newly allocated surface.

5.3.2.51 #define CD_APPLET_SET_SURFACE_ON_MY_ICON(*pSurface*)

Apply a surface on the applet's icon, and redraw it.

Parameters

pSurface	the surface to draw on your icon.

5.3.2.52 #define CD_APPLET_SET_IMAGE_ON_MY_ICON(*cIconName*)

Apply an image on the applet's icon. The image is resized at the same size as the icon. Does not trigger the icon refresh.

Parameters

cIconName	name of an icon or path to an image.

5.3.2.53 #define CD_APPLET_SET_USER_IMAGE_ON_MY_ICON(*cIconName, cDefaultLocalImageName*)

Apply an image on the applet's icon, clearing it beforehand, and adding the reflect. The image is searched in any possible locations, and the default image provided is used if the search was fruitless (taken in the installation folder of the applet).

Parameters

cIconName	name of an icon or path to an image.
cDefaultLocal↩ ImageName	name of an image to use as a fallback (taken in the applet's installation folder).

5.3.2.54 #define CD_APPLET_SET_DEFAULT_IMAGE_ON_MY_ICON_IF_NONE

Apply the default icon on the applet's icon if there is no image yet.

5.3.2.55 #define CD_APPLET_SET_NAME_FOR_MY_ICON(*cIconName*)

Set a new label on the applet's icon.

Parameters

cIconName	the label.

5.3.2.56 #define CD_APPLET_SET_NAME_FOR_MY_ICON_PRINTF(*cIconNameFormat, ...*)

Set a new label on the applet's icon.

Parameters

cIconName↩ Format	the label, in a 'printf'-like format.
...	values to be written in the string.

5.3.2.57 #define CD_APPLET_SET_QUICK_INFO_ON_MY_ICON(*cQuickInfo*)

Set a quick-info on the applet's icon.

Parameters

cQuickInfo	the quick-info. This is a small text (a few characters) that is superimposed on the icon.

5.3.2.58 #define CD_APPLET_SET_QUICK_INFO_ON_MY_ICON_PRINTF(*cQuickInfoFormat, ...*)

Set a quick-info on the applet's icon.

Parameters

cQuickInfo↵ Format	the label, in a 'printf'-like format.
...	values to be written in the string.

5.3.2.59 #define CD_APPLET_SET_HOURS_MINUTES_AS_QUICK_INFO(*iTimeInSeconds*)

Write the time in hours-minutes as a quick-info on the applet's icon.

Parameters

iTimeInSeconds	the time in seconds.

5.3.2.60 #define CD_APPLET_SET_MINUTES_SECONDES_AS_QUICK_INFO(*iTimeInSeconds*)

Write the time in minutes-secondes as a quick-info on the applet's icon.

Parameters

iTimeInSeconds	the time in seconds.

5.3.2.61 #define CD_APPLET_SET_SIZE_AS_QUICK_INFO(*iSizeInBytes*)

Write a size in bytes as a quick-info on the applet's icon.

Parameters

iSizeInBytes	the size in bytes, converted into a readable format.

5.3.2.62 #define CD_APPLET_SET_STATIC_ICON

Prevent the applet's icon to be animated when the mouse hovers it (call it once at init).

5.3.2.63 #define CD_APPLET_UNSET_STATIC_ICON

Prevent the applet's icon to be animated when the mouse hovers it (call it once at init).

5.3.2.64 #define CD_APPLET_SET_ALWAYS_VISIBLE_ICON(*bAlwaysVisible*)

Make the applet's icon always visible, even when the dock is hidden.

5.3.2.65 #define CD_APPLET_ANIMATE_MY_ICON(*cAnimationName, iAnimationLength*)

Launch an animation on the applet's icon.

Parameters

cAnimation↵ Name	name of the animation.
iAnimation↵ Length	number of rounds the animation should be played.

5.3.2.66 #define CD_APPLET_STOP_ANIMATING_MY_ICON

Stop any animation on the applet's icon.

5.3.2.67 #define CD_APPLET_DEMANDS_ATTENTION(*cAnimationName, iAnimationLength*)

Make applet's icon demanding the attention : it will launch the given animation, and the icon will be visible even if the dock is hidden.

Parameters

cAnimation↵ Name	name of the animation.
iAnimation↵ Length	number of rounds the animation should be played, or 0 for an endless animation.

5.3.2.68 #define CD_APPLET_STOP_DEMANDING_ATTENTION

Stop the demand of attention on the applet's icon.

5.3.2.69 #define CD_APPLET_GET_MY_ICON_EXTENT(*iWidthPtr, iHeightPtr*)

Get the dimension allocated to the surface/texture of the applet's icon.

Parameters

iWidthPtr	pointer to the width.
iHeightPtr	pointer to the height.

5.3.2.70 #define CD_APPLET_START_DRAWING_MY_ICON

Initiate an OpenGL drawing session on the applet's icon.

5.3.2.71 #define CD_APPLET_START_DRAWING_MY_ICON_CAIRO

Initiate a Cairo drawing session on the applet's icon.

5.3.2.72 #define CD_APPLET_START_DRAWING_MY_ICON_OR_RETURN(*...*)

Initiate an OpenGL drawing session on the applet's icon, or quit the function if failed.

Parameters

...	value to return in case of failure.

5.3.2.73 #define CD_APPLET_START_DRAWING_MY_ICON_OR_RETURN_CAIRO(*...*)

Initiate a Cairo drawing session on the applet's icon, or quit the function if failed.

Parameters

...	value to return in case of failure.

5.3.2.74 #define CD_APPLET_FINISH_DRAWING_MY_ICON

Terminate an OpenGL drawing session on the applet's icon. Does not trigger the icon's redraw.

5.3.2.75 #define CD_APPLET_FINISH_DRAWING_MY_ICON_CAIRO

Terminate an OpenGL drawing session on the applet's icon. Does not trigger the icon's redraw.

5.3.2.76 #define CD_APPLET_ADD_OVERLAY_ON_MY_ICON(*cImageFile, iPosition*)

Add an overlay from an image on the applet's icon.

Parameters

cImageFile	an image (if it's not a path, it is searched amongst the current theme's images)
iPosition	position where to display the overlay

Returns

the overlay, or NULL if the image couldn't be loaded.

5.3.2.77 #define CD_APPLET_PRINT_OVERLAY_ON_MY_ICON(*cImageFile, iPosition*)

Print an overlay from an image on the applet's icon (it can't be removed without erasing the icon).

Parameters

cImageFile	an image (if it's not a path, it is searched amongst the current theme's images)
iPosition	position where to display the overlay

Returns

TRUE if the overlay has been successfuly printed.

5.3.2.78 #define CD_APPLET_REMOVE_OVERLAY_ON_MY_ICON(*iPosition*)

Remove an overlay from the applet's icon. The overlay is destroyed.

Parameters

iPosition	position of the overlay

5.3.2.79 #define CD_APPLET_ADD_DATA_RENDERER_ON_MY_ICON(*pAttr*)

Add a Data Renderer the applet's icon.

Parameters

pAttr	the attributes of the Data Renderer. They allow you to define its properties.

5.3.2.80 #define CD_APPLET_RELOAD_MY_DATA_RENDERER(*...*)

Reload the Data Renderer of the applet's icon, without changing any of its parameters. Previous values are kept.

5.3.2.81 #define CD_APPLET_RENDER_NEW_DATA_ON_MY_ICON(*pValues*)

Add new values to the Data Renderer of the applet's icon. Values are a table of 'double', having the same size as defined when the data renderer was created (1 by default). It also triggers the redraw of the icon.

Parameters

pValues	the values, a table of double of the correct size.

5.3.2.82 #define CD_APPLET_REMOVE_MY_DATA_RENDERER

Completely remove the Data Renderer of the applet's icon, including the values associated with.

5.3.2.83 #define CD_APPLET_SET_MY_DATA_RENDERER_HISTORY_TO_MAX

Set the history size of the Data Renderer of the applet's icon to the maximum size, that is to say 1 value per pixel.

5.3.2.84 #define CD_APPLET_MY_CONTAINER_IS_OPENGL

Say if the applet's container currently supports OpenGL.

5.3.2.85 #define CD_APPLET_SET_DESKLET_RENDERER_WITH_DATA(*cRendererName, pConfig*)

Set a renderer to the applet's desklet and create myDrawContext. Call it at the beginning of init and also reload, to take into account the desklet's resizing.

Parameters

cRendererName	name of the renderer.
pConfig	configuration data for the renderer, or NULL.

5.3.2.86 #define CD_APPLET_SET_DESKLET_RENDERER(*cRendererName*)

Set a renderer to the applet's desklet and create myDrawContext. Call it at the beginning of init and also reload, to take into account the desklet's resizing.

Parameters

cRendererName	name of the renderer.

5.3.2.87 #define CD_APPLET_SET_STATIC_DESKLET

Prevent the desklet from being rotated. Use it if your desklet has some static GtkWidget inside.

5.3.2.88 #define CD_APPLET_ALLOW_NO_CLICKABLE_DESKLET

Prevent the desklet from being transparent to click. Use it if your desklet has no meaning in being unclickable.

5.3.2.89 #define CD_APPLET_DELETE_MY_ICONS_LIST

Delete the list of icons of an applet (keep the subdock in dock mode).

5.3.2.90 #define CD_APPLET_REMOVE_ICON_FROM_MY_ICONS_LIST(*pIcon*)

Remove an icon from the list of icons of an applet. The icon is destroyed and should not be used after that.

Parameters

pIcon	the icon to remove.

Returns

whether the icon has been removed or not. In any case, the icon is freed.

5.3.2.91 #define CD_APPLET_DETACH_ICON_FROM_MY_ICONS_LIST(*pIcon*)

Detach an icon from the list of icons of an applet. The icon is not destroyed.

Parameters

pIcon	the icon to remove.

Returns

whether the icon has been removed or not.

5.3.2.92 #define CD_APPLET_LOAD_MY_ICONS_LIST(*pIconList,* *cDockRendererName,* *cDeskletRendererName,* *pDeskletRendererConfig*)

Load a list of icons into an applet, with the given renderer for the sub-dock or the desklet. The icons will be loaded automatically in an idle process.

Parameters

pIconList	a list of icons. It will belong to the applet's container after that.
cDock↩ RendererName	name of a renderer in case the applet is in dock mode.

cDesklet←RendererName	name of a renderer in case the applet is in desklet mode.
pDesklet←RendererConfig	possible configuration parameters for the desklet renderer.

5.3.2.93 #define CD_APPLET_ADD_ICON_IN_MY_ICONS_LIST(*pIcon*)

Add an icon into an applet. The view previously set by CD_APPLET_LOAD_MY_ICONS_LIST will be used. The icon will be loaded automatically in an idle process.

Parameters

pIcon	an icon.

5.3.2.94 #define CD_APPLET_MY_ICONS_LIST

Get the list of icons of your applet. It is either the icons of your sub-dock or of your desklet.

5.3.2.95 #define CD_APPLET_MY_ICONS_LIST_CONTAINER

Get the container of the icons of your applet. It is either your sub-dock or your desklet.

5.3.2.96 #define CD_APPLET_MANAGE_APPLICATION(*cApplicationClass*)

Let your applet control the window of an external program, instead of the Taskbar.

Parameters

cApplication←Class	the class of the application you wish to control (in lower case), or NULL to stop controling any appli.

5.3.2.97 #define D_(*message*)

Macro for gettext, similar to _() et N_(), but with the domain of the applet. Surround all your strings with this, so that 'xgettext' can find them and automatically include them in the translation files.

5.3.3 Enumeration Type Documentation

5.3.3.1 enum CairoDockInfoDisplay

type of possible display on a Icon.

Enumerator

CAIRO_DOCK_INFO_NONE don't display anything.

CAIRO_DOCK_INFO_ON_ICON display info on the icon (as quick-info).

CAIRO_DOCK_INFO_ON_LABEL display on the label of the icon.

5.3.4 Function Documentation

5.3.4.1 void cairo_dock_set_icon_surface_full (cairo_t ∗ *pIconContext,* cairo_surface_t ∗ *pSurface,* double *fScale,* double *fAlpha,* Icon ∗ *pIcon*)

Apply a surface on a context, with a zoom and a transparency factor. The context is cleared beforehand with the default icon background.

Parameters

pIconContext	the drawing context; is not altered by the function.
pSurface	the surface to apply.
fScale	zoom factor.
fAlpha	transparency in [0,1].
pIcon	the icon.

5.3.4.2 gboolean cairo_dock_set_image_on_icon (cairo_t ∗ *pIconContext,* const gchar ∗ *cIconName,* Icon ∗ *pIcon,* GldiContainer ∗ *pContainer*)

Apply an image on the context of an icon, clearing it beforehand, and adding the reflect.

Parameters

pIconContext	the drawing context; is not altered by the function.
cIconName	name or path to an icon image.
pIcon	the icon.
pContainer	the container of the icon.

Returns

TRUE if everything went smoothly.

5.3.4.3 void cairo_dock_set_image_on_icon_with_default (cairo_t ∗ *pIconContext,* const gchar ∗ *cImage,* Icon ∗ *pIcon,* GldiContainer ∗ *pContainer,* const gchar ∗ *cDefaultImagePath*)

Apply an image on the context of an icon, clearing it beforehand, and adding the reflect. The image is searched in any possible locations, and the default image provided is used if the search was fruitless.

Parameters

pIconContext	the drawing context; is not altered by the function.
cImage	name of an image to apply on the icon.
pIcon	the icon.
pContainer	the container of the icon.
cDefaultImage↩ Path	path to a default image.

5.3.4.4 gchar∗ cairo_dock_get_human_readable_size (long long int *iSizeInBytes*)

Convert a size in bytes into a readable format.

Parameters

iSizeInBytes	size in bytes.

Returns

a newly allocated string.

5.3.4.5 void cairo_dock_play_sound (const gchar ∗ *cSoundPath*)

Play a sound, through Alsa or PulseAudio.

Parameters

cSoundPath	path to an audio file.

5.4 cairo-dock-applet-manager.h File Reference

Macros

- #define GLDI_OBJECT_IS_APPLET_ICON(obj)

5.4.1 Detailed Description

This class handles the Applet Icons, which are icons used by module instances. Note: they are not UserIcon, because they are created by and belongs to a ModuleInstance, which is the actual object belonging to the user.

5.4.2 Macro Definition Documentation

5.4.2.1 #define GLDI_OBJECT_IS_APPLET_ICON(*obj*)

Say if an object is a AppletIcon.

Parameters

obj	the object.

Returns

TRUE if the object is a AppletIcon.

5.5 cairo-dock-applications-manager.h File Reference

Macros

- #define GLDI_OBJECT_IS_APPLI_ICON(obj)

Functions

- void cairo_dock_start_applications_manager (CairoDock ∗pDock)
- GList ∗ cairo_dock_get_current_applis_list (void)
- Icon ∗ cairo_dock_get_current_active_icon (void)
- Icon ∗ cairo_dock_get_appli_icon (GldiWindowActor ∗actor)
- void cairo_dock_foreach_appli_icon (GldiIconFunc pFunction, gpointer pUserData)

5.5.1 Detailed Description

This class manages the list of icons representing a window, ie the Taskbar.

5.5.2 Macro Definition Documentation

5.5.2.1 #define GLDI_OBJECT_IS_APPLI_ICON(*obj*)

Say if an object is an AppliIcon.

Parameters

obj	the object.

Returns

TRUE if the object is a AppliIcon.

5.5.3 Function Documentation

5.5.3.1 void cairo_dock_start_applications_manager (CairoDock ∗ *pDock*)

Start the applications manager. It will load all the appli-icons, and keep monitoring them. If enabled, it will insert them into the dock.

Parameters

pDock	the main dock

5.5.3.2 GList∗ cairo_dock_get_current_applis_list (void)

Get the list of appli-icons, including the icons not currently displayed in the dock. You can then order the list by z-order, name, etc.

Returns

a newly allocated list of appli-icons. You must free the list when you're done with it, but not the icons.

5.5.3.3 Icon∗ cairo_dock_get_current_active_icon (void)

Get the icon of the currently active window, if any.

Returns

the icon (maybe not inside a dock, maybe NULL).

5.5.3.4 Icon∗ cairo_dock_get_appli_icon (GldiWindowActor ∗ *actor*)

Get the icon of a given window, if any.

Parameters

actor	the window actor

Returns

the icon (maybe not inside a dock, maybe NULL).

5.5.3.5 void cairo_dock_foreach_appli_icon (GldiIconFunc *pFunction,* gpointer *pUserData*)

Run a function on all Appli icons.

Parameters

pFunction	function to be called
pUserData	data passed to the function.

5.6 cairo-dock-cinnamon-integration.h File Reference

5.6.1 Detailed Description

This class implements the integration of Cinnamon inside Cairo-Dock.

5.7 cairo-dock-class-manager.h File Reference

Data Structures

- struct _CairoDockClassAppli

  Definition of a Class of application.

Macros

- #define cairo_dock_register_class(cDesktopFile)

Functions

- void gldi_window_foreach_inhibitor (GldiWindowActor ∗actor, GldiIconRFunc callback, gpointer data)
- void cairo_dock_set_data_from_class (const gchar ∗cClass, Icon ∗pIcon)

5.7.1 Detailed Description

This class handles the Class Icons, which are icons pointing to the sub-dock of a class.

This class handles the managment of the applications classes. Classes are used to group the windows of a same program, and to bind a launcher to the launched application.

5.7.2 Macro Definition Documentation

5.7.2.1 #define cairo_dock_register_class(*cDesktopFile*)

Register a class corresponding to a desktop file. Launchers can then derive from the class.

Parameters

cDesktopFile	the desktop file path or name; if it's a name or if the path couldn't be found, it will be searched in the common directories.

Returns

the class ID in a newly allocated string.

5.7.3 Function Documentation

5.7.3.1 void gldi_window_foreach_inhibitor (GldiWindowActor * *actor*, GldiIconRFunc *callback*, gpointer *data*)

Run a function on each Icon that inhibites a given window.

Parameters

actor	the window actor
callback	function to be called
data	data passed to the callback

5.7.3.2 void cairo_dock_set_data_from_class (const gchar ∗ *cClass*, Icon ∗ *pIcon*)

Make a launcher derive from a class. Parameters of the icon that are not NULL are not overwritten.

Parameters

cClass	the class name
pIcon	the icon

5.8 cairo-dock-compiz-integration.h File Reference

5.8.1 Detailed Description

This class implements the integration of Compiz inside Cairo-Dock.

5.9 cairo-dock-config.h File Reference

Functions

- void cairo_dock_load_current_theme (void)
- gboolean cairo_dock_is_loading (void)
- void cairo_dock_decrypt_string (const gchar ∗cEncryptedString, gchar ∗∗cDecryptedString)
- void cairo_dock_encrypt_string (const gchar ∗cDecryptedString, gchar ∗∗cEncryptedString)

5.9.1 Detailed Description

This class manages the configuration system of Cairo-Dock. Cairo-Dock and any items (icons, root docks, modules, etc) are configured by conf files. Conf files containes some information usable by the GUI manager to build a corresponding config panel and update the conf file automatically, which relieves you from this thankless task.

5.9.2 Function Documentation

5.9.2.1 void cairo_dock_load_current_theme (void)

Load the current theme. This will (re)load all the parameters of Cairo-Dock and all the plug-ins, as if you just started the dock.

5.9.2.2 gboolean cairo_dock_is_loading (void)

Say if Cairo-Dock is loading.

Returns

TRUE if the global config is being loaded (this happens when a theme is loaded).

5.9.2.3 void cairo_dock_decrypt_string (const gchar * *cEncryptedString,* gchar ** *cDecryptedString*)

Decrypt a string (uses DES-encryption from libcrypt).

Parameters

cEncryptedString	the encrypted string.
cDecryptedString	the decrypted string.

5.9.2.4 void cairo_dock_encrypt_string (const gchar ∗ *cDecryptedString,* gchar ∗∗ *cEncryptedString*)

Encrypt a string (uses DES-encryption from libcrypt).

Parameters

cDecryptedString	the decrypted string.
cEncryptedString	the encrypted string.

5.10 cairo-dock-container.h File Reference

Data Structures

- struct _GldiContainer

  Definition of a Container, whom derive Dock, Desklet, Dialog and FlyingContainer.
- struct _GldiContainerManagerBackend

  Definition of the Container backend. It defines some operations that should be, but are not, provided by GTK.

Macros

- #define CAIRO_CONTAINER(p)

  Get the Container part of a pointer.
- #define CAIRO_DOCK_IS_CONTAINER(obj)
- #define gldi_container_enable_drop(pContainer, pCallBack, data)

Enumerations

- enum GldiContainerNotifications {
  NOTIFICATION_BUILD_CONTAINER_MENU,
  NOTIFICATION_BUILD_ICON_MENU,
  NOTIFICATION_CLICK_ICON,
  NOTIFICATION_DOUBLE_CLICK_ICON,
  NOTIFICATION_MIDDLE_CLICK_ICON,
  NOTIFICATION_SCROLL_ICON,
  NOTIFICATION_ENTER_ICON,
  NOTIFICATION_START_DRAG_DATA,
  NOTIFICATION_DROP_DATA,
  NOTIFICATION_MOUSE_MOVED,
  NOTIFICATION_KEY_PRESSED,
  NOTIFICATION_UPDATE,
  NOTIFICATION_UPDATE_SLOW,
  NOTIFICATION_RENDER }

  signals
- enum CairoDockTypeHorizontality

  Main orientation of a container.

Functions

- void gldi_container_reserve_space (GldiContainer ∗pContainer, int left, int right, int top, int bottom, int left_←
start_y, int left_end_y, int right_start_y, int right_end_y, int top_start_x, int top_end_x, int bottom_start_x, int bottom_end_x)
- int gldi_container_get_current_desktop_index (GldiContainer ∗pContainer)
- void gldi_container_move (GldiContainer ∗pContainer, int iNumDesktop, int iAbsolutePositionX, int i←
AbsolutePositionY)
- gboolean gldi_container_is_active (GldiContainer ∗pContainer)
- void gldi_container_present (GldiContainer ∗pContainer)
- void cairo_dock_redraw_container (GldiContainer ∗pContainer)
- void cairo_dock_redraw_container_area (GldiContainer ∗pContainer, GdkRectangle ∗pArea)
- void cairo_dock_redraw_icon (Icon ∗icon)
- void gldi_container_notify_drop_data (GldiContainer ∗pContainer, gchar ∗cReceivedData, Icon ∗pPointed←
Icon, double fOrder)
- GtkWidget ∗ gldi_container_build_menu (GldiContainer ∗pContainer, Icon ∗icon)

5.10.1 Detailed Description

This class defines the Containers, that are classic or hardware accelerated animated windows, and exposes common functions, such as redrawing a part of a container or popping a menu on a container.

A Container is a rectangular on-screen located surface, has the notion of orientation, can hold external datas, monitors the mouse position, and has its own animation loop.

Docks, Desklets, Dialogs, and Flying-containers all derive from Containers.

5.10.2 Macro Definition Documentation

5.10.2.1 #define CAIRO_DOCK_IS_CONTAINER(*obj*)

Say if an object is a Container.

Parameters

obj	the object.

Returns

TRUE if the object is a Container.

5.10.2.2 #define gldi_container_enable_drop(*pContainer, pCallBack, data*)

Enable a Container to accept drag-and-drops.

Parameters

pContainer	a container.
pCallBack	the function that will be called when some data is received.
data	data passed to the callback.

5.10.3 Enumeration Type Documentation

5.10.3.1 enum GldiContainerNotifications

signals

Enumerator

NOTIFICATION_BUILD_CONTAINER_MENU notification called when the menu is being built on a container. data : {Icon, GldiContainer, GtkMenu, gboolean∗}

NOTIFICATION_BUILD_ICON_MENU notification called when the menu is being built on an icon (possibly NULL). data : {Icon, GldiContainer, GtkMenu}

NOTIFICATION_CLICK_ICON notification called when use clicks on an icon data : {Icon, CairoDock, int}

NOTIFICATION_DOUBLE_CLICK_ICON notification called when the user double-clicks on an icon. data : {Icon, CairoDock}

NOTIFICATION_MIDDLE_CLICK_ICON notification called when the user middle-clicks on an icon. data : {Icon, CairoDock}

NOTIFICATION_SCROLL_ICON notification called when the user scrolls on an icon. data : {Icon, CairoDock, int}

NOTIFICATION_ENTER_ICON notification called when the mouse enters an icon. data : {Icon, CairoDock, gboolean∗}

NOTIFICATION_START_DRAG_DATA notification called when the mouse enters a dock while dragging an object.

NOTIFICATION_DROP_DATA notification called when something is dropped inside a container. data ↩ : {gchar∗, Icon, double∗, CairoDock}

NOTIFICATION_MOUSE_MOVED notification called when the mouse has moved inside a container.

NOTIFICATION_KEY_PRESSED notification called when a key is pressed in a container that has the focus.

NOTIFICATION_UPDATE notification called for the fast rendering loop on a container.

NOTIFICATION_UPDATE_SLOW notification called for the slow rendering loop on a container.

NOTIFICATION_RENDER notification called when a container is rendered.

5.10.4 Function Documentation

5.10.4.1 void gldi_container_reserve_space (GldiContainer ∗ *pContainer*, int *left*, int *right*, int *top*, int *bottom*, int *left_start_y*, int *left_end_y*, int *right_start_y*, int *right_end_y*, int *top_start_x*, int *top_end_x*, int *bottom_start_x*, int *bottom_end_x*)

Reserve a space on the screen for a Container; other windows won't overlap this space when maximised.

Parameters

pContainer	the container
left	
right	
top	
bottom	
left_start_y	
left_end_y	
right_start_y	
right_end_y	
top_start_x	
top_end_x	
bottom_start_x	
bottom_end_x	

5.10.4.2 int gldi_container_get_current_desktop_index (GldiContainer ∗ *pContainer*)

Get the desktop and viewports a Container is placed on.

Parameters

pContainer	the container

Returns

an index representing the desktop and viewports.

5.10.4.3 void gldi_container_move (GldiContainer ∗ *pContainer,* int *iNumDesktop,* int *iAbsolutePositionX,* int *iAbsolutePositionY*)

Move a Container to a given desktop, viewport, and position (similar to gtk_window_move except that the position is defined on the whole desktop (made of all viewports); it's only useful if the Container is sticky).

Parameters

pContainer	the container
iNumDesktop	desktop number
iAbsolute←PositionX	horizontal position on the virtual screen
iAbsolute←PositionY	vertical position on the virtual screen

5.10.4.4 gboolean gldi_container_is_active (GldiContainer ∗ *pContainer*)

Tell if a Container is the current active window (similar to gtk_window_is_active but actually works).

Parameters

pContainer	the container

Returns

TRUE if the Container is the current active window.

5.10.4.5 void gldi_container_present (GldiContainer ∗ *pContainer*)

Show a Container and make it take the focus (similar to gtk_window_present, but bypasses the WM focus steal prevention).

Parameters

pContainer	the container

5.10.4.6 void cairo_dock_redraw_container (GldiContainer ∗ *pContainer*)

Clear and trigger the redraw of a Container.

Parameters

pContainer	the Container to redraw.

5.10.4.7 void cairo_dock_redraw_container_area (GldiContainer ∗ *pContainer,* GdkRectangle ∗ *pArea*)

Clear and trigger the redraw of a part of a container.

Parameters

pContainer	the Container to redraw.
pArea	the zone to redraw.

5.10.4.8 void cairo_dock_redraw_icon (Icon ∗ *icon*)

Clear and trigger the redraw of an Icon. The drawing is not done immediately, but when the expose event is received.

Parameters

icon	l'icone a retracer.

5.10.4.9 void gldi_container_notify_drop_data (GldiContainer ∗ *pContainer,* gchar ∗ *cReceivedData,* Icon ∗ *pPointedIcon,* double *fOrder*)

Notify everybody that a drop has just occured.

Parameters

cReceivedData	the dropped data.
pPointedIcon	the icon which was pointed when the drop occured.
fOrder	the order of the icon if the drop occured on it, or LAST_ORDER if the drop occured between 2 icons.
pContainer	the container of the icon

5.10.4.10 GtkWidget∗ gldi_container_build_menu (GldiContainer ∗ *pContainer,* Icon ∗ *icon*)

Build the main menu of a Container.

Parameters

icon	the icon that was left-clicked, or NULL if none.
pContainer	the container that was left-clicked.

Returns

the menu.

5.11 cairo-dock-core.h File Reference

5.11.1 Detailed Description

This class instanciates the different core managers.

5.12 cairo-dock-data-renderer-manager.h File Reference

Macros

- #define GLDI_OBJECT_IS_DATA_RENDERER(obj)

Functions

- CairoDockGLFont ∗ cairo_dock_get_default_data_renderer_font (void)

5.12.1 Detailed Description

This class manages the list of available Data Renderers and their global ressources.

5.12.2 Macro Definition Documentation

5.12.2.1 #define GLDI_OBJECT_IS_DATA_RENDERER(*obj*)

Say if an object is a DataRenderer.

Parameters

obj	the object.

Returns

TRUE if the object is a DataRenderer.

5.12.3 Function Documentation

5.12.3.1 CairoDockGLFont∗ cairo_dock_get_default_data_renderer_font (void)

Get the default GLX font for Data Renderer. It can render strings of ASCII characters fastly. Don't destroy it.

Returns

the default GLX font

5.13 cairo-dock-data-renderer.h File Reference

Data Structures

- struct _CairoDataRendererAttribute

  Generic DataRenderer attributes structure. The attributes of any implementation of a DataRenderer will derive from this class.
- struct _CairoDataRendererInterface

  Interface of a DataRenderer.
- struct _CairoDataRenderer

  Generic DataRenderer. Any implementation of a DataRenderer will derive from this class.

Macros

- #define cairo_dock_get_icon_data_renderer(pIcon)
- #define CAIRO_DATA_RENDERER(r)
- #define cairo_data_renderer_get_data(pRenderer)
- #define CAIRO_DATA_RENDERER_ATTRIBUTE(pAttr)
- #define cairo_data_renderer_get_nb_values(pRenderer)
- #define cairo_data_renderer_get_min_value(pRenderer, i)
- #define cairo_data_renderer_get_max_value(pRenderer, i)

- #define cairo_data_renderer_get_value(pRenderer, i, t)
- #define cairo_data_renderer_get_current_value(pRenderer, i)
- #define cairo_data_renderer_get_previous_value(pRenderer, i)
- #define cairo_data_renderer_get_normalized_value(pRenderer, i, t)
- #define cairo_data_renderer_get_normalized_current_value(pRenderer, i)
- #define cairo_data_renderer_get_normalized_previous_value(pRenderer, i)
- #define cairo_data_renderer_get_normalized_current_value_with_latency(pRenderer, i)
- #define cairo_data_renderer_format_value_full(pRenderer, i, cBuffer)
- #define cairo_data_renderer_format_value(pRenderer, i)

Typedefs

- typedef void(∗ CairoDataRendererFormatValueFunc)(CairoDataRenderer ∗pRenderer, int iNumValue, gchar ∗cFormatBuffer, int iBufferLength, gpointer data)

  Prototype of a function used to format the values in a short readable format (to be displayed as quick-info).

Functions

- CairoDockGLFont ∗ cairo_dock_get_default_data_renderer_font (void)
- void cairo_dock_add_new_data_renderer_on_icon (Icon ∗pIcon, GldiContainer ∗pContainer, CairoData←RendererAttribute ∗pAttribute)
- void cairo_dock_render_new_data_on_icon (Icon ∗pIcon, GldiContainer ∗pContainer, cairo_t ∗pCairo←Context, double ∗pNewValues)
- void cairo_dock_remove_data_renderer_on_icon (Icon ∗pIcon)
- void cairo_dock_reload_data_renderer_on_icon (Icon ∗pIcon, GldiContainer ∗pContainer)
- void cairo_dock_resize_data_renderer_history (Icon ∗pIcon, int iNewMemorySize)
- void cairo_dock_refresh_data_renderer (Icon ∗pIcon, GldiContainer ∗pContainer)

5.13.1 Detailed Description

This class defines the Data Renderer structure and API. A Data Renderer is a generic way to display a set of values on an icon. For instance you could represent the (cpu, memory, temperature) evolution over the time.

You bind a Data Renderer with /ref cairo_dock_add_new_data_renderer_on_icon. You can specify some attributes of the Data Renderer, especially the model that will be used; currently, 3 models are available: "gauge", "graph" and "progressbar".

You then feed the Data Renderer with /ref cairo_dock_render_new_data_on_icon, providing it the correct number of values.

To remove the Data Renderer from an icon, use /ref cairo_dock_remove_data_renderer_on_icon.

5.13.2 Macro Definition Documentation

5.13.2.1 #define cairo_dock_get_icon_data_renderer(*pIcon*)

Structure Access

5.13.2.2 #define CAIRO_DATA_RENDERER(*r*)

Get the elementary part of a Data Renderer

Parameters

r	a high level data renderer

Returns

a CairoDataRenderer∗

5.13.2.3 #define cairo_data_renderer_get_data(*pRenderer*)

Get the data of a Data Renderer

Parameters

pRenderer	a data renderer

Returns

a CairoDataToRenderer∗

5.13.2.4 #define CAIRO_DATA_RENDERER_ATTRIBUTE(*pAttr*)

Get the elementary part of a Data Renderer Attribute

Parameters

pAttr	a high level data renderer attribute

Returns

a CairoDataRendererAttribute∗

5.13.2.5 #define cairo_data_renderer_get_nb_values(*pRenderer*)

Get the number of values a DataRenderer displays. It's also the size of any of its arrays.

Parameters

pRenderer	a data renderer

Returns

number of values a DataRenderer displays

5.13.2.6 #define cairo_data_renderer_get_min_value(*pRenderer, i*)

Data Access Get the lower range of the i-th value.

Parameters

pRenderer	a data renderer
i	the number of the value

Returns

a double

5.13.2.7 #define cairo_data_renderer_get_max_value(*pRenderer*, *i*)

Get the upper range of the i-th value.

Parameters

pRenderer	a data renderer
i	the number of the value

Returns

a double

5.13.2.8 #define cairo_data_renderer_get_value(*pRenderer, i, t*)

Get the i-th value at the time t.

Parameters

pRenderer	a data renderer
i	the number of the value
t	the time (in number of steps)

Returns

a double

5.13.2.9 #define cairo_data_renderer_get_current_value(*pRenderer, i*)

Get the current i-th value.

Parameters

pRenderer	a data renderer
i	the number of the value

Returns

a double

5.13.2.10 #define cairo_data_renderer_get_previous_value(*pRenderer, i*)

Get the previous i-th value.

Parameters

pRenderer	a data renderer
i	the number of the value

Returns

a double

5.13.2.11 #define cairo_data_renderer_get_normalized_value(*pRenderer, i, t*)

Get the normalized i-th value (between 0 and 1) at the time t.

Parameters

pRenderer	a data renderer
i	the number of the value
t	the time (in number of steps)

Returns

a double in [0,1]

5.13.2.12 #define cairo_data_renderer_get_normalized_current_value(*pRenderer,* *i*)

Get the normalized current i-th value (between 0 and 1).

Parameters

pRenderer	a data renderer
i	the number of the value

Returns

a double in [0,1]

5.13.2.13 #define cairo_data_renderer_get_normalized_previous_value(*pRenderer,* *i*)

Get the normalized previous i-th value (between 0 and 1).

Parameters

pRenderer	a data renderer
i	the number of the value

Returns

a double in [0,1]

5.13.2.14 #define cairo_data_renderer_get_normalized_current_value_with_latency(*pRenderer,* *i*)

Get the normalized current i-th value (between 0 and 1), taking into account the latency of the smooth movement.

Parameters

pRenderer	a data renderer
i	the number of the value

Returns

a double in [0,1]

5.13.2.15 #define cairo_data_renderer_format_value_full(*pRenderer,* *i,* *cBuffer*)

Data Format Write a value in a readable text format.

Parameters

pRenderer	a data renderer
i	the number of the value
cBuffer	a buffer where to write

5.13.2.16 #define cairo_data_renderer_format_value(*pRenderer, i*)

Write a value in a readable text format in the renderer text buffer.

Parameters

pRenderer	a data renderer
i	the number of the value

5.13.3 Function Documentation

5.13.3.1 CairoDockGLFont∗ cairo_dock_get_default_data_renderer_font (void)

Renderer manipulation Get the default GLX font for Data Renderer. It can render strings of digits from 0 to 9. Don't destroy it.

Returns

the default GLX font

5.13.3.2 void cairo_dock_add_new_data_renderer_on_icon (Icon ∗ *pIcon,* GldiContainer ∗ *pContainer,* CairoDataRendererAttribute ∗ *pAttribute*)

Add a Data Renderer on an icon. If the icon already has a Data Renderer, it is replaced by the new one, keeping the history alive.

Parameters

pIcon	the icon
pContainer	the icon's container
pAttribute	attributes defining the Renderer

5.13.3.3 void cairo_dock_render_new_data_on_icon (Icon ∗ *pIcon,* GldiContainer ∗ *pContainer,* cairo_t ∗ *pCairoContext,* double ∗ *pNewValues*)

Draw the current values associated with the Renderer on the icon.

Parameters

pIcon	the icon
pContainer	the icon's container
pCairoContext	a drawing context on the icon
pNewValues	a set a new values (must be of the size defined on the creation of the Renderer)

5.13.3.4 void cairo_dock_remove_data_renderer_on_icon (Icon ∗ *pIcon*)

Remove the Data Renderer of an icon. All the allocated ressources will be freed.

Parameters

pIcon	the icon

5.13.3.5 void cairo_dock_reload_data_renderer_on_icon (Icon ∗ *pIcon*, GldiContainer ∗ *pContainer*)

Reload the Data Renderer of an icon, keeping the history and the attributes. This is intended to be used when the icon size changes.

Parameters

pIcon	the icon
pContainer	the icon's container

5.13.3.6 void cairo_dock_resize_data_renderer_history (Icon ∗ *pIcon*, int *iNewMemorySize*)

Resize the history of a DataRenderer of an icon, that is to say change the number of previous values that are remembered by the DataRenderer.

Parameters

pIcon	the icon
iNewMemory←Size	the new size of history

5.13.3.7 void cairo_dock_refresh_data_renderer (Icon ∗ *pIcon*, GldiContainer ∗ *pContainer*)

Redraw the DataRenderer of an icon, with the current values.

Parameters

pIcon	the icon
pContainer	the icon's container

5.14 cairo-dock-dbus.h File Reference

Macros

- #define cairo_dock_dbus_get_property_in_value(pDbusProxy, cInterface, cProperty, pProperties)

  deprecated...

Functions

- DBusGConnection ∗ cairo_dock_get_session_connection (void)
- gboolean cairo_dock_register_service_name (const gchar ∗cServiceName)
- gboolean cairo_dock_dbus_is_enabled (void)
- DBusGProxy ∗ cairo_dock_create_new_session_proxy (const char ∗name, const char ∗path, const char ∗interface)
- DBusGProxy ∗ cairo_dock_create_new_system_proxy (const char ∗name, const char ∗path, const char ∗interface)
- gboolean cairo_dock_dbus_detect_application (const gchar ∗cName)
- gboolean cairo_dock_dbus_detect_system_application (const gchar ∗cName)
- gboolean cairo_dock_dbus_get_boolean (DBusGProxy ∗pDbusProxy, const gchar ∗cAccessor)

- guint cairo_dock_dbus_get_uinteger (DBusGProxy ∗pDbusProxy, const gchar ∗cAccessor)
- int cairo_dock_dbus_get_integer (DBusGProxy ∗pDbusProxy, const gchar ∗cAccessor)
- gchar ∗ cairo_dock_dbus_get_string (DBusGProxy ∗pDbusProxy, const gchar ∗cAccessor)
- gchar ∗∗ cairo_dock_dbus_get_string_list (DBusGProxy ∗pDbusProxy, const gchar ∗cAccessor)
- guchar ∗ cairo_dock_dbus_get_uchar (DBusGProxy ∗pDbusProxy, const gchar ∗cAccessor)
- void cairo_dock_dbus_call (DBusGProxy ∗pDbusProxy, const gchar ∗cCommand)

5.14.1 Detailed Description

This class defines numerous convenient functions to use DBus inside Cairo-Dock. DBus is used to communicate and interact with other running applications.

5.14.2 Function Documentation

5.14.2.1 DBusGConnection∗ cairo_dock_get_session_connection (void)

Get the connection to the 'session' Bus.

Returns

the connection to the bus.

5.14.2.2 gboolean cairo_dock_register_service_name (const gchar ∗ *cServiceName*)

Register a new service on the session bus.

Parameters

cServiceName	name of the service.

Returns

TRUE in case of success, false otherwise.

5.14.2.3 gboolean cairo_dock_dbus_is_enabled (void)

Say if the bus is available or not.

Returns

TRUE if the connection to the bus has been established.

5.14.2.4 DBusGProxy∗ cairo_dock_create_new_session_proxy (const char ∗ *name,* const char ∗ *path,* const char ∗ *interface*)

Create a new proxy for the 'session' connection.

Parameters

name	a name on the bus.

path	the path.
interface	name of the interface.

Returns

the newly created proxy. Use g_object_unref when your done with it.

5.14.2.5 DBusGProxy∗ cairo_dock_create_new_system_proxy (const char ∗ *name,* const char ∗ *path,* const char ∗ *interface*)

Create a new proxy for the 'system' connection.

Parameters

name	a name on the bus.
path	the path.
interface	name of the interface.

Returns

the newly created proxy. Use g_object_unref when your done with it.

5.14.2.6 gboolean cairo_dock_dbus_detect_application (const gchar ∗ *cName*)

Detect if an application is currently running on Session bus.

Parameters

cName	name of the application.

Returns

TRUE if the application is running and has a service on the bus.

5.14.2.7 gboolean cairo_dock_dbus_detect_system_application (const gchar ∗ *cName*)

Detect if an application is currently running on System bus.

Parameters

cName	name of the application.

Returns

TRUE if the application is running and has a service on the bus.

5.14.2.8 gboolean cairo_dock_dbus_get_boolean (DBusGProxy ∗ *pDbusProxy,* const gchar ∗ *cAccessor*)

Get the value of a 'boolean' parameter on the bus.

Parameters

pDbusProxy	proxy to the connection.
cAccessor	name of the accessor.

Returns

the value of the parameter.

5.14.2.9 guint cairo_dock_dbus_get_uinteger (DBusGProxy ∗ *pDbusProxy,* const gchar ∗ *cAccessor*)

Get the value of an 'unsigned integer' parameter non signe on the bus.

Parameters

pDbusProxy	proxy to the connection.
cAccessor	name of the accessor.

Returns

the value of the parameter.

5.14.2.10 int cairo_dock_dbus_get_integer (DBusGProxy ∗ *pDbusProxy,* const gchar ∗ *cAccessor*)

Get the value of a 'integer' parameter on the bus.

Parameters

pDbusProxy	proxy to the connection.
cAccessor	name of the accessor.

Returns

the value of the parameter.

5.14.2.11 gchar∗ cairo_dock_dbus_get_string (DBusGProxy ∗ *pDbusProxy,* const gchar ∗ *cAccessor*)

Get the value of a 'string' parameter on the bus.

Parameters

pDbusProxy	proxy to the connection.
cAccessor	name of the accessor.

Returns

the value of the parameter, to be freeed with g_free.

5.14.2.12 gchar∗∗ cairo_dock_dbus_get_string_list (DBusGProxy ∗ *pDbusProxy,* const gchar ∗ *cAccessor*)

Get the value of a 'string list' parameter on the bus.

Parameters

pDbusProxy	proxy to the connection.
cAccessor	name of the accessor.

Returns

the value of the parameter, to be freeed with g_strfreev.

5.14.2.13 guchar∗ cairo_dock_dbus_get_uchar (DBusGProxy ∗ *pDbusProxy,* const gchar ∗ *cAccessor*)

Get the value of an 'unsigned char' parameter on the bus.

Parameters

pDbusProxy	proxy to the connection.
cAccessor	name of the accessor.

Returns

the value of the parameter.

5.14.2.14 void cairo_dock_dbus_call (DBusGProxy ∗ *pDbusProxy,* const gchar ∗ *cCommand*)

Call a command on the bus.

Parameters

pDbusProxy	proxy to the connection.
cCommand	name of the commande.

5.15 cairo-dock-default-view.h File Reference

5.15.1 Detailed Description

This class implements the Dock rendering interface and provides the "default" view.

5.16 cairo-dock-desklet-factory.h File Reference

Data Structures

- struct _CairoDeskletDecoration

  Decoration of a Desklet.
- struct _CairoDeskletAttr

  Configuration attributes of a Desklet.
- struct _CairoDeskletRenderer

  Definition of a Desklet's renderer.
- struct _CairoDesklet

  Definition of a Desklet, which derives from a Container.

Macros

- #define GLDI_OBJECT_IS_DESKLET(obj)
- #define CAIRO_DESKLET(pContainer)
- #define gldi_desklet_add_interactive_widget(pDesklet, pInteractiveWidget)

Enumerations

- enum CairoDeskletVisibility {
  CAIRO_DESKLET_NORMAL,
  CAIRO_DESKLET_KEEP_ABOVE,
  CAIRO_DESKLET_KEEP_BELOW,
  CAIRO_DESKLET_ON_WIDGET_LAYER,
  CAIRO_DESKLET_RESERVE_SPACE }

  Type of accessibility of a Desklet.

Functions

- CairoDesklet ∗ gldi_desklet_new (CairoDeskletAttr ∗attr)
- void gldi_desklet_add_interactive_widget_with_margin (CairoDesklet ∗pDesklet, GtkWidget ∗pInteractive←Widget, int iRightMargin)
- void gldi_desklet_set_margin (CairoDesklet ∗pDesklet, int iRightMargin)
- GtkWidget ∗ gldi_desklet_steal_interactive_widget (CairoDesklet ∗pDesklet)
- void gldi_desklet_hide (CairoDesklet ∗pDesklet)
- void gldi_desklet_show (CairoDesklet ∗pDesklet)
- void gldi_desklet_set_accessibility (CairoDesklet ∗pDesklet, CairoDeskletVisibility iVisibility, gboolean b←SaveState)
- void gldi_desklet_set_sticky (CairoDesklet ∗pDesklet, gboolean bSticky)
- void gldi_desklet_lock_position (CairoDesklet ∗pDesklet, gboolean bPositionLocked)

5.16.1 Detailed Description

This file is a part of the Cairo-Dock project Login : ctaf42@gmail.com Started on Sun Jan 27 18:35:38 2008 Cedric GESTES $Id$

Author(s)

- Cedric GESTES ctaf42@gmail.com
- Fabrice REY

Copyright : (C) 2008 Cedric GESTES E-mail : see the 'copyright' file.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with this program. If not, see http://www.gnu.org/licenses/. This class defines the Desklets, that are Widgets placed directly on your desktop. A Desklet is a container that holds 1 applet's icon plus an optionnal list of other icons and an optionnal GTK widget, has a decoration, suports several accessibility types (like Compiz Widget Layer), and has a renderer. Desklets can be resized or moved directly with the mouse, and can be rotated in the 3 directions of space. To actually create or destroy a Desklet, use the Desklet Manager's functoins in cairo-dock-desklet-manager.h.

5.16.2 Macro Definition Documentation

5.16.2.1 #define GLDI_OBJECT_IS_DESKLET(*obj*)

Say if an object is a Desklet.

Parameters

obj	the object.

Returns

TRUE if the object is a Desklet.

5.16.2.2 #define CAIRO_DESKLET(*pContainer*)

Cast a Container into a Desklet.

Parameters

pContainer	the container.

Returns

the desklet.

5.16.2.3 #define gldi_desklet_add_interactive_widget(*pDesklet, pInteractiveWidget*)

Add a GtkWidget to a desklet. Only 1 widget is allowed per desklet, if you need more, you can just use a Gtk↵ Container, and place as many widget as you want inside.

Parameters

pInteractive↵ Widget	the widget to add.
pDesklet	the desklet.

5.16.3 Enumeration Type Documentation

5.16.3.1 enum CairoDeskletVisibility

Type of accessibility of a Desklet.

Enumerator

CAIRO_DESKLET_NORMAL Normal, like normal window.

CAIRO_DESKLET_KEEP_ABOVE always above

CAIRO_DESKLET_KEEP_BELOW always below

CAIRO_DESKLET_ON_WIDGET_LAYER on the Compiz widget layer

CAIRO_DESKLET_RESERVE_SPACE prevent other windows form overlapping it

5.16.4 Function Documentation

5.16.4.1 CairoDesklet∗ gldi_desklet_new (CairoDeskletAttr ∗ *attr*)

Create a new desklet.

Parameters

attr	the attributes of the desklet

Returns

the desklet.

5.16.4.2 void gldi_desklet_add_interactive_widget_with_margin (CairoDesklet ∗ *pDesklet,* GtkWidget ∗ *pInteractiveWidget,* int *iRightMargin*)

Add a GtkWidget to a desklet. Only 1 widget is allowed per desklet, if you need more, you can just use a Gtk↩ Container, and place as many widget as you want inside.

Parameters

pInteractive↩ Widget	the widget to add.
pDesklet	the desklet.
iRightMargin	right margin, in pixels, useful to keep a clickable zone on the desklet, or 0 if you don't want a margin.

5.16.4.3 void gldi_desklet_set_margin (CairoDesklet ∗ *pDesklet,* int *iRightMargin*)

Set the right margin of a desklet. This is useful to keep a clickable zone on the desklet when you put a GTK widget inside.

Parameters

pDesklet	the desklet.
iRightMargin	right margin, in pixels.

5.16.4.4 GtkWidget∗ gldi_desklet_steal_interactive_widget (CairoDesklet ∗ *pDesklet*)

Detach the interactive widget from a desklet. The widget can then be placed anywhere after that. You have to unref it after you placed it into a container, or to destroy it.

Parameters

pDesklet	the desklet with an interactive widget.

Returns

the widget.

5.16.4.5 void gldi_desklet_hide (CairoDesklet ∗ *pDesklet*)

Hide a desklet.

Parameters

pDesklet	the desklet.

5.16.4.6 void gldi_desklet_show (CairoDesklet ∗ *pDesklet*)

Show a desklet, and give it the focus.

Parameters

pDesklet	the desklet.

5.16.4.7 void gldi_desklet_set_accessibility (CairoDesklet ∗ *pDesklet,* CairoDeskletVisibility *iVisibility,* gboolean *bSaveState*)

Set a desklet's accessibility. For Widget Layer, the WM must support it and the correct rule must be set up in the WM (for instance for Compiz : class=Cairo-dock & type=utility). The function automatically sets up the rule for Compiz (if Dbus is activated).

Parameters

pDesklet	the desklet.
iVisibility	the new accessibility.
bSaveState	whether to save the new state in the conf file.

5.16.4.8 void gldi_desklet_set_sticky (CairoDesklet ∗ *pDesklet,* gboolean *bSticky*)

Set a desklet sticky (i.e. visible on all desktops), or not. In case the desklet is set unsticky, its current desktop/viewport is saved.

Parameters

pDesklet	the desklet.
bSticky	whether the desklet should be sticky or not.

5.16.4.9 void gldi_desklet_lock_position (CairoDesklet ∗ *pDesklet,* gboolean *bPositionLocked*)

Lock the position of a desklet. This makes the desklet impossible to rotate, drag with the mouse, or retach to the dock. The new state is saved in conf.

Parameters

pDesklet	the desklet.
bPositionLocked	whether the position should be locked or not.

5.17 cairo-dock-desklet-manager.h File Reference

Typedefs

- typedef gboolean(∗ GldiDeskletForeachFunc)(CairoDesklet ∗pDesklet, gpointer data)

  Definition of a function that runs through all desklets.

Enumerations

- enum CairoDeskletNotifications {
  NOTIFICATION_ENTER_DESKLET,
  NOTIFICATION_LEAVE_DESKLET,
  NOTIFICATION_CONFIGURE_DESKLET }

  signals

Functions

- CairoDesklet ∗ gldi_desklets_foreach (GldiDeskletForeachFunc pCallback, gpointer user_data)
- void gldi_desklets_foreach_icons (GldiIconFunc pFunction, gpointer pUserData)
- void gldi_desklets_set_visible (gboolean bOnWidgetLayerToo)
- void gldi_desklets_set_visibility_to_default (void)

5.17.1 Detailed Description

This file is a part of the Cairo-Dock project

Login : `ctaf42@gmail.com` Started on Sun Jan 27 18:35:38 2008 Cedric GESTES $Id$

Author(s)

- Cedric GESTES `ctaf42@gmail.com`
- Fabrice REY

This class manages the Desklets, that are Widgets placed directly on your desktop. A Desklet is a container that holds 1 applet's icon plus an optionnal list of other icons and an optionnal GTK widget, has a decoration, suports several accessibility types (like Compiz Widget Layer), and has a renderer. Desklets can be resized or moved directly with the mouse, and can be rotated in the 3 directions of space.

5.17.2 Enumeration Type Documentation

5.17.2.1 enum **CairoDeskletNotifications**

signals

Enumerator

NOTIFICATION_ENTER_DESKLET notification called when the mouse enters a desklet.

NOTIFICATION_LEAVE_DESKLET notification called when the mouse leave a desklet.

NOTIFICATION_CONFIGURE_DESKLET notification called when a desklet is resized or moved on the screen.

5.17.3 Function Documentation

5.17.3.1 **CairoDesklet**∗ gldi_desklets_foreach (**GldiDeskletForeachFunc** *pCallback,* gpointer *user_data*)

Run a function through all the desklets. If the callback returns TRUE, then the loop ends and the function returns the current desklet.

Parameters

pCallback	function to be called on eash desklet. If it returns TRUE, the loop ends and the function returns the current desklet.
user_data	data to be passed to the callback.

Returns

the found desklet, or NULL.

5.17.3.2 void gldi_desklets_foreach_icons (GldiIconFunc *pFunction*, gpointer *pUserData*)

Execute an action on all icons being inside a desklet.

Parameters

pFunction	the action.
pUserData	data passed to the callback.

5.17.3.3 void gldi_desklets_set_visible (gboolean *bOnWidgetLayerToo*)

Make all desklets visible. Their accessibility is set to CAIRO_DESKLET_NORMAL.

Parameters

bOnWidget↩ LayerToo	TRUE if you want to act on the desklet that are on the WidgetLayer as well.

5.17.3.4 void gldi_desklets_set_visibility_to_default (void)

Reset the desklets accessibility to the state defined in their conf file.

5.18 cairo-dock-desktop-manager.h File Reference

Data Structures

- struct _GldiDesktopManagerBackend

  Definition of the Desktop Manager backend.
- struct _GldiDesktopBackground

  Definition of a Desktop Background Buffer. It has a reference count so that it can be shared across all the lib.

Enumerations

- enum CairoDesktopNotifications {
  NOTIFICATION_DESKTOP_CHANGED,
  NOTIFICATION_DESKTOP_GEOMETRY_CHANGED,
  NOTIFICATION_DESKTOP_VISIBILITY_CHANGED,
  NOTIFICATION_KBD_STATE_CHANGED,
  NOTIFICATION_DESKTOP_NAMES_CHANGED,
  NOTIFICATION_DESKTOP_WALLPAPER_CHANGED,
  NOTIFICATION_SHORTKEY_PRESSED,
  NOTIFICATION_KEYMAP_CHANGED }

  signals

Functions

- void gldi_desktop_manager_register_backend (GldiDesktopManagerBackend ∗pBackend)
- gboolean gldi_desktop_present_class (const gchar ∗cClass)
- gboolean gldi_desktop_present_windows (void)
- gboolean gldi_desktop_present_desktops (void)
- gboolean gldi_desktop_show_widget_layer (void)
- gboolean gldi_desktop_set_on_widget_layer (GldiContainer ∗pContainer, gboolean bOnWidgetLayer)
- void gldi_desktop_get_current (int ∗iCurrentDesktop, int ∗iCurrentViewportX, int ∗iCurrentViewportY)

5.18.1 Detailed Description

This class manages the desktop: screen geometry, current desktop/viewport, etc, and notifies for any change on it.

5.18.2 Enumeration Type Documentation

5.18.2.1 enum CairoDesktopNotifications

signals

Enumerator

NOTIFICATION_DESKTOP_CHANGED notification called when the user switches to another desktop/viewport. data : NULL

NOTIFICATION_DESKTOP_GEOMETRY_CHANGED notification called when the geometry of the desktop has changed (number of viewports/desktops, dimensions). data: resolution-has-changed

NOTIFICATION_DESKTOP_VISIBILITY_CHANGED notification called when the desktop is shown/hidden. data: NULL

NOTIFICATION_KBD_STATE_CHANGED notification called when the state of the keyboard has changed.

NOTIFICATION_DESKTOP_NAMES_CHANGED notification called when the names of the desktops have changed

NOTIFICATION_DESKTOP_WALLPAPER_CHANGED notification called when the wallpaper has changed

NOTIFICATION_SHORTKEY_PRESSED notification called when a shortkey that has been registered by the dock is pressed. data: keycode, modifiers

NOTIFICATION_KEYMAP_CHANGED notification called when the keymap changed, before and after updating it. data: updated

5.18.3 Function Documentation

5.18.3.1 void gldi_desktop_manager_register_backend (GldiDesktopManagerBackend ∗ *pBackend*)

Register a Desktop Manager backend. NULL functions do not overwrite existing ones.

Parameters

pBackend	a Desktop Manager backend; can be freeed after.

5.18.3.2 gboolean gldi_desktop_present_class (const gchar ∗ *cClass*)

Present all the windows of a given class.

Parameters

cClass	the class.

Returns

TRUE on success

5.18.3.3 gboolean gldi_desktop_present_windows (void)

Present all the windows of the current desktop.

Returns

TRUE on success

5.18.3.4 gboolean gldi_desktop_present_desktops (void)

Present all the desktops.

Returns

TRUE on success

5.18.3.5 gboolean gldi_desktop_show_widget_layer (void)

Show the Widget Layer.

Returns

TRUE on success

5.18.3.6 gboolean gldi_desktop_set_on_widget_layer (GldiContainer ∗ *pContainer,* gboolean *bOnWidgetLayer*)

Set a Container to be displayed on the Widget Layer.

Parameters

pContainer	a container.
bOnWidgetLayer	whether to set or unset the option.

Returns

TRUE on success

5.18.3.7 void gldi_desktop_get_current (int ∗ *iCurrentDesktop,* int ∗ *iCurrentViewportX,* int ∗ *iCurrentViewportY*)

Get the current workspace (desktop and viewport).

Parameters

iCurrentDesktop	will be filled with the current desktop number
iCurrent↩ ViewportX	will be filled with the current horizontal viewport number
iCurrent↩ ViewportY	will be filled with the current vertical viewport number

5.19 cairo-dock-dialog-factory.h File Reference

Data Structures

- struct _CairoDialogRenderer

  Definition of a Dialog renderer. It draws the inside of the Dialog.
- struct _CairoDialogDecorator

  Definition of a Dialog/Menu decorator. It draws the frame of the Dialog/Menu.
- struct _CairoDialog

  Definition of a Dialog.

Macros

- #define CAIRO_DOCK_IS_DIALOG(obj)
- #define CAIRO_DIALOG(pContainer)

Functions

- CairoDialog ∗ gldi_dialog_new (CairoDialogAttr ∗pAttribute)
- CairoDialog ∗ gldi_dialog_show (const gchar ∗cText, Icon ∗pIcon, GldiContainer ∗pContainer, double fTime↩ Length, const gchar ∗cIconPath, GtkWidget ∗pInteractiveWidget, CairoDockActionOnAnswerFunc pAction↩ Func, gpointer data, GFreeFunc pFreeDataFunc)
- CairoDialog ∗ gldi_dialog_show_temporary_with_icon_printf (const gchar ∗cText, Icon ∗pIcon, GldiContainer ∗pContainer, double fTimeLength, const gchar ∗cIconPath,...) G_GNUC_PRINTF(1
- CairoDialog CairoDialog ∗ gldi_dialog_show_temporary_with_icon (const gchar ∗cText, Icon ∗pIcon, Gldi↩ Container ∗pContainer, double fTimeLength, const gchar ∗cIconPath)
- CairoDialog ∗ gldi_dialog_show_temporary (const gchar ∗cText, Icon ∗pIcon, GldiContainer ∗pContainer, double fTimeLength)
- CairoDialog ∗ gldi_dialog_show_temporary_with_default_icon (const gchar ∗cText, Icon ∗pIcon, Gldi↩ Container ∗pContainer, double fTimeLength)
- CairoDialog ∗ gldi_dialog_show_with_question (const gchar ∗cText, Icon ∗pIcon, GldiContainer ∗pContainer, const gchar ∗cIconPath, CairoDockActionOnAnswerFunc pActionFunc, gpointer data, GFreeFunc pFree↩ DataFunc)
- CairoDialog ∗ gldi_dialog_show_with_entry (const gchar ∗cText, Icon ∗pIcon, GldiContainer ∗pContainer, const gchar ∗cIconPath, const gchar ∗cTextForEntry, CairoDockActionOnAnswerFunc pActionFunc, gpointer data, GFreeFunc pFreeDataFunc)
- CairoDialog ∗ gldi_dialog_show_with_value (const gchar ∗cText, Icon ∗pIcon, GldiContainer ∗pContainer, const gchar ∗cIconPath, double fValue, double fMaxValue, CairoDockActionOnAnswerFunc pActionFunc, gpointer data, GFreeFunc pFreeDataFunc)
- CairoDialog ∗ gldi_dialog_show_general_message (const gchar ∗cMessage, double fTimeLength)
- int gldi_dialog_show_and_wait (const gchar ∗cText, Icon ∗pIcon, GldiContainer ∗pContainer, const gchar ∗cIconPath, GtkWidget ∗pInteractiveWidget)
- GtkWidget ∗ gldi_dialog_steal_interactive_widget (CairoDialog ∗pDialog)

5.19.1 Detailed Description

This class defines the Dialog container, useful to bring interaction with the user. A Dialog is a container that points to an icon. It contains the following optionnal components :

- a message
- an image on its left
- a interaction widget below it
- some buttons at the bottom.

A Dialog is constructed with a set of attributes grouped inside a _CairoDialogAttribute. It has a Decorator that draws its shape, and a Renderer that draws its content.

To add buttons, you specify a list of images in the attributes. "ok" and "cancel" are key words for the default ok/cancel buttons. You also has to provide a callback function that will be called on click. When the user clicks on a button, the function is called with the number of the clicked button, counted from 0. -1 and -2 are set if the user pushed the Return or Escape keys. The dialog is unreferenced after the user's answer, so *you have to reference the dialog in the callback if you want to keep the dialog alive*.

This class defines various helper functions to build a Dialog.

Note that Dialogs and Menus share the same rendering.

5.19.2 Macro Definition Documentation

5.19.2.1 #define CAIRO_DOCK_IS_DIALOG(*obj*)

Say if an object is a Dialog.

Parameters

obj	the object.

Returns

TRUE if the object is a dialog.

5.19.2.2 #define CAIRO_DIALOG(*pContainer*)

Cast a Container into a Dialog.

Parameters

pContainer	the container.

Returns

the dialog.

5.19.3 Function Documentation

5.19.3.1 CairoDialog∗ gldi_dialog_new (CairoDialogAttr ∗ *pAttribute*)

Create a new dialog.

Parameters

pAttribute	attributes of the dialog.

Returns

the dialog.

5.19.3.2 CairoDialog∗ gldi_dialog_show (const gchar ∗ *cText*, Icon ∗ *pIcon*, GldiContainer ∗ *pContainer*, double *fTimeLength*, const gchar ∗ *cIconPath*, GtkWidget ∗ *pInteractiveWidget*, CairoDockActionOnAnswerFunc *pActionFunc*, gpointer *data*, GFreeFunc *pFreeDataFunc*)

Pop up a dialog with a message, a widget, 2 buttons ok/cancel and an icon, all optionnal.

Parameters

cText	the message to display.
pIcon	the icon that will hold the dialog.
pContainer	the container of the icon.
fTimeLength	the duration of the dialog (in ms), or 0 for an unlimited dialog.
cIconPath	path to an icon to display in the margin.
pInteractive←Widget	a GTK widget; It is destroyed with the dialog. Use 'cairo_dock_steal_interactive_widget_←from_dialog()' before if you want to keep it alive.
pActionFunc	the callback called when the user makes its choice. NULL means there will be no buttons.
data	data passed as a parameter of the callback.
pFreeDataFunc	function used to free the data when the dialog is destroyed, or NULL if unnecessary.

Returns

the newly created dialog.

5.19.3.3 CairoDialog∗ gldi_dialog_show_temporary_with_icon_printf (const gchar ∗ *cText*, Icon ∗ *pIcon*, GldiContainer ∗ *pContainer*, double *fTimeLength*, const gchar ∗ *cIconPath*, ...)

Pop up a dialog with a message, and a limited duration, and an icon in the margin.

Parameters

cText	the message to display.
pIcon	the icon that will hold the dialog.
pContainer	the container of the icon.
fTimeLength	the duration of the dialog (in ms), or 0 for an unlimited dialog.
cIconPath	path to an icon.
...	arguments to insert in the message, in a printf way.

Returns

the newly created dialog.

5.19.3.4 CairoDialog CairoDialog∗ gldi_dialog_show_temporary_with_icon (const gchar ∗ *cText*, Icon ∗ *pIcon*, GldiContainer ∗ *pContainer*, double *fTimeLength*, const gchar ∗ *cIconPath*)

Pop up a dialog with a message, and a limited duration, and an icon in the margin.

Parameters

cText	the message to display.
pIcon	the icon that will hold the dialog.
pContainer	the container of the icon.
fTimeLength	the duration of the dialog (in ms), or 0 for an unlimited dialog.
cIconPath	path to an icon.

Returns

the newly created dialog.

5.19.3.5 CairoDialog∗ gldi_dialog_show_temporary (const gchar ∗ *cText,* Icon ∗ *pIcon,* GldiContainer ∗ *pContainer,* double *fTimeLength*)

Pop up a dialog with a message, and a limited duration, with no icon.

Parameters

cText	the message to display.
pIcon	the icon that will hold the dialog.
pContainer	the container of the icon.
fTimeLength	the duration of the dialog (in ms), or 0 for an unlimited dialog.

Returns

the newly created dialog et visible, avec une reference a 1.

5.19.3.6 CairoDialog∗ gldi_dialog_show_temporary_with_default_icon (const gchar ∗ *cText,* Icon ∗ *pIcon,* GldiContainer ∗ *pContainer,* double *fTimeLength*)

Pop up a dialog with a message, and a limited duration, and a default icon.

Parameters

cText	the format of the message to display.
pIcon	the icon that will hold the dialog.
pContainer	the container of the icon.
fTimeLength	the duration of the dialog (in ms), or 0 for an unlimited dialog.

Returns

the newly created dialog et visible, avec une reference a 1.

5.19.3.7 CairoDialog∗ gldi_dialog_show_with_question (const gchar ∗ *cText,* Icon ∗ *pIcon,* GldiContainer ∗ *pContainer,* const gchar ∗ *cIconPath,* CairoDockActionOnAnswerFunc *pActionFunc,* gpointer *data,* GFreeFunc *pFreeDataFunc*)

Pop up a dialog with a question and 2 buttons ok/cancel. The dialog is unreferenced after the user has answered, so if you want to keep it alive, you have to reference it in the callback.

Parameters

cText	the message to display.
pIcon	the icon that will hold the dialog.
pContainer	the container of the icon.
cIconPath	path to an icon to display in the margin.
pActionFunc	the callback.
data	data passed as a parameter of the callback.
pFreeDataFunc	function used to free the data.

Returns

the newly created dialog et visible, avec une reference a 1.

5.19.3.8 CairoDialog∗ gldi_dialog_show_with_entry (const gchar ∗ *cText,* Icon ∗ *pIcon,* GldiContainer ∗ *pContainer,* const gchar ∗ *cIconPath,* const gchar ∗ *cTextForEntry,* CairoDockActionOnAnswerFunc *pActionFunc,* gpointer *data,* GFreeFunc *pFreeDataFunc*)

Pop up a dialog with a text entry and 2 buttons ok/cancel. The dialog is unreferenced after the user has answered, so if you want to keep it alive, you have to reference it in the callback.

Parameters

cText	the message to display.
pIcon	the icon that will hold the dialog.
pContainer	the container of the icon.
cIconPath	path to an icon to display in the margin.
cTextForEntry	text to display initially in the entry.
pActionFunc	the callback.
data	data passed as a parameter of the callback.
pFreeDataFunc	function used to free the data.

Returns

the newly created dialog.

5.19.3.9 CairoDialog∗ gldi_dialog_show_with_value (const gchar ∗ *cText,* Icon ∗ *pIcon,* GldiContainer ∗ *pContainer,* const gchar ∗ *cIconPath,* double *fValue,* double *fMaxValue,* CairoDockActionOnAnswerFunc *pActionFunc,* gpointer *data,* GFreeFunc *pFreeDataFunc*)

Pop up a dialog with an horizontal scale between 0 and fMaxValue and 2 buttons ok/cancel. The dialog is unreferenced after the user has answered, so if you want to keep it alive, you have to reference it in the callback.

Parameters

cText	the message to display.
pIcon	the icon that will hold the dialog.
pContainer	the container of the icon.
cIconPath	path to an icon to display in the margin.
fValue	initial value of the scale.
fMaxValue	maximum value of the scale.
pActionFunc	the callback.

data	data passed as a parameter of the callback.
pFreeDataFunc	function used to free the data.

Returns

the newly created dialog.

5.19.3.10 CairoDialog∗ gldi_dialog_show_general_message (const gchar ∗ *cMessage,* double *fTimeLength*)

Pop up a dialog, pointing on "the best icon possible". This allows to display a general message.

Parameters

cMessage	the message.
fTimeLength	life time of the dialog, in ms.

Returns

the newly created dialog, visible and with a reference of 1.

5.19.3.11 int gldi_dialog_show_and_wait (const gchar ∗ *cText,* Icon ∗ *pIcon,* GldiContainer ∗ *pContainer,* const gchar ∗ *cIconPath,* GtkWidget ∗ *pInteractiveWidget*)

Pop up a dialog with GTK widget and 2 buttons ok/cancel, and block until the user makes its choice.

Parameters

cText	the message to display.
pIcon	the icon that will hold the dialog.
pContainer	the container of the icon.
cIconPath	path to an icon to display in the margin.
pInteractive↩ Widget	an interactive widget.

Returns

the number of the button that was clicked : 0 or -1 for OK, 1 or -2 for CANCEL, -3 if the dialog has been destroyed before. The dialog is destroyed after the user choosed, but the interactive widget is not destroyed, which allows to retrieve the changes made by the user. Destroy it with 'gtk_widget_destroy' when you're done with it.

5.19.3.12 GtkWidget∗ gldi_dialog_steal_interactive_widget (CairoDialog ∗ *pDialog*)

Detach the interactive widget from a dialog. The widget can then be placed anywhere after that. You have to unref it after you placed it into a container, or to destroy it.

Parameters

pDialog	the desklet with an interactive widget.

Returns

the widget.

5.20 cairo-dock-dialog-manager.h File Reference

Typedefs

- typedef void(∗ CairoDockActionOnAnswerFunc)(int iClickedButton, GtkWidget ∗pInteractiveWidget, gpointer data, CairoDialog ∗pDialog)

  Definition of a generic callback of a dialog, called when the user clicks on a button. Buttons are numbered from 0, -1 means 'Return' and -2 means 'Escape'.

Enumerations

- enum CairoDialogNotifications

  signals

Functions

- void gldi_dialogs_remove_on_icon (Icon ∗icon)
- void gldi_dialog_hide (CairoDialog ∗pDialog)
- void gldi_dialog_unhide (CairoDialog ∗pDialog)
- void gldi_dialog_toggle_visibility (CairoDialog ∗pDialog)

5.20.1 Detailed Description

This class manages the Dialogs, that are useful to bring interaction with the user.

With dialogs, you can pop-up messages, ask for question, etc. Any GTK widget can be embedded inside a dialog, giving you any possible interaction with the user.

The most generic way to build a Dialog is to fill a _CairoDialogAttr and pass it to gldi_dialog_new.

But in most of case, you can just use one of the following convenient functions, that will do the job for you.

- to show a message, you can use gldi_dialog_show_temporary_with_icon
- to ask the user a choice, a value or a text, you can use gldi_dialog_show_with_question, gldi_dialog_show↩_with_value or gldi_dialog_show_with_entry.
- if you want to pop up only 1 dialog at once on a given icon, use gldi_dialogs_remove_on_icon before you pop up your dialog.

5.20.2 Function Documentation

5.20.2.1 void gldi_dialogs_remove_on_icon (Icon ∗ *icon*)

Remove the dialogs attached to an icon.

Parameters

icon	the icon you want to delete all dialogs from.

5.20.2.2 void gldi_dialog_hide (CairoDialog ∗ *pDialog*)

Hide a dialog.

Parameters

pDialog	the dialog.

5.20.2.3 void gldi_dialog_unhide (CairoDialog ∗ *pDialog*)

Show a dialog and give it focus.

Parameters

pDialog	the dialog.

5.20.2.4 void gldi_dialog_toggle_visibility (CairoDialog ∗ *pDialog*)

Toggle the visibility of a dialog.

Parameters

pDialog	the dialog.

5.21 cairo-dock-dock-facility.h File Reference

Macros

- #define cairo_dock_get_available_docks_for_icon(pIcon)

Functions

- void cairo_dock_update_dock_size (CairoDock ∗pDock)
- Icon ∗ cairo_dock_calculate_dock_icons (CairoDock ∗pDock)
- void cairo_dock_show_subdock (Icon ∗pPointedIcon, CairoDock ∗pParentDock)
- GList ∗ cairo_dock_get_available_docks (CairoDock ∗pParentDock, CairoDock ∗pSubDock)
- void cairo_dock_calculate_icons_positions_at_rest_linear (GList ∗pIconList, double fFlatDockWidth)
- Icon ∗ cairo_dock_apply_wave_effect_linear (CairoDock ∗pDock)
- double cairo_dock_get_current_dock_width_linear (CairoDock ∗pDock)
- void cairo_dock_check_if_mouse_inside_linear (CairoDock ∗pDock)
- void cairo_dock_check_can_drop_linear (CairoDock ∗pDock)
- GList ∗ cairo_dock_get_first_drawn_element_linear (GList ∗icons)

5.21.1 Detailed Description

This class contains functions to manipulate docks. Some functions are dedicated to linear docks, that is to say when the icon's position can be defined by 1 coordinate inside a non looped interval; it doesn't mean they have to be drawn on a straight line though, see the Curve view.

5.21.2 Macro Definition Documentation

5.21.2.1 #define cairo_dock_get_available_docks_for_icon(*pIcon*)

Get a list of available docks where an user icon can be placed. Its current parent dock is excluded, as well as its sub-dock (if any) and its children.

Parameters

pIcon	the icon

Returns

a list of CairoDock∗

5.21.3 Function Documentation

5.21.3.1 void cairo_dock_update_dock_size (CairoDock ∗ *pDock*)

Compute the maximum size of a dock, and resize it if necessary. It takes into account the size limit, and moves the dock so that it stays centered. Also updates the dock's background if necessary, and re-place the appli thumbnails.

Parameters

pDock	the dock.

5.21.3.2 Icon∗ cairo_dock_calculate_dock_icons (CairoDock ∗ *pDock*)

Calculate the position of all icons inside a dock, and triggers the enter/leave events according to the position of the mouse.

Parameters

pDock	the dock.

Returns

the pointed icon, or NULL if none is pointed.

5.21.3.3 void cairo_dock_show_subdock (Icon ∗ *pPointedIcon,* CairoDock ∗ *pParentDock*)

Pop up a sub-dock.

Parameters

pPointedIcon	icon pointing on the sub-dock.
pParentDock	dock containing the icon.

5.21.3.4 GList∗ cairo_dock_get_available_docks (CairoDock ∗ *pParentDock,* CairoDock ∗ *pSubDock*)

Get a list of available docks.

Parameters

pParentDock	excluding this dock if not NULL
pSubDock	excluding this dock and its children if not NULL

Returns

a list of CairoDock∗

5.21.3.5 void cairo_dock_calculate_icons_positions_at_rest_linear (GList ∗ *pIconList,* double *fFlatDockWidth*)

Calculate the position at rest (when the mouse is outside of the dock and its size is normal) of the icons of a linear dock.

Parameters

pIconList	a list of icons.
fFlatDockWidth	width of all the icons placed next to each other.

5.21.3.6 Icon∗ cairo_dock_apply_wave_effect_linear (CairoDock ∗ *pDock*)

Apply a wave effect on the icons of a linear dock. It is the famous zoom when the mouse hovers an icon.

Parameters

pDock	a linear dock.

Returns

the pointed icon, or NULL if none is pointed.

5.21.3.7 double cairo_dock_get_current_dock_width_linear (CairoDock ∗ *pDock*)

Get the current width of all the icons of a linear dock. It doesn't take into account any decoration or frame, only the space occupied by the icons.

Parameters

pDock	a linear dock.

Returns

the dock's width.

5.21.3.8 void cairo_dock_check_if_mouse_inside_linear (CairoDock ∗ *pDock*)

Check the position of the mouse inside a linear dock. It can be inside, on the edge, or outside. Update the 'iMouse↩
PositionType' field.

Parameters

pDock	a linear dock.

5.21.3.9 void cairo_dock_check_can_drop_linear (CairoDock ∗ *pDock*)

Check if one can drop inside a linear dock. Drop is allowed between 2 icons of the launchers group, if the user is dragging something over the dock. Update the 'bCanDrop' field.

Parameters

pDock	a linear dock.

5.21.3.10 GList∗ cairo_dock_get_first_drawn_element_linear (GList ∗ *icons*)

Get the first icon to be drawn inside a linear dock, so that if you draw from left to right, the pointed icon will be drawn at last.

Parameters

icons	a list of icons of a linear dock.

Returns

the element of the list that contains the first icon to draw.

5.22 cairo-dock-dock-factory.h File Reference

Data Structures

- struct _CairoDockRenderer

  Dock's renderer, also known as 'view'.
- struct _CairoDock

  Definition of a Dock, which derives from a Container.

Macros

- #define GLDI_OBJECT_IS_DOCK(obj)
- #define CAIRO_DOCK(p)

Functions

- CairoDock ∗ gldi_dock_new (const gchar ∗cDockName)
- CairoDock ∗ gldi_subdock_new (const gchar ∗cDockName, const gchar ∗cRendererName, CairoDock ∗p↩ParentDock, GList ∗pIconList)
- void cairo_dock_remove_icons_from_dock (CairoDock ∗pDock, CairoDock ∗pReceivingDock)

5.22.1 Detailed Description

This class defines the Docks, and gives the way to create, destroy, and fill them.

A dock is a container that holds a set of icons and a renderer (also known as view).

It has the ability to be placed anywhere on the screen edges and to resize itself automatically to fit the screen's size.

It supports internal dragging of its icons with the mouse, and dragging of itself with alt+mouse.

A dock can be either a main-dock (not linked to any icon) or a sub-dock (linked to an icon of another dock), and there can be as many docks of each sort as you want.

5.22.2 Macro Definition Documentation

5.22.2.1 #define GLDI_OBJECT_IS_DOCK(*obj*)

Say if an object is a Dock.

Parameters

obj	the object.

Returns

TRUE if the object is a Dock.

5.22.2.2 #define CAIRO_DOCK(*p*)

Cast a Container into a Dock.

Parameters

p	the container to consider as a dock.

Returns

the dock.

5.22.3 Function Documentation

5.22.3.1 CairoDock∗ gldi_dock_new (const gchar ∗ *cDockName*)

Create a new root dock.

Parameters

cDockName	the name that identifies the dock

Returns

the new dock.

5.22.3.2 CairoDock∗ gldi_subdock_new (const gchar ∗ *cDockName,* const gchar ∗ *cRendererName,* CairoDock ∗ *pParentDock,* GList ∗ *pIconList*)

Create a new dock of type "sub-dock", and load a given list of icons inside. The list then belongs to the dock, so it must not be freeed after that. The buffers of each icon are loaded, so they just need to have an image filename and a name.

Parameters

cDockName	the name that identifies the dock.
cRendererName	name of a renderer. If NULL, the default renderer will be applied.
pParentDock	the parent dock.
pIconList	a list of icons that will be loaded and inserted into the new dock (optional).

Returns

the new dock.

5.22.3.3 void cairo_dock_remove_icons_from_dock (CairoDock ∗ *pDock,* CairoDock ∗ *pReceivingDock*)

Remove all icons from a dock (and its sub-docks). If the receiving dock is NULL, the icons are destroyed and removed from the current theme itself.

Parameters

pDock	a dock.
pReceivingDock	the dock that will receive the icons, or NULL to destroy and remove the icons.

5.23 cairo-dock-dock-manager.h File Reference

Macros

- #define gldi_dock_get_name(pDock)

Enumerations

- enum GldiIconSize

    TODO: harmonize the values with the simple config -> make some public functions...
- enum CairoDocksNotifications {
  NOTIFICATION_ENTER_DOCK,
  NOTIFICATION_LEAVE_DOCK,
  NOTIFICATION_INSERT_ICON,
  NOTIFICATION_REMOVE_ICON,
  NOTIFICATION_ICON_MOVED }

    signals

Functions

- gchar ∗ gldi_dock_get_readable_name (CairoDock ∗pDock)
- CairoDock ∗ gldi_dock_get (const gchar ∗cDockName)
- Icon ∗ cairo_dock_search_icon_pointing_on_dock (CairoDock ∗pDock, CairoDock ∗∗pParentDock)
- void gldi_dock_rename (CairoDock ∗pDock, const gchar ∗cNewName)
- void gldi_docks_foreach (GHFunc pFunction, gpointer pUserData)
- void gldi_docks_foreach_root (GFunc pFunction, gpointer pUserData)
- void gldi_icons_foreach_in_docks (GldiIconFunc pFunction, gpointer pUserData)
- void cairo_dock_reload_buffers_in_all_docks (gboolean bUpdateIconSize)
- void gldi_dock_add_conf_file_for_name (const gchar ∗cDockName)
- gchar ∗ gldi_dock_add_conf_file (void)
- void gldi_docks_redraw_all_root (void)
- void gldi_dock_set_visibility (CairoDock ∗pDock, CairoDockVisibility iVisibility)

5.23.1 Detailed Description

This class manages all the Docks. Each Dock has a name that is unique. A Dock can be a sub-dock or a root-dock, whether there exists an icon that points on it or not, but there is no fundamental difference between both.

5.23.2 Macro Definition Documentation

5.23.2.1 #define gldi_dock_get_name(*pDock*)

Get the name of a Dock.

Parameters

pDock	the dock.

Returns

the name of the dock, that identifies it.

5.23.3 Enumeration Type Documentation

5.23.3.1 enum CairoDocksNotifications

signals

Enumerator

NOTIFICATION_ENTER_DOCK notification called when the mouse enters a dock.

NOTIFICATION_LEAVE_DOCK notification called when the mouse leave a dock.

NOTIFICATION_INSERT_ICON notification called when an icon has just been inserted into a dock. data : {Icon, CairoDock}

NOTIFICATION_REMOVE_ICON notification called when an icon is going to be removed from a dock. data : {Icon, CairoDock}

NOTIFICATION_ICON_MOVED notification called when an icon is moved inside a dock. data : {Icon, Cairo↩ Dock}

5.23.4 Function Documentation

5.23.4.1 gchar∗ gldi_dock_get_readable_name (CairoDock ∗ *pDock*)

Get a readable name for a main Dock, suitable for display (like "Bottom dock"). Sub-Docks names are defined by the user, so you can just use gldi_dock_get_name for them.

Parameters

pDock	the dock.

Returns

the readable name of the dock, or NULL if not found. Free it when you're done.

5.23.4.2 CairoDock∗ gldi_dock_get (const gchar ∗ *cDockName*)

Get a Dock from a given name.

Parameters

cDockName	the name of the dock.

Returns

the dock that has been registerd under this name, or NULL if none exists.

5.23.4.3 Icon∗ cairo_dock_search_icon_pointing_on_dock (CairoDock ∗ *pDock,* CairoDock ∗∗ *pParentDock*)

Search an icon pointing on a dock. If several icons point on it, the first one will be returned.

Parameters

pDock	the dock.
pParentDock	if not NULL, this will be filled with the dock containing the icon.

Returns

the icon pointing on the dock.

5.23.4.4 void gldi_dock_rename (CairoDock ∗ *pDock,* const gchar ∗ *cNewName*)

Rename a dock. Update the container's name of all of its icons.

Parameters

pDock	the dock (optional).
cNewName	the new name.

5.23.4.5 void gldi_docks_foreach (GHFunc *pFunction*, gpointer *pUserData*)

Execute an action on all docks.

Parameters

pFunction	the action.
pUserData	data passed to the callback.

5.23.4.6 void gldi_docks_foreach_root (GFunc *pFunction*, gpointer *pUserData*)

Execute an action on all main docks.

Parameters

pFunction	the action.
pUserData	data passed to the callback.

5.23.4.7 void gldi_icons_foreach_in_docks (GldiIconFunc *pFunction*, gpointer *pUserData*)

Execute an action on all icons being inside a dock.

Parameters

pFunction	the action.
pUserData	data passed to the callback.

5.23.4.8 void cairo_dock_reload_buffers_in_all_docks (gboolean *bUpdateIconSize*)

(Re)load all buffers of all icons in all docks.

Parameters

bUpdateIcon←Size	TRUE to recalculate the icons and docks size.

5.23.4.9 void gldi_dock_add_conf_file_for_name (const gchar * *cDockName*)

Add a config file for a root dock. Does not create the dock (use gldi_dock_new for that). If the config file already exists, it is overwritten (use gldi_dock_get to check if the name is already used).

Parameters

cDockName	name of the dock.

5.23.4.10 gchar* gldi_dock_add_conf_file (void)

Add a config file for a new root dock. Does not create the dock (use gldi_dock_new for that).

Returns

the unique name for the new dock, to be passed to gldi_dock_new.

5.23.4.11 void gldi_docks_redraw_all_root (void)

Redraw every root docks.

5.23.4.12 void gldi_dock_set_visibility (CairoDock ∗ *pDock,* CairoDockVisibility *iVisibility*)

Set the visibility of a root dock. Perform all the necessary actions.

Parameters

pDock	a root dock.
iVisibility	its new visibility.

5.24 cairo-dock-dock-visibility.h File Reference

Functions

- GldiWindowActor ∗ gldi_dock_search_overlapping_window (CairoDock ∗pDock)

5.24.1 Detailed Description

This class manages the visibility of Docks.

5.24.2 Function Documentation

5.24.2.1 GldiWindowActor∗ gldi_dock_search_overlapping_window (CairoDock ∗ *pDock*)

Get the application whose window overlaps a dock, or NULL if none.

Parameters

pDock	the dock to test.

Returns

the window actor, or NULL if none has been found.

5.25 cairo-dock-draw-opengl.h File Reference

Macros

- #define cairo_dock_create_texture_from_image(cImagePath)
- #define _cairo_dock_delete_texture(iTexture)
- #define _cairo_dock_enable_texture(...)
- #define _cairo_dock_disable_texture(...)
- #define _cairo_dock_set_alpha(fAlpha)
- #define _cairo_dock_set_blend_source(...)
- #define _cairo_dock_set_blend_alpha(...)

- #define _cairo_dock_set_blend_over(...)
- #define _cairo_dock_set_blend_pbuffer(...)
- #define _cairo_dock_apply_texture_at_size(iTexture, w, h)
- #define _cairo_dock_apply_texture(iTexture)
- #define _cairo_dock_apply_texture_at_size_with_alpha(iTexture, w, h, fAlpha)

Functions

- void cairo_dock_render_one_icon_opengl (Icon ∗icon, CairoDock ∗pDock, double fDockMagnitude, gboolean bUseText)
- GLuint cairo_dock_create_texture_from_surface (cairo_surface_t ∗pImageSurface)
- GLuint cairo_dock_create_texture_from_raw_data (const guchar ∗pTextureRaw, int iWidth, int iHeight)
- GLuint cairo_dock_create_texture_from_image_full (const gchar ∗cImagePath, double ∗fImageWidth, double ∗fImageHeight)
- void cairo_dock_update_icon_texture (Icon ∗pIcon)

5.25.1 Detailed Description

This class provides some useful functions to draw with OpenGL.

5.25.2 Macro Definition Documentation

5.25.2.1 #define cairo_dock_create_texture_from_image(*cImagePath*)

Load an image on the dock into an OpenGL texture. The texture will have the same size as the image.

Parameters

cImagePath	path to an image.

Returns

the newly allocated texture, to be destroyed with _cairo_dock_delete_texture.

5.25.2.2 #define _cairo_dock_delete_texture(*iTexture*)

Delete an OpenGL texture from the Graphic Card.

Parameters

iTexture	variable containing the ID of a texture.

5.25.2.3 #define _cairo_dock_enable_texture(*...*)

Enable texture drawing.

5.25.2.4 #define _cairo_dock_disable_texture(*...*)

Disable texture drawing.

5.25.2.5 #define _cairo_dock_set_alpha(*fAlpha*)

Set the alpha channel to a current value, other channels are set to 1.

Parameters

fAlpha	alpha

5.25.2.6 #define _cairo_dock_set_blend_source(...)

Set the color blending to overwrite.

5.25.2.7 #define _cairo_dock_set_blend_alpha(...)

Set the color blending to mix, for premultiplied texture.

5.25.2.8 #define _cairo_dock_set_blend_over(...)

Set the color blending to mix.

5.25.2.9 #define _cairo_dock_set_blend_pbuffer(...)

Set the color blending to mix on a pbuffer.

5.25.2.10 #define _cairo_dock_apply_texture_at_size(*iTexture*, *w*, *h*)

Draw a texture centered on the current point, at a given size.

Parameters

iTexture	the texture
w	width
h	height

5.25.2.11 #define _cairo_dock_apply_texture(*iTexture*)

Apply a texture centered on the current point and at the given scale.

Parameters

iTexture	the texture

5.25.2.12 #define _cairo_dock_apply_texture_at_size_with_alpha(*iTexture*, *w*, *h*, *fAlpha*)

Draw a texture centered on the current point, at a given size, and with a given transparency.

Parameters

iTexture	the texture
w	width
h	height
fAlpha	the transparency, between 0 and 1.

5.25.3 Function Documentation

5.25.3.1 void cairo_dock_render_one_icon_opengl (**Icon** ∗ *icon,* **CairoDock** ∗ *pDock,* double *fDockMagnitude,* gboolean *bUseText*)

Draw an icon, according to its current parameters : position, transparency, reflect, rotation, stretching. Also draws its indicators, label, and quick-info. It generates a CAIRO_DOCK_RENDER_ICON notification.

Parameters

icon	the icon to draw.
pDock	the dock containing the icon.
fDockMagnitude	current magnitude of the dock.
bUseText	TRUE to draw the labels.

5.25.3.2 GLuint cairo_dock_create_texture_from_surface (cairo_surface_t ∗ *pImageSurface*)

Load a cairo surface into an OpenGL texture. The surface can be destroyed after that if you don't need it. The texture will have the same size as the surface.

Parameters

pImageSurface	the surface, created with one of the 'cairo_dock_create_surface_xxx' functions.

Returns

the newly allocated texture, to be destroyed with _cairo_dock_delete_texture.

5.25.3.3 GLuint cairo_dock_create_texture_from_raw_data (const guchar ∗ *pTextureRaw,* int *iWidth,* int *iHeight*)

Load a pixels buffer representing an image into an OpenGL texture.

Parameters

pTextureRaw	a buffer of pixels.
iWidth	width of the image.
iHeight	height of the image.

Returns

the newly allocated texture, to be destroyed with _cairo_dock_delete_texture.

5.25.3.4 GLuint cairo_dock_create_texture_from_image_full (const gchar ∗ *cImagePath,* double ∗ *fImageWidth,* double ∗ *fImageHeight*)

Load an image on the dock into an OpenGL texture. The texture will have the same size as the image. The size is given as an output, if you need it for some reason.

Parameters

cImagePath	path to an image.
fImageWidth	pointer that will be filled with the width of the image.
fImageHeight	pointer that will be filled with the height of the image.

Returns

the newly allocated texture, to be destroyed with _cairo_dock_delete_texture.

5.25.3.5 void cairo_dock_update_icon_texture (Icon ∗ *pIcon*)

Update the icon's texture with its current cairo surface. This allows you to draw an icon with libcairo, and just copy the result to the OpenGL texture to be able to draw the icon in OpenGL too.

Parameters

pIcon	the icon.

5.26 cairo-dock-draw.h File Reference

Macros

- #define cairo_dock_erase_cairo_context(pCairoContext)

Functions

- cairo_t ∗ cairo_dock_create_drawing_context_generic (GldiContainer ∗pContainer)
  CONTEXT ///.
- cairo_t ∗ cairo_dock_create_drawing_context_on_container (GldiContainer ∗pContainer)
- cairo_t ∗ cairo_dock_create_drawing_context_on_area (GldiContainer ∗pContainer, GdkRectangle ∗pArea, double ∗fBgColor)
- void cairo_dock_draw_rounded_rectangle (cairo_t ∗pCairoContext, double fRadius, double fLineWidth, double fFrameWidth, double fFrameHeight)
- void cairo_dock_draw_icon_cairo (Icon ∗icon, CairoDock ∗pDock, cairo_t ∗pCairoContext)
- void cairo_dock_render_one_icon (Icon ∗icon, CairoDock ∗pDock, cairo_t ∗pCairoContext, double fDock←Magnitude, gboolean bUseText)
- void cairo_dock_draw_string (cairo_t ∗pCairoContext, CairoDock ∗pDock, double fStringLineWidth, gboolean bIsLoop, gboolean bForceConstantSeparator)

5.26.1 Detailed Description

This class provides some useful functions to draw with libcairo.

5.26.2 Macro Definition Documentation

5.26.2.1 #define cairo_dock_erase_cairo_context(*pCairoContext*)

Erase a drawing context, making it fully transparent. You don't need to erase a newly created context.

Parameters

pCairoContext	a drawing context.

5.26.3 Function Documentation

5.26.3.1 cairo_t∗ cairo_dock_create_drawing_context_generic (GldiContainer ∗ *pContainer*)

CONTEXT ///.

Create a generic drawing context, to be used as a source context (for instance, for creating a surface).

Parameters

pContainer	a container.

Returns

the context on which to draw. Is never NULL, test it with cairo_status() before use it, and destroy it with cairo_destroy() when you're done with it.

5.26.3.2 cairo_t∗ cairo_dock_create_drawing_context_on_container (GldiContainer ∗ *pContainer*)

Create a drawing context to draw on a container. It handles fake transparency.

Parameters

pContainer	the container on which you want to draw.

Returns

the newly allocated context, to be destroyed with 'cairo_destroy'.

5.26.3.3 cairo_t∗ cairo_dock_create_drawing_context_on_area (GldiContainer ∗ *pContainer,* GdkRectangle ∗ *pArea,* double ∗ *fBgColor*)

Create a drawing context to draw on a part of a container. It handles fake transparency.

Parameters

pContainer	the container on which you want to draw
pArea	part of the container to draw.
fBgColor	background color (rgba) to fill the area with, or NULL to let it transparent.

Returns

the newly allocated context, with a clip corresponding to the area, to be destroyed with 'cairo_destroy'.

5.26.3.4 void cairo_dock_draw_rounded_rectangle (cairo_t ∗ *pCairoContext,* double *fRadius,* double *fLineWidth,* double *fFrameWidth,* double *fFrameHeight*)

Compute the path of a rectangle with rounded corners. It doesn't stroke it, use cairo_stroke or cairo_fill to draw the line or the inside.

Parameters

pCairoContext	a drawing context; the current matrix is not altered, but the current path is.
fRadius	radius if the corners.
fLineWidth	width of the line.
fFrameWidth	width of the rectangle, without the corners.
fFrameHeight	height of the rectangle, including the corners.

5.26.3.5 void cairo_dock_draw_icon_cairo (Icon ∗ *icon,* CairoDock ∗ *pDock,* cairo_t ∗ *pCairoContext*)

Draw an icon and its reflect on a dock. Only draw the icon's image and reflect, and nothing else.

Parameters

icon	the icon to draw.
pDock	the dock containing the icon.
pCairoContext	a context on the dock, not altered by the function.

5.26.3.6 void cairo_dock_render_one_icon (Icon ∗ *icon,* CairoDock ∗ *pDock,* cairo_t ∗ *pCairoContext,* double *fDockMagnitude,* gboolean *bUseText*)

Draw an icon, according to its current parameters : position, transparency, reflect, rotation, stretching. Also draws its indicators, label, and quick-info. It generates a CAIRO_DOCK_RENDER_ICON notification.

Parameters

icon	the icon to draw.
pDock	the dock containing the icon.
pCairoContext	a context on the dock, it is altered by the function.
fDockMagnitude	current magnitude of the dock.
bUseText	TRUE to draw the labels.

5.26.3.7 void cairo_dock_draw_string (cairo_t ∗ *pCairoContext,* CairoDock ∗ *pDock,* double *fStringLineWidth,* gboolean *bIsLoop,* gboolean *bForceConstantSeparator*)

Draw a string linking the center of all the icons of a dock.

Parameters

pCairoContext	a context on the dock, not altered by the function.
pDock	the dock.
fStringLineWidth	width of the line.
bIsLoop	TRUE to loop (link the last icon to the first one).
bForce↩ Constant↩ Separator	TRUE to consider separators having a constant size.

5.27 cairo-dock-file-manager.h File Reference

Data Structures

- struct _CairoDockDesktopEnvBackend

  Definition of the Desktop Environment backend.

Enumerations

- enum CairoDockDesktopEnv

  Type of available Desktop Environments.
- enum CairoDockFMEventType

  Type of events that can occur to a file.
- enum CairoDockFMSortType

  Type of sorting available on files.

Functions

- void cairo_dock_fm_register_vfs_backend (CairoDockDesktopEnvBackend ∗pVFSBackend)
- GList ∗ cairo_dock_fm_list_directory (const gchar ∗cURI, CairoDockFMSortType g_fm_iSortType, int iNew↩ IconsType, gboolean bListHiddenFiles, int iNbMaxFiles, gchar ∗∗cFullURI)
- gsize cairo_dock_fm_measure_diretory (const gchar ∗cBaseURI, gint iCountType, gboolean bRecursive, gint ∗pCancel)
- gboolean cairo_dock_fm_get_file_info (const gchar ∗cBaseURI, gchar ∗∗cName, gchar ∗∗cURI, gchar ∗∗c↩ IconName, gboolean ∗bIsDirectory, int ∗iVolumeID, double ∗fOrder, CairoDockFMSortType iSortType)
- gboolean cairo_dock_fm_get_file_properties (const gchar ∗cURI, guint64 ∗iSize, time_t ∗iLastModification↩ Time, gchar ∗∗cMimeType, int ∗iUID, int ∗iGID, int ∗iPermissionsMask)
- gboolean cairo_dock_fm_launch_uri (const gchar ∗cURI)
- gboolean cairo_dock_fm_add_monitor_full (const gchar ∗cURI, gboolean bDirectory, const gchar ∗c↩ MountedURI, CairoDockFMMonitorCallback pCallback, gpointer data)

- gboolean cairo_dock_fm_remove_monitor_full (const gchar ∗cURI, gboolean bDirectory, const gchar ∗c↩ MountedURI)
- gboolean cairo_dock_fm_mount_full (const gchar ∗cURI, int iVolumeID, CairoDockFMMountCallback p↩ Callback, gpointer user_data)
- gboolean cairo_dock_fm_unmount_full (const gchar ∗cURI, int iVolumeID, CairoDockFMMountCallback p↩ Callback, gpointer user_data)
- gchar ∗ cairo_dock_fm_is_mounted (const gchar ∗cURI, gboolean ∗bIsMounted)
- gboolean cairo_dock_fm_can_eject (const gchar ∗cURI)
- gboolean cairo_dock_fm_eject_drive (const gchar ∗cURI)
- gboolean cairo_dock_fm_delete_file (const gchar ∗cURI, gboolean bNoTrash)
- gboolean cairo_dock_fm_rename_file (const gchar ∗cOldURI, const gchar ∗cNewName)
- gboolean cairo_dock_fm_move_file (const gchar ∗cURI, const gchar ∗cDirectoryURI)
- gboolean cairo_dock_fm_create_file (const gchar ∗cURI, gboolean bDirectory)
- GList ∗ cairo_dock_fm_list_apps_for_file (const gchar ∗cURI)
- gboolean cairo_dock_fm_empty_trash (void)
- gchar ∗ cairo_dock_fm_get_trash_path (const gchar ∗cNearURI, gchar ∗∗cFileInfoPath)
- gchar ∗ cairo_dock_fm_get_desktop_path (void)
- gboolean cairo_dock_fm_logout (void)
- gboolean cairo_dock_fm_shutdown (void)
- gboolean cairo_dock_fm_reboot (void)
- gboolean cairo_dock_fm_lock_screen (void)
- gboolean cairo_dock_fm_setup_time (void)
- gboolean cairo_dock_fm_show_system_monitor (void)
- Icon ∗ cairo_dock_fm_create_icon_from_URI (const gchar ∗cURI, GldiContainer ∗pContainer, CairoDockF↩ MSortType iFileSortType)
- int cairo_dock_get_file_size (const gchar ∗cFilePath)
- int cairo_dock_fm_get_pid (const gchar ∗cProcessName)
- gboolean cairo_dock_fm_monitor_pid (const gchar ∗cProcessName, gboolean bCheckSameProcess, G↩ SourceFunc pCallback, gboolean bAlwaysLaunch, gpointer pUserData)

5.27.1 Detailed Description

This class manages the integration into the desktop environment, which includes :

- the VFS (Virtual File System)
- the various desktop-related tools.

5.27.2 Function Documentation

5.27.2.1 void cairo_dock_fm_register_vfs_backend (CairoDockDesktopEnvBackend ∗ *pVFSBackend*)

Register a environment backend, overwriting any previous backend.

5.27.2.2 GList∗ cairo_dock_fm_list_directory (const gchar ∗ *cURI,* CairoDockFMSortType *g_fm_iSortType,* int *iNewIconsType,* gboolean *bListHiddenFiles,* int *iNbMaxFiles,* gchar ∗∗ *cFullURI*)

List the content of a directory and turn it into a list of icons.

5.27.2.3 gsize cairo_dock_fm_measure_diretory (const gchar ∗ *cBaseURI,* gint *iCountType,* gboolean *bRecursive,* gint ∗ *pCancel*)

Measure a directory (number of files or total size).

5.27.2.4 gboolean cairo_dock_fm_get_file_info (const gchar ∗ *cBaseURI,* gchar ∗∗ *cName,* gchar ∗∗ *cURI,* gchar ∗∗ *cIconName,* gboolean ∗ *bIsDirectory,* int ∗ *iVolumeID,* double ∗ *fOrder,* CairoDockFMSortType *iSortType*)

Get the main info to represent a file.

5.27.2.5 gboolean cairo_dock_fm_get_file_properties (const gchar ∗ *cURI,* guint64 ∗ *iSize,* time_t ∗ *iLastModificationTime,* gchar ∗∗ *cMimeType,* int ∗ *iUID,* int ∗ *iGID,* int ∗ *iPermissionsMask*)

Get some properties about a file.

5.27.2.6 gboolean cairo_dock_fm_launch_uri (const gchar ∗ *cURI*)

Open a file with the default application.

5.27.2.7 gboolean cairo_dock_fm_add_monitor_full (const gchar ∗ *cURI,* gboolean *bDirectory,* const gchar ∗ *cMountedURI,* CairoDockFMMonitorCallback *pCallback,* gpointer *data*)

Add a monitor on an URI. It will be called each time a modification occurs on the file.

5.27.2.8 gboolean cairo_dock_fm_remove_monitor_full (const gchar ∗ *cURI,* gboolean *bDirectory,* const gchar ∗ *cMountedURI*)

Remove a monitor on an URI.

5.27.2.9 gboolean cairo_dock_fm_mount_full (const gchar ∗ *cURI,* int *iVolumeID,* CairoDockFMMountCallback *pCallback,* gpointer *user_data*)

Mount a point.

5.27.2.10 gboolean cairo_dock_fm_unmount_full (const gchar ∗ *cURI,* int *iVolumeID,* CairoDockFMMountCallback *pCallback,* gpointer *user_data*)

Unmount a point.

5.27.2.11 gchar∗ cairo_dock_fm_is_mounted (const gchar ∗ *cURI,* gboolean ∗ *bIsMounted*)

Say if a point is currently mounted.

5.27.2.12 gboolean cairo_dock_fm_can_eject (const gchar ∗ *cURI*)

Say if a point can be ejected (like a CD player).

5.27.2.13 gboolean cairo_dock_fm_eject_drive (const gchar ∗ *cURI*)

Eject a drive, like a CD player.

5.27.2.14 gboolean cairo_dock_fm_delete_file (const gchar ∗ *cURI,* gboolean *bNoTrash*)

Delete a file.

5.27.2.15 gboolean cairo_dock_fm_rename_file (const gchar ∗ *cOldURI*, const gchar ∗ *cNewName*)

Rename a file.

5.27.2.16 gboolean cairo_dock_fm_move_file (const gchar ∗ *cURI*, const gchar ∗ *cDirectoryURI*)

Move a file.

5.27.2.17 gboolean cairo_dock_fm_create_file (const gchar ∗ *cURI*, gboolean *bDirectory*)

Create a new file.

5.27.2.18 GList∗ cairo_dock_fm_list_apps_for_file (const gchar ∗ *cURI*)

Get the list of applications that can open a given file. Returns a list of strings arrays : {name, command, icon}.

5.27.2.19 gboolean cairo_dock_fm_empty_trash (void)

Empty the Trash.

5.27.2.20 gchar∗ cairo_dock_fm_get_trash_path (const gchar ∗ *cNearURI*, gchar ∗∗ *cFileInfoPath*)

Get the path to the Trash.

5.27.2.21 gchar∗ cairo_dock_fm_get_desktop_path (void)

Get the path to the Desktop.

5.27.2.22 gboolean cairo_dock_fm_logout (void)

Raise the logout panel.

5.27.2.23 gboolean cairo_dock_fm_shutdown (void)

Raise the shutdown panel.

5.27.2.24 gboolean cairo_dock_fm_reboot (void)

Raise the reboot panel.

5.27.2.25 gboolean cairo_dock_fm_lock_screen (void)

Lock the screen.

5.27.2.26 gboolean cairo_dock_fm_setup_time (void)

Raise the panel to configure the time.

5.27.2.27 gboolean cairo_dock_fm_show_system_monitor (void)

Raise the default system monitor.

5.27.2.28 Icon∗ cairo_dock_fm_create_icon_from_URI (const gchar ∗ *cURI*, GldiContainer ∗ *pContainer*, CairoDockFMSortType *iFileSortType*)

Create an Icon representing a given URI.

5.27.2.29 int cairo_dock_get_file_size (const gchar ∗ *cFilePath*)

Get the size of a local file.

Parameters

cFilePath	path of a file on the hard disk.

Returns

the size of the file, or 0 if it doesn't exist.

5.27.2.30 int cairo_dock_fm_get_pid (const gchar ∗ *cProcessName*)

Get process ID given its name

Parameters

cProcessName	name of the process

Returns

the PID if it exists or -1

5.27.2.31 gboolean cairo_dock_fm_monitor_pid (const gchar ∗ *cProcessName*, gboolean *bCheckSameProcess*, GSourceFunc *pCallback*, gboolean *bAlwaysLaunch*, gpointer *pUserData*)

Monitor a process. Call a function when the process is no longer running

Parameters

cProcessName	name(es) of the process(es)
bCheckSame←Process	TRUE to check if first match is running. FALSE to check every time if this process name is running even if it's not the same PID.
pCallback	function to call when the process is no longer running
bAlwaysLaunch	TRUE to launch the callback function even if the process is not running or if there is an error
pUserData	data to pass to pCallback

Returns

FALSE if the process is not running or if there is an error

5.28 cairo-dock-gauge.h File Reference

Typedefs

- typedef struct _CairoGaugeAttribute CairoGaugeAttribute

Attributes of a Gauge.

5.28.1 Detailed Description

This class defines the Gauge, which derives from the DataRenderer. All you need to know is the attributes that define a Gauge, the API to use is the common API for DataRenderer, defined in cairo-dock-data-renderer.h.

5.29 cairo-dock-gnome-shell-integration.h File Reference

5.29.1 Detailed Description

This class implements the integration of Gnome-Shell inside Cairo-Dock.

5.30 cairo-dock-graph.h File Reference

Data Structures

- struct _CairoGraphAttribute

  Attributes of a Graph.

Enumerations

- enum CairoDockTypeGraph {
  CAIRO_DOCK_GRAPH_LINE,
  CAIRO_DOCK_GRAPH_PLAIN,
  CAIRO_DOCK_GRAPH_BAR,
  CAIRO_DOCK_GRAPH_CIRCLE,
  CAIRO_DOCK_GRAPH_CIRCLE_PLAIN }

  Types of graph.

5.30.1 Detailed Description

This class defines the Graph, which derives from the DataRenderer. All you need to know is the attributes that define a Graph, the API to use is the common API for DataRenderer, defined in cairo-dock-data-renderer.h.

5.30.2 Enumeration Type Documentation

5.30.2.1 enum CairoDockTypeGraph

Types of graph.

Enumerator

CAIRO_DOCK_GRAPH_LINE a continuous line.

CAIRO_DOCK_GRAPH_PLAIN a continuous plain graph.

CAIRO_DOCK_GRAPH_BAR a histogram.

CAIRO_DOCK_GRAPH_CIRCLE a circle.

CAIRO_DOCK_GRAPH_CIRCLE_PLAIN a plain circle.

5.31 cairo-dock-gui-factory.h File Reference

Data Structures

- struct _CairoDockGroupKeyWidget

  Definition of a widget corresponding to a given (group;key) pair.

Enumerations

- enum CairoDockGUIWidgetType {
  CAIRO_DOCK_WIDGET_CHECK_BUTTON,
  CAIRO_DOCK_WIDGET_CHECK_CONTROL_BUTTON,
  CAIRO_DOCK_WIDGET_SPIN_INTEGER,
  CAIRO_DOCK_WIDGET_HSCALE_INTEGER,
  CAIRO_DOCK_WIDGET_SIZE_INTEGER,
  CAIRO_DOCK_WIDGET_SPIN_DOUBLE,
  CAIRO_DOCK_WIDGET_COLOR_SELECTOR_RGB,
  CAIRO_DOCK_WIDGET_COLOR_SELECTOR_RGBA,
  CAIRO_DOCK_WIDGET_HSCALE_DOUBLE,
  CAIRO_DOCK_WIDGET_VIEW_LIST,
  CAIRO_DOCK_WIDGET_THEME_LIST,
  CAIRO_DOCK_WIDGET_ANIMATION_LIST,
  CAIRO_DOCK_WIDGET_DIALOG_DECORATOR_LIST,
  CAIRO_DOCK_WIDGET_DESKLET_DECORATION_LIST,
  CAIRO_DOCK_WIDGET_DESKLET_DECORATION_LIST_WITH_DEFAULT,
  CAIRO_DOCK_WIDGET_DOCK_LIST,
  CAIRO_DOCK_WIDGET_ICONS_LIST,
  CAIRO_DOCK_WIDGET_ICON_THEME_LIST,
  CAIRO_DOCK_WIDGET_SCREENS_LIST,
  CAIRO_DOCK_WIDGET_JUMP_TO_MODULE,
  CAIRO_DOCK_WIDGET_JUMP_TO_MODULE_IF_EXISTS,
  CAIRO_DOCK_WIDGET_LAUNCH_COMMAND,
  CAIRO_DOCK_WIDGET_LAUNCH_COMMAND_IF_CONDITION,
  CAIRO_DOCK_WIDGET_STRING_ENTRY,
  CAIRO_DOCK_WIDGET_FILE_SELECTOR,
  CAIRO_DOCK_WIDGET_IMAGE_SELECTOR,
  CAIRO_DOCK_WIDGET_FOLDER_SELECTOR,
  CAIRO_DOCK_WIDGET_SOUND_SELECTOR,
  CAIRO_DOCK_WIDGET_SHORTKEY_SELECTOR,
  CAIRO_DOCK_WIDGET_CLASS_SELECTOR,
  CAIRO_DOCK_WIDGET_PASSWORD_ENTRY,
  CAIRO_DOCK_WIDGET_FONT_SELECTOR,
  CAIRO_DOCK_WIDGET_LIST,
  CAIRO_DOCK_WIDGET_LIST_WITH_ENTRY,
  CAIRO_DOCK_WIDGET_NUMBERED_LIST,
  CAIRO_DOCK_WIDGET_NUMBERED_CONTROL_LIST,
  CAIRO_DOCK_WIDGET_NUMBERED_CONTROL_LIST_SELECTIVE,
  CAIRO_DOCK_WIDGET_TREE_VIEW_SORT,
  CAIRO_DOCK_WIDGET_TREE_VIEW_SORT_AND_MODIFY,
  CAIRO_DOCK_WIDGET_TREE_VIEW_MULTI_CHOICE,
  CAIRO_DOCK_WIDGET_EMPTY_WIDGET,
  CAIRO_DOCK_WIDGET_EMPTY_FULL,
  CAIRO_DOCK_WIDGET_TEXT_LABEL,
  CAIRO_DOCK_WIDGET_LINK,
  CAIRO_DOCK_WIDGET_HANDBOOK,
  CAIRO_DOCK_WIDGET_SEPARATOR,
  CAIRO_DOCK_WIDGET_FRAME,

CAIRO_DOCK_WIDGET_EXPANDER }

Types of widgets that Cairo-Dock can automatically build.

- enum CairoDockGUIModelColumns

  Model used for combo-box and tree-view. CAIRO_DOCK_MODEL_NAME is the name as displayed in the widget, and CAIRO_DOCK_MODEL_RESULT is the resulting string effectively written in the config file.

Functions

- CairoDockGroupKeyWidget ∗ cairo_dock_gui_find_group_key_widget_in_list (GSList ∗pWidgetList, const gchar ∗cGroupName, const gchar ∗cKeyName)

5.31.1 Detailed Description

This class handles the construction of the common widgets used in the conf files.

A conf file is a common group/key file, with the following syntax :

```
[Group]
#comment about key1
key1 = 1
#comment about key2
key2 = pouic
```

Each key in the conf file has a comment.

The first character of the comment defines the type of widget. Known types are listed in the CairoDockGUIWidget↩Type enum.

A key can be a behaviour key or an appearance key. Appearance keys are keys that defines the look of the appli, they belong to the theme. Behaviour keys are keys that define some configuration parameters, that depends on the user. To mark a key as an apppearance one, suffix the widget character with a '+'. Thus, keys not marked with a '+' won't be loaded when the user loads a theme, except if he forces it.

After the widget character and its suffix, some widget accept a list of values. For instance, a spinbutton can have a min and a max limits, a list can have pre-defined elements, etc. Such values are set between '[' and ']' brackets, and separated by ';' inside.

After that, let a blank to start the widget description. It will appear on the left of the widget; description must be short enough to fit the config panel width.

You can complete this description with a tooltip. To do that, on a new comment line, add some text between '{' and '}' brackets. Tooltips appear above the widget when you let the mouse over it for ∼1 second. They can be as long as you want. Use '
' to insert new lines inside the tooltip.

5.31.2 Enumeration Type Documentation

5.31.2.1 enum CairoDockGUIWidgetType

Types of widgets that Cairo-Dock can automatically build.

Enumerator

CAIRO_DOCK_WIDGET_CHECK_BUTTON boolean in a button to tick.

CAIRO_DOCK_WIDGET_CHECK_CONTROL_BUTTON boolean in a button to tick, that will control the sensitivity of the next widget.

CAIRO_DOCK_WIDGET_SPIN_INTEGER integer in a spin button.

CAIRO_DOCK_WIDGET_HSCALE_INTEGER integer in an horizontal scale.

CAIRO_DOCK_WIDGET_SIZE_INTEGER pair of integers for dimansion WidthxHeight

CAIRO_DOCK_WIDGET_SPIN_DOUBLE double in a spin button.

CAIRO_DOCK_WIDGET_COLOR_SELECTOR_RGB 3 doubles with a color selector (RGB).

CAIRO_DOCK_WIDGET_COLOR_SELECTOR_RGBA 4 doubles with a color selector (RGBA).

CAIRO_DOCK_WIDGET_HSCALE_DOUBLE double in an horizontal scale.

CAIRO_DOCK_WIDGET_VIEW_LIST list of views.

CAIRO_DOCK_WIDGET_THEME_LIST list of themes in a combo, with preview and readme.

CAIRO_DOCK_WIDGET_ANIMATION_LIST list of available animations.

CAIRO_DOCK_WIDGET_DIALOG_DECORATOR_LIST list of available dialog decorators.

CAIRO_DOCK_WIDGET_DESKLET_DECORATION_LIST list of available desklet decorations.

CAIRO_DOCK_WIDGET_DESKLET_DECORATION_LIST_WITH_DEFAULT same but with the 'default' choice too.

CAIRO_DOCK_WIDGET_DOCK_LIST list of existing docks.

CAIRO_DOCK_WIDGET_ICONS_LIST list of icons of a dock.

CAIRO_DOCK_WIDGET_ICON_THEME_LIST list of installed icon themes.

CAIRO_DOCK_WIDGET_SCREENS_LIST list of screens

CAIRO_DOCK_WIDGET_JUMP_TO_MODULE a button to jump to another module inside the config panel.

CAIRO_DOCK_WIDGET_JUMP_TO_MODULE_IF_EXISTS same but only if the module exists.

CAIRO_DOCK_WIDGET_LAUNCH_COMMAND a button to launch a specific command.

CAIRO_DOCK_WIDGET_LAUNCH_COMMAND_IF_CONDITION a button to launch a specific command with a condition.

CAIRO_DOCK_WIDGET_STRING_ENTRY a text entry.

CAIRO_DOCK_WIDGET_FILE_SELECTOR a text entry with a file selector.

CAIRO_DOCK_WIDGET_IMAGE_SELECTOR a text entry with a file selector, files are filtered to only display images.

CAIRO_DOCK_WIDGET_FOLDER_SELECTOR a text entry with a folder selector.

CAIRO_DOCK_WIDGET_SOUND_SELECTOR a text entry with a file selector and a 'play' button, for sound files.

CAIRO_DOCK_WIDGET_SHORTKEY_SELECTOR a text entry with a shortkey selector.

CAIRO_DOCK_WIDGET_CLASS_SELECTOR a text entry with a class selector.

CAIRO_DOCK_WIDGET_PASSWORD_ENTRY a text entry, where text is hidden and the result is encrypted in the .conf file.

CAIRO_DOCK_WIDGET_FONT_SELECTOR a font selector button.

CAIRO_DOCK_WIDGET_LIST a text list.

CAIRO_DOCK_WIDGET_LIST_WITH_ENTRY a combo-entry, that is to say a list where one can add a custom choice.

CAIRO_DOCK_WIDGET_NUMBERED_LIST a combo where the number of the line is used for the choice.

CAIRO_DOCK_WIDGET_NUMBERED_CONTROL_LIST a combo where the number of the line is used for the choice, and for controlling the sensitivity of the widgets below.

CAIRO_DOCK_WIDGET_NUMBERED_CONTROL_LIST_SELECTIVE a combo where the number of the line is used for the choice, and for controlling the sensitivity of the widgets below; controlled widgets are indicated in the list : {entry;index first widget;nb widgets}.

CAIRO_DOCK_WIDGET_TREE_VIEW_SORT a tree view, where lines are numbered and can be moved up and down.

CAIRO_DOCK_WIDGET_TREE_VIEW_SORT_AND_MODIFY a tree view, where lines can be added, removed, and moved up and down.

CAIRO_DOCK_WIDGET_TREE_VIEW_MULTI_CHOICE a tree view, where lines are numbered and can be selected or not.

CAIRO_DOCK_WIDGET_EMPTY_WIDGET an empty GtkContainer, in case you need to build custom widgets.

CAIRO_DOCK_WIDGET_EMPTY_FULL an empty GtkContainer, the same but using full available space.

CAIRO_DOCK_WIDGET_TEXT_LABEL a simple text label.

CAIRO_DOCK_WIDGET_LINK a simple text label.

CAIRO_DOCK_WIDGET_HANDBOOK a label containing the handbook of the applet.

CAIRO_DOCK_WIDGET_SEPARATOR an horizontal separator.

CAIRO_DOCK_WIDGET_FRAME a frame. The previous frame will be closed.

CAIRO_DOCK_WIDGET_EXPANDER a frame inside an expander. The previous frame will be closed.

5.31.3 Function Documentation

5.31.3.1 CairoDockGroupKeyWidget∗ cairo_dock_gui_find_group_key_widget_in_list (GSList ∗ *pWidgetList,* const gchar ∗ *cGroupName,* const gchar ∗ *cKeyName*)

Get a widget from a list of widgets representing a configuration window. The widgets represent a pair (group,key) as defined in the config file.

Parameters

pWidgetList	list of widgets built from the config file
cGroupName	name of the group the widget belongs to
cKeyName	name of the key the widget represents

Returns

the widget asociated with the (group,key) , or NULL if none is found

5.32 cairo-dock-gui-manager.h File Reference

Data Structures

- struct _CairoDockGuiBackend

  Definition of the GUI interface for modules.

Macros

- #define cairo_dock_reload_current_module_widget(pModuleInstance)

Typedefs

- typedef gboolean(∗ CairoDockApplyConfigFunc)(gpointer data)

  Definition of the callback called when the user apply the config panel.

Functions

- void cairo_dock_set_status_message (GtkWidget ∗pWindow, const gchar ∗cMessage)
- void cairo_dock_set_status_message_printf (GtkWidget ∗pWindow, const gchar ∗cFormat,...) G_GNUC_P←
  RINTF(2

5.32.1 Detailed Description

This class provides functions to act on configuration windows.

It also defines the interface that a GUI backend should implement.

Note: GUIs are built from a .conf file; .conf files are normal group/key files, but with some special indications in the comments. Each key will be represented by a pre-defined widget, that is defined by the first caracter of its comment. The comment also contains a description of the key, and an optionnal tooltip. See cairo-dock-gui-factory.h for the list of pre-defined widgets and a short explanation on how to use them inside a conf file. The file 'cairo-dock.conf' can be an useful example.

5.32.2 Macro Definition Documentation

5.32.2.1 #define cairo_dock_reload_current_module_widget(*pModuleInstance*)

Reload the widget of a given module instance if it is currently opened (the current page is displayed). This is useful if the module has modified its conf file and wishes to display the changes.

Parameters

pModule←Instance	an instance of a module.

5.32.3 Function Documentation

5.32.3.1 void cairo_dock_set_status_message (GtkWidget ∗ *pWindow,* const gchar ∗ *cMessage*)

Display a message on a given window that has a status-bar. If no window is provided, the current config panel will be used.

Parameters

pWindow	window where the message should be displayed, or NULL to target the config panel.
cMessage	the message.

5.32.3.2 void cairo_dock_set_status_message_printf (GtkWidget ∗ *pWindow,* const gchar ∗ *cFormat,* ...)

Display a message on a given window that has a status-bar. If no window is provided, the current config panel will be used.

Parameters

pWindow	window where the message should be displayed, or NULL to target the config panel.
cFormat	the message, in a printf-like format
...	arguments of the format.

5.33 cairo-dock-hiding-effect.h File Reference

5.33.1 Detailed Description

This class implements the rendering interface for hiding docks.

5.34 cairo-dock-icon-container.h File Reference

5.34.1 Detailed Description

This class implements the rendering interface for icons pointing on a sub-dock.

5.35 cairo-dock-icon-facility.h File Reference

Macros

- #define cairo_dock_icon_is_being_inserted(icon)
- #define cairo_dock_icon_is_being_removed(icon)
- #define cairo_dock_get_icon_order(icon)
- #define cairo_dock_get_next_element(ic, list)
- #define cairo_dock_get_previous_element(ic, list)
- #define cairo_dock_set_icon_static(icon, _bStatic)
- #define cairo_dock_set_icon_always_visible(icon, _bAlwaysVisible)
- #define gldi_icon_mark_as_launching(pIcon)
- #define gldi_icon_is_launching(pIcon)

Functions

- CairoDockIconGroup cairo_dock_get_icon_type (Icon ∗icon)
- int cairo_dock_compare_icons_order (Icon ∗icon1, Icon ∗icon2)
- int cairo_dock_compare_icons_name (Icon ∗icon1, Icon ∗icon2)
- int cairo_dock_compare_icons_extension (Icon ∗icon1, Icon ∗icon2)
- GList ∗ cairo_dock_sort_icons_by_order (GList ∗pIconList)
- GList ∗ cairo_dock_sort_icons_by_name (GList ∗pIconList)
- Icon ∗ cairo_dock_get_first_icon (GList ∗pIconList)
- Icon ∗ cairo_dock_get_last_icon (GList ∗pIconList)
- Icon ∗ cairo_dock_get_first_icon_of_group (GList ∗pIconList, CairoDockIconGroup iGroup)
- Icon ∗ cairo_dock_get_last_icon_of_group (GList ∗pIconList, CairoDockIconGroup iGroup)
- Icon ∗ cairo_dock_get_first_icon_of_order (GList ∗pIconList, CairoDockIconGroup iGroup)
- Icon ∗ cairo_dock_get_last_icon_of_order (GList ∗pIconList, CairoDockIconGroup iGroup)
- Icon ∗ cairo_dock_get_pointed_icon (GList ∗pIconList)
- Icon ∗ cairo_dock_get_next_icon (GList ∗pIconList, Icon ∗pIcon)
- Icon ∗ cairo_dock_get_previous_icon (GList ∗pIconList, Icon ∗pIcon)
- Icon ∗ cairo_dock_get_icon_with_command (GList ∗pIconList, const gchar ∗cCommand)
- Icon ∗ cairo_dock_get_icon_with_base_uri (GList ∗pIconList, const gchar ∗cBaseURI)
- Icon ∗ cairo_dock_get_icon_with_name (GList ∗pIconList, const gchar ∗cName)
- Icon ∗ cairo_dock_get_icon_with_subdock (GList ∗pIconList, CairoDock ∗pSubDock)
- void cairo_dock_get_icon_extent (Icon ∗pIcon, int ∗iWidth, int ∗iHeight)
- void cairo_dock_get_current_icon_size (Icon ∗pIcon, GldiContainer ∗pContainer, double ∗fSizeX, double ∗f↩ SizeY)
- void cairo_dock_compute_icon_area (Icon ∗icon, GldiContainer ∗pContainer, GdkRectangle ∗pArea)
- void gldi_icon_set_name (Icon ∗pIcon, const gchar ∗cIconName)
- void gldi_icon_set_name_printf (Icon ∗pIcon, const gchar ∗cIconNameFormat,...) G_GNUC_PRINTF(2
- void void gldi_icon_set_quick_info (Icon ∗pIcon, const gchar ∗cQuickInfo)
- void gldi_icon_set_quick_info_printf (Icon ∗pIcon, const gchar ∗cQuickInfoFormat,...) G_GNUC_PRINTF(2
- gboolean cairo_dock_begin_draw_icon (Icon ∗pIcon, gint iRenderingMode)
- void cairo_dock_end_draw_icon (Icon ∗pIcon)

5.35.1 Detailed Description

This class provides utility functions on Icons.

5.35.2 Macro Definition Documentation

5.35.2.1 #define cairo_dock_icon_is_being_inserted(*icon*)

Say whether an icon is currently being inserted.

5.35.2.2 #define cairo_dock_icon_is_being_removed(*icon*)

Say whether an icon is currently being removed.

5.35.2.3 #define cairo_dock_get_icon_order(*icon*)

Get the group order of an icon. 3 groups are available by default : launchers, applis, and applets, and each group has an order.

5.35.2.4 #define cairo_dock_get_next_element(*ic,* *list*)

Get the next element in a list, looping if necessary..

Parameters

ic	the current element.
list	a list.

Returns

the next element, or the first element of the list if ’ic’ is the last one.

5.35.2.5 #define cairo_dock_get_previous_element(*ic,* *list*)

Get the previous element in a list, looping if necessary..

Parameters

ic	the current element.
list	a list.

Returns

the previous element, or the last element of the list if ’ic’ is the first one.

5.35.2.6 #define cairo_dock_set_icon_static(*icon,* *_bStatic*)

Make an icon static or not. Static icons are not animated when mouse hovers them.

Parameters

icon	an icon.
_bStatic	static or not.

5.35.2.7 #define cairo_dock_set_icon_always_visible(*icon,* *_bAlwaysVisible*)

Make an icon always visible, even when the dock is hidden.

Parameters

icon	an icon.
_bAlwaysVisible	whether the icon is always visible or not.

5.35.2.8 #define gldi_icon_mark_as_launching(*pIcon*)

Mark an Icon as 'launching'. This states lasts until the corresponding window appears (with a timeout of 15 seconds). Typically used to prevent the program from being started 2 times in a row, or to keep the animation running until the program is started.

5.35.2.9 #define gldi_icon_is_launching(*pIcon*)

Tell if an Icon is being launched.

5.35.3 Function Documentation

5.35.3.1 CairoDockIconGroup cairo_dock_get_icon_type (Icon ∗ *icon*)

Get the type of an icon according to its content (launcher, appli, applet). This can be different from its group.

Parameters

icon	the icon.

Returns

the type of the icon.

5.35.3.2 int cairo_dock_compare_icons_order (Icon ∗ *icon1,* Icon ∗ *icon2*)

Compare 2 icons with the order relation on (group order, icon order).

Parameters

icon1	an icon.
icon2	another icon.

Returns

-1 if icon1 $<$ icon2, 1 if icon1 $>$ icon2, 0 if icon1 = icon2.

5.35.3.3 int cairo_dock_compare_icons_name (Icon ∗ *icon1,* Icon ∗ *icon2*)

Compare 2 icons with the order relation on the name (case unsensitive alphabetical order).

Parameters

icon1	an icon.
icon2	another icon.

Returns

-1 if icon1 $<$ icon2, 1 if icon1 $>$ icon2, 0 if icon1 = icon2.

5.35.3.4 int cairo_dock_compare_icons_extension (Icon * *icon1*, Icon * *icon2*)

Compare 2 icons with the order relation on the extension of their URIs (case unsensitive alphabetical order).

Parameters

icon1	an icon.
icon2	another icon.

Returns

-1 if icon1 $<$ icon2, 1 if icon1 $>$ icon2, 0 if icon1 = icon2.

5.35.3.5 GList∗ cairo_dock_sort_icons_by_order (GList ∗ *pIconList*)

Sort a list with the order relation on (group order, icon order).

Parameters

pIconList	a list of icons.

Returns

the sorted list. Elements are the same as the initial list, only their order has changed.

5.35.3.6 GList∗ cairo_dock_sort_icons_by_name (GList ∗ *pIconList*)

Sort a list with the alphabetical order on the icons' name.

Parameters

pIconList	a list of icons.

Returns

the sorted list. Elements are the same as the initial list, only their order has changed. Icon's orders are updated to reflect the new order.

5.35.3.7 Icon∗ cairo_dock_get_first_icon (GList ∗ *pIconList*)

Get the first icon of a list of icons.

Parameters

pIconList	a list of icons.

Returns

the first icon, or NULL if the list is empty.

5.35.3.8 Icon∗ cairo_dock_get_last_icon (GList ∗ *pIconList*)

Get the last icon of a list of icons.

Parameters

pIconList	a list of icons.

Returns

the last icon, or NULL if the list is empty.

5.35.3.9 Icon∗ cairo_dock_get_first_icon_of_group (GList ∗ *pIconList,* CairoDockIconGroup *iGroup*)

Get the first icon of a given group.

Parameters

pIconList	a list of icons.
iGroup	the group of icon.

Returns

the first found icon with this group, or NULL if none matches.

5.35.3.10 Icon∗ cairo_dock_get_last_icon_of_group (GList ∗ *pIconList,* CairoDockIconGroup *iGroup*)

Get the last icon of a given group.

Parameters

pIconList	a list of icons.
iGroup	the group of icon.

Returns

the last found icon with this group, or NULL if none matches.

5.35.3.11 Icon∗ cairo_dock_get_first_icon_of_order (GList ∗ *pIconList,* CairoDockIconGroup *iGroup*)

Get the first icon whose group has the same order as a given one.

Parameters

pIconList	a list of icons.
iGroup	a group of icon.

Returns

the first found icon, or NULL if none matches.

5.35.3.12 Icon∗ cairo_dock_get_last_icon_of_order (GList ∗ *pIconList,* CairoDockIconGroup *iGroup*)

Get the last icon whose group has the same order as a given one.

Parameters

pIconList	a list of icons.
iGroup	a group of icon.

Returns

the last found icon, or NULL if none matches.

5.35.3.13 Icon∗ cairo_dock_get_pointed_icon (GList ∗ *pIconList*)

Get the currently pointed icon in a list of icons.

Parameters

pIconList	a list of icons.

Returns

the icon whose field 'bPointed' is TRUE, or NULL if none is pointed.

5.35.3.14 Icon∗ cairo_dock_get_next_icon (GList ∗ *pIconList,* Icon ∗ *pIcon*)

Get the icon next to a given one. The cost is O(n).

Parameters

pIconList	a list of icons.
pIcon	an icon in the list.

Returns

the icon whose left neighboor is pIcon, or NULL if the list is empty or if pIcon is the last icon.

5.35.3.15 Icon∗ cairo_dock_get_previous_icon (GList ∗ *pIconList,* Icon ∗ *pIcon*)

Get the icon previous to a given one. The cost is O(n).

Parameters

pIconList	a list of icons.
pIcon	an icon in the list.

Returns

the icon whose right neighboor is pIcon, or NULL if the list is empty or if pIcon is the first icon.

5.35.3.16 Icon∗ cairo_dock_get_icon_with_command (GList ∗ *pIconList,* const gchar ∗ *cCommand*)

Search an icon with a given command in a list of icons.

Parameters

pIconList	a list of icons.
cCommand	the command.

Returns

the first icon whose field 'cCommand' is identical to the given command, or NULL if no icon matches.

5.35.3.17 Icon∗ cairo_dock_get_icon_with_base_uri (GList ∗ *pIconList,* const gchar ∗ *cBaseURI*)

Search an icon with a given URI in a list of icons.

Parameters

pIconList	a list of icons.
cBaseURI	the URI.

Returns

the first icon whose field 'cURI' is identical to the given URI, or NULL if no icon matches.

5.35.3.18 Icon∗ cairo_dock_get_icon_with_name (GList ∗ *pIconList,* const gchar ∗ *cName*)

Search an icon with a given name in a list of icons.

Parameters

pIconList	a list of icons.
cName	the name.

Returns

the first icon whose field 'cName' is identical to the given name, or NULL if no icon matches.

5.35.3.19 Icon∗ cairo_dock_get_icon_with_subdock (GList ∗ *pIconList,* CairoDock ∗ *pSubDock*)

Search the icon pointing on a given sub-dock in a list of icons.

Parameters

pIconList	a list of icons.
pSubDock	a sub-dock.

Returns

the first icon whose field 'pSubDock' is equal to the given sub-dock, or NULL if no icon matches.

5.35.3.20 void cairo_dock_get_icon_extent (Icon ∗ *pIcon,* int ∗ *iWidth,* int ∗ *iHeight*)

Get the dimension allocated to the surface/texture of an icon.

Parameters

pIcon	the icon.
iWidth	pointer to the width.
iHeight	pointer to the height.

5.35.3.21 void cairo_dock_get_current_icon_size (Icon ∗ *pIcon,* GldiContainer ∗ *pContainer,* double ∗ *fSizeX,* double ∗ *fSizeY*)

Get the current size of an icon as it is seen on the screen (taking into account the zoom and the ratio).

Parameters

pIcon	the icon
pContainer	its container
fSizeX	pointer to the X size (horizontal)
fSizeY	pointer to the Y size (vertical)

5.35.3.22 void cairo_dock_compute_icon_area (Icon ∗ *icon,* GldiContainer ∗ *pContainer,* GdkRectangle ∗ *pArea*)

Get the total zone used by an icon on its container (taking into account reflect, gap to reflect, zoom and stretching).

Parameters

icon	the icon
pContainer	its container
pArea	a rectangle filled with the zone used by the icon on its container.

5.35.3.23 void gldi_icon_set_name (Icon ∗ *pIcon,* const gchar ∗ *cIconName*)

Set the label of an icon. If it has a sub-dock, it is renamed (the name is possibly altered to stay unique). The label buffer is updated too.

Parameters

pIcon	the icon.
cIconName	the new label of the icon. You can even pass pIcon->cName.

5.35.3.24 void gldi_icon_set_name_printf (Icon ∗ *pIcon,* const gchar ∗ *cIconNameFormat,* ...)

Same as above, but takes a printf-like format string.

Parameters

pIcon	the icon.
cIconName↩ Format	the new label of the icon, in a 'printf' way.
...	data to be inserted into the string.

5.35.3.25 void void gldi_icon_set_quick_info (Icon ∗ *pIcon,* const gchar ∗ *cQuickInfo*)

Set the quick-info of an icon. This is a small text (a few characters) that is superimposed on the icon.

Parameters

pIcon	the icon.
cQuickInfo	the text of the quick-info. If NULL, will just remove the current the quick-info.

5.35.3.26 void gldi_icon_set_quick_info_printf (Icon ∗ *pIcon,* const gchar ∗ *cQuickInfoFormat,* ...)

Same as above, but takes a printf-like format string.

Parameters

pIcon	the icon.
cQuickInfo↩ Format	the text of the quick-info, in a 'printf' way.
...	data to be inserted into the string.

5.35.3.27 gboolean cairo_dock_begin_draw_icon (Icon ∗ *pIcon,* gint *iRenderingMode*)

Initiate an OpenGL drawing session on an icon's texture.

Parameters

pIcon	the icon on which to draw.
iRenderingMode	rendering mode. 0:normal, 1:don't clear the current texture, so that the drawing will be superimposed on it, 2:keep the current icon texture unchanged for all the drawing (the drawing is made on another texture).

Returns

TRUE if you can proceed to the drawing, FALSE if an error occured.

5.35.3.28 void cairo_dock_end_draw_icon (Icon ∗ *pIcon*)

Finish an OpenGL drawing session on an icon.

Parameters

pIcon	the icon on which to draw.

Returns

TRUE if you can proceed to the drawing, FALSE if an error occured.

5.36 cairo-dock-icon-factory.h File Reference

Data Structures

- struct _IconInterface

  Icon's interface.
- struct _Icon

  Definition of an Icon.
- struct _CairoIconContainerRenderer

  Definition of an Icon container (= an icon holding a sub-dock) renderer.

Macros

- #define CAIRO_DOCK_IS_ICON(obj)
- #define CAIRO_DOCK_IS_APPLI(icon)
- #define CAIRO_DOCK_IS_APPLET(icon)
- #define CAIRO_DOCK_IS_MULTI_APPLI(icon)
- #define CAIRO_DOCK_IS_AUTOMATIC_SEPARATOR(icon)
- #define CAIRO_DOCK_IS_USER_SEPARATOR(icon)
- #define CAIRO_DOCK_IS_NORMAL_APPLI(icon)
- #define CAIRO_DOCK_IS_DETACHABLE_APPLET(icon)

Enumerations

- enum CairoDockIconGroup

  Available groups of icons.
- enum CairoDockAnimationState

  Animation state of an icon, sorted by priority.

Functions

- Icon ∗ gldi_icon_new (void)
- Icon ∗ cairo_dock_create_dummy_launcher (gchar ∗cName, gchar ∗cFileName, gchar ∗cCommand, gchar ∗cQuickInfo, double fOrder)
- void cairo_dock_load_icon_image (Icon ∗icon, GldiContainer ∗pContainer)
- void cairo_dock_load_icon_text (Icon ∗icon)
- void cairo_dock_load_icon_quickinfo (Icon ∗icon)
- void cairo_dock_load_icon_buffers (Icon ∗pIcon, GldiContainer ∗pContainer)

5.36.1 Detailed Description

This class defines the items contained in containers : Icons. An icon can either be:

- a launcher (it has a command, a class, and possible an X window ID)
- an appli (it has a X window ID and a class, no command)
- an applet (it has a module instance and no command, possibly a class)
- a container (it has a sub-dock and no class nor command)
- a class icon (it has a bsub-dock and a class, but no command nor X ID)
- a separator (it has nothing)

The class defines the methods used to create a generic Icon and to load its various buffers. Specialized Icons are created by the corresponding factory.

5.36.2 Macro Definition Documentation

5.36.2.1 #define CAIRO_DOCK_IS_ICON(*obj*)

Say if an object is an Icon.

Parameters

obj	the object.

Returns

TRUE if the object is an icon.

5.36.2.2 #define CAIRO_DOCK_IS_APPLI(*icon*)

TRUE if the icon holds a window.

Parameters

icon	an icon.

5.36.2.3 #define CAIRO_DOCK_IS_APPLET(*icon*)

TRUE if the icon holds an instance of a module.

Parameters

icon	an icon.

5.36.2.4 #define CAIRO_DOCK_IS_MULTI_APPLI(*icon*)

TRUE if the icon is an icon pointing on the sub-dock of a class.

Parameters

icon	an icon.

5.36.2.5 #define CAIRO_DOCK_IS_AUTOMATIC_SEPARATOR(*icon*)

TRUE if the icon is an automatic separator.

Parameters

icon	an icon.

5.36.2.6 #define CAIRO_DOCK_IS_USER_SEPARATOR(*icon*)

TRUE if the icon is a separator added by the user.

Parameters

icon	an icon.

5.36.2.7 #define CAIRO_DOCK_IS_NORMAL_APPLI(*icon*)

TRUE if the icon is an icon d'appli only.

Parameters

icon	an icon.

5.36.2.8 #define CAIRO_DOCK_IS_DETACHABLE_APPLET(*icon*)

TRUE if the icon is an icon d'applet detachable en desklet.

Parameters

icon	an icon.

5.36.3 Function Documentation

5.36.3.1 Icon∗ gldi_icon_new (void)

Create an empty icon.

Returns

the newly allocated icon object.

5.36.3.2 Icon∗ cairo_dock_create_dummy_launcher (gchar ∗ *cName,* gchar ∗ *cFileName,* gchar ∗ *cCommand,* gchar ∗ *cQuickInfo,* double *fOrder*)

Create an Icon that will behave like a launcher. It's especially useful for applets that want to fill a sub-dock or a desklet (the icon is not loaded by the function). Be careful that the strings are not duplicated. Therefore, you must use g_strdup() if you want to set a constant string; and must not free the strings after calling this function.

Parameters

cName	label of the icon
cFileName	name of an image
cCommand	a command, or NULL
cQuickInfo	a quick-info, or NULL
fOrder	order of the icon in its container.

Returns

the newly created icon.

5.36.3.3 void cairo_dock_load_icon_image (Icon ∗ *icon,* GldiContainer ∗ *pContainer*)

Fill the image buffer (surface & texture) of a given icon, according to its type. Set its size if necessary, and fills the reflection buffer for cairo.

Parameters

icon	the icon.
pContainer	its container.

5.36.3.4 void cairo_dock_load_icon_text (Icon ∗ *icon*)

Fill the label buffer (surface & texture) of a given icon, according to a text description.

Parameters

icon	the icon.

5.36.3.5 void cairo_dock_load_icon_quickinfo (Icon ∗ *icon*)

Fill the quick-info buffer (surface & texture) of a given icon, according to a text description.

Parameters

icon	the icon.

5.36.3.6 void cairo_dock_load_icon_buffers (Icon ∗ *pIcon,* GldiContainer ∗ *pContainer*)

Fill all the buffers (surfaces & textures) of a given icon, according to its type. Set its size accordingly, and fills the reflection buffer for cairo. Label and quick-info are loaded with the current global text description.

Parameters

pIcon	the icon.
pContainer	its container.

5.37 cairo-dock-icon-manager.h File Reference

Enumerations

- enum CairoIconNotifications {
  NOTIFICATION_UNFOLD_SUBDOCK,
  NOTIFICATION_UPDATE_ICON,
  NOTIFICATION_UPDATE_ICON_SLOW,
  NOTIFICATION_PRE_RENDER_ICON,
  NOTIFICATION_RENDER_ICON,
  NOTIFICATION_STOP_ICON,
  NOTIFICATION_REQUEST_ICON_ANIMATION }

    signals

Functions

- void gldi_icons_foreach (GldiIconFunc pFunction, gpointer pUserData)
- gint cairo_dock_search_icon_size (GtkIconSize iIconSize)
- gchar ∗ cairo_dock_search_icon_s_path (const gchar ∗cFileName, gint iDesiredIconSize)

5.37.1 Detailed Description

This class manages the icons parameters and their associated ressources.

Specialized Icons are handled by the corresponding manager.

5.37.2 Enumeration Type Documentation

5.37.2.1 enum CairoIconNotifications

signals

Enumerator

NOTIFICATION_UNFOLD_SUBDOCK notification called when an icon's sub-dock is starting to (un)fold. data : {Icon}

NOTIFICATION_UPDATE_ICON notification called when an icon is updated in the fast rendering loop.

NOTIFICATION_UPDATE_ICON_SLOW notification called when an icon is updated in the slow rendering loop.

NOTIFICATION_PRE_RENDER_ICON notification called when the background of an icon is rendered.

NOTIFICATION_RENDER_ICON notification called when an icon is rendered.

NOTIFICATION_STOP_ICON notification called when an icon is stopped, for instance before it is removed.

NOTIFICATION_REQUEST_ICON_ANIMATION notification called when someone asks for an animation for a given icon.

5.37.3 Function Documentation

5.37.3.1 void gldi_icons_foreach (GldiIconFunc *pFunction*, gpointer *pUserData*)

Execute an action on all icons.

Parameters

pFunction	the action.
pUserData	data passed to the callback.

5.37.3.2 gint cairo_dock_search_icon_size (GtkIconSize *iIconSize*)

Search the icon size of a GtkIconSize.

Parameters

iIconSize	a GtkIconSize

Returns

the maximum between the width and the height of the icon size in pixel (or 128 if there is a problem)

5.37.3.3 gchar∗ cairo_dock_search_icon_s_path (const gchar ∗ *cFileName*, gint *iDesiredIconSize*)

Search the path of an icon into the defined icons themes. It also handles the '∼' caracter in paths.

Parameters

cFileName	name of the icon file.
iDesiredIconSize	desired icon size if we use icons from user icons theme.

Returns

the complete path of the icon, or NULL if not found.

5.38 cairo-dock-image-buffer.h File Reference

Data Structures

- struct _CairoDockImageBuffer

  Definition of an Image Buffer. It provides an unified interface for a cairo/opengl image buffer.

Macros

- #define cairo_dock_load_image_buffer(pImage, cImageFile, iWidth, iHeight, iLoadModifier)
- #define cairo_dock_apply_image_buffer_surface(pImage, pCairoContext)
- #define cairo_dock_apply_image_buffer_texture(pImage)

Functions

- gchar ∗ cairo_dock_search_image_s_path (const gchar ∗cImageFile)
- void cairo_dock_load_image_buffer_full (CairoDockImageBuffer ∗pImage, const gchar ∗cImageFile, int i↩ Width, int iHeight, CairoDockLoadImageModifier iLoadModifier, double fAlpha)
- void cairo_dock_load_image_buffer_from_surface (CairoDockImageBuffer ∗pImage, cairo_surface_t ∗p↩ Surface, int iWidth, int iHeight)
- CairoDockImageBuffer ∗ cairo_dock_create_image_buffer (const gchar ∗cImageFile, int iWidth, int iHeight, CairoDockLoadImageModifier iLoadModifier)
- void cairo_dock_unload_image_buffer (CairoDockImageBuffer ∗pImage)
- void cairo_dock_free_image_buffer (CairoDockImageBuffer ∗pImage)
- void cairo_dock_apply_image_buffer_surface_with_offset (const CairoDockImageBuffer ∗pImage, cairo_↩ t ∗pCairoContext, double x, double y, double fAlpha)
- void cairo_dock_apply_image_buffer_texture_with_offset (const CairoDockImageBuffer ∗pImage, double x, double y)
- void cairo_dock_apply_image_buffer_surface_at_size (const CairoDockImageBuffer ∗pImage, cairo_t ∗p↩ CairoContext, int w, int h, double x, double y, double fAlpha)
- void cairo_dock_apply_image_buffer_texture_at_size (const CairoDockImageBuffer ∗pImage, int w, int h, double x, double y)
- void cairo_dock_create_icon_fbo (void)
- void cairo_dock_destroy_icon_fbo (void)

5.38.1 Detailed Description

This class defines a generic image API that works for both Cairo and OpenGL. It allows to easily load and display images, without having to care the rendering mode. It supports animated images (an animated image is made of several frames, ordered side by side from left to right).

Use cairo_dock_create_image_buffer to create an image buffer from a file, or cairo_dock_load_image_buffer to load an image into an existing image buffer. Use cairo_dock_free_image_buffer to destroy it or cairo_dock_unload_↩ image_buffer to unload and reset it to 0.

Use cairo_dock_apply_image_buffer_surface or cairo_dock_apply_image_buffer_texture to display the image.

5.38.2 Macro Definition Documentation

5.38.2.1 cairo_dock_load_image_buffer(*pImage, cImageFile, iWidth, iHeight, iLoadModifier*)

Load an image into an ImageBuffer. If the image is given by its sole name, it is taken in the root folder of the current theme.

Parameters

pImage	an ImageBuffer.
cImageFile	name of a file
iWidth	width it should be loaded. The resulting width can be different depending on the modifier.
iHeight	height it should be loaded. The resulting width can be different depending on the modifier.
iLoadModifier	modifier

5.38.2.2 #define cairo_dock_apply_image_buffer_surface(*pImage, pCairoContext*)

Draw an ImageBuffer on a cairo context.

Parameters

pImage	an ImageBuffer.
pCairoContext	the current cairo context.

5.38.2.3 #define cairo_dock_apply_image_buffer_texture(*pImage*)

Draw an ImageBuffer on the current OpenGL context.

Parameters

pImage	an ImageBuffer.

5.38.3 Function Documentation

5.38.3.1 gchar∗ cairo_dock_search_image_s_path (const gchar ∗ *cImageFile*)

Find the path of an image. '∼' is handled, as well as the 'images' folder of the current theme. Use cairo_dock_↩ search_icon_s_path to search theme icons.

Parameters

cImageFile	a file name or path. If it's already a path, it will just be duplicated.

Returns

the path of the file, or NULL if it has not been found.

5.38.3.2 void cairo_dock_load_image_buffer_full (CairoDockImageBuffer ∗ *pImage,* const gchar ∗ *cImageFile,* int *iWidth,* int *iHeight,* CairoDockLoadImageModifier *iLoadModifier,* double *fAlpha*)

Load an image into an ImageBuffer with a given transparency. If the image is given by its sole name, it is taken in the root folder of the current theme.

Parameters

pImage	an ImageBuffer.
cImageFile	name of a file
iWidth	width it should be loaded.
iHeight	height it should be loaded.
iLoadModifier	modifier
fAlpha	transparency (1:fully opaque)

5.38.3.3 void cairo_dock_load_image_buffer_from_surface (CairoDockImageBuffer ∗ *pImage,* cairo_surface_t ∗ *pSurface,* int *iWidth,* int *iHeight*)

Load a surface into an ImageBuffer.

Parameters

pImage	an ImageBuffer.
pSurface	a cairo surface
iWidth	width of the surface
iHeight	height of the surface

5.38.3.4 CairoDockImageBuffer∗ cairo_dock_create_image_buffer (const gchar ∗ *cImageFile,* int *iWidth,* int *iHeight,* CairoDockLoadImageModifier *iLoadModifier*)

Create and load an image into an ImageBuffer. If the image is given by its sole name, it is taken in the root folder of the current theme.

Parameters

cImageFile	name of a file
iWidth	width it should be loaded.
iHeight	height it should be loaded.
iLoadModifier	modifier

Returns

a newly allocated ImageBuffer.

5.38.3.5 void cairo_dock_unload_image_buffer (CairoDockImageBuffer ∗ *pImage*)

Reset an ImageBuffer's ressources. It can be used to load another image then.

Parameters

pImage	an ImageBuffer.

5.38.3.6 void cairo_dock_free_image_buffer (CairoDockImageBuffer ∗ *pImage*)

Reset and free an ImageBuffer.

Parameters

pImage	an ImageBuffer.

5.38.3.7 void cairo_dock_apply_image_buffer_surface_with_offset (const CairoDockImageBuffer ∗ *pImage*, cairo_t ∗ *pCairoContext*, double *x*, double *y*, double *fAlpha*)

Draw an ImageBuffer with an offset on a Cairo context, at the size it was loaded.

Parameters

pImage	an ImageBuffer.
pCairoContext	the current cairo context.
x	horizontal offset.
y	vertical offset.
fAlpha	transparency (in [0;1])

5.38.3.8 void cairo_dock_apply_image_buffer_texture_with_offset (const CairoDockImageBuffer ∗ *pImage*, double *x*, double *y*)

Draw an ImageBuffer with an offset on the current OpenGL context, at the size it was loaded.

Parameters

pImage	an ImageBuffer.
x	horizontal offset.
y	vertical offset.

5.38.3.9 void cairo_dock_apply_image_buffer_surface_at_size (const CairoDockImageBuffer ∗ *pImage*, cairo_t ∗ *pCairoContext*, int *w*, int *h*, double *x*, double *y*, double *fAlpha*)

Draw an ImageBuffer with an offset on a Cairo context, at a given size.

Parameters

pImage	an ImageBuffer.
pCairoContext	the current cairo context.
w	requested width
h	requested height
x	horizontal offset.
y	vertical offset.
fAlpha	transparency (in [0;1])

5.38.3.10 void cairo_dock_apply_image_buffer_texture_at_size (const CairoDockImageBuffer ∗ *pImage*, int *w*, int *h*, double *x*, double *y*)

Draw an ImageBuffer on the current OpenGL context at a given size.

Parameters

pImage	an ImageBuffer.
w	requested width
h	requested height
x	horizontal offset.
y	vertical offset.

5.38.3.11 void cairo_dock_create_icon_fbo (void)

Create an FBO to render the icons inside a dock.

5.38.3.12 void cairo_dock_destroy_icon_fbo (void)

Destroy the icons FBO.

5.39 cairo-dock-indicator-manager.h File Reference

5.39.1 Detailed Description

This class manages the indicators.

5.40 cairo-dock-keybinder.h File Reference

Macros

- #define gldi_shortkey_could_grab(binding)

Typedefs

- typedef void(∗ CDBindkeyHandler)(const gchar ∗keystring, gpointer user_data)

  Definition of a callback, called when a shortcut is pressed by the user.

Functions

- GldiShortkey ∗ gldi_shortkey_new (const gchar ∗keystring, const gchar ∗cDemander, const gchar ∗c↩Description, const gchar ∗cIconFilePath, const gchar ∗cConfFilePath, const gchar ∗cGroupName, const gchar ∗cKeyName, CDBindkeyHandler handler, gpointer user_data)
- gboolean gldi_shortkey_rebind (GldiShortkey ∗binding, const gchar ∗cNewKeyString, const gchar ∗cNew↩Description)
- gboolean cairo_dock_trigger_shortkey (const gchar ∗cKeyString)

5.40.1 Detailed Description

This class defines the Shortkeys, which are objects that bind a keyboard shortcut to an action. The keyboard shortcut is defined globally on the desktop, that is to say they will be effective whatever window has the focus. Keyboard shortcuts are of the form <alt>F1 or <ctrl><shift>s.

Use gldi_shortkey_new to create a new shortkey, and simply unref it with gldi_object_unref to unbind the keyboard shortcut. To update a binding (whenever the shortcut or the description change, or just to re-grab it), use gldi_↩shortkey_rebind.

5.40.2 Macro Definition Documentation

5.40.2.1 #define gldi_shortkey_could_grab(*binding*)

Says if the shortkey of a key binding could be grabbed.

Parameters

binding	a key binding.

Returns

TRUE iif the shortkey has been successfuly grabbed by the key binding.

5.40.3 Function Documentation

5.40.3.1 GldiShortkey∗ gldi_shortkey_new (const gchar ∗ *keystring,* const gchar ∗ *cDemander,* const gchar ∗ *cDescription,* const gchar ∗ *cIconFilePath,* const gchar ∗ *cConfFilePath,* const gchar ∗ *cGroupName,* const gchar ∗ *cKeyName,* CDBindkeyHandler *handler,* gpointer *user_data*)

Create a new shortkey, that binds an action to a shortkey. Unref it when you don't want it anymore, or when 'user_data' is freed.

Parameters

keystring	a shortcut.
cDemander	the actor making the demand
cDescription	a short description of the action
cIconFilePath	an icon that represents the demander
cConfFilePath	conf file where the shortkey stored
cGroupName	group name where it's stored in the conf file
cKeyName	key name where it's stored in the conf file
handler	function called when the shortkey is pressed by the user

user_data	data passed to the callback

Returns

the shortkey, already bound.

5.40.3.2 gboolean gldi_shortkey_rebind (GldiShortkey ∗ *binding,* const gchar ∗ *cNewKeyString,* const gchar ∗ *cNewDescription*)

Rebind a shortkey to a new one. If the shortkey is the same, don't re-bind it.

Parameters

binding	a key binding.
cNewKeyString	the new shortkey
cNewDescription	the new description, or NULL to keep the current one.

Returns

TRUE on success

5.40.3.3 gboolean cairo_dock_trigger_shortkey (const gchar ∗ *cKeyString*)

Trigger a given shortkey. It will be as if the user effectively pressed the shortkey on its keyboard. It uses the 'XTest' X extension.

Parameters

cKeyString	a shortkey.

Returns

TRUE if success.

5.41 cairo-dock-keyfile-utilities.h File Reference

Functions

- GKeyFile ∗ cairo_dock_open_key_file (const gchar ∗cConfFilePath)
- void cairo_dock_write_keys_to_file (GKeyFile ∗pKeyFile, const gchar ∗cConfFilePath)
- void cairo_dock_merge_conf_files (const gchar ∗cConfFilePath, gchar ∗cReplacementConfFilePath, gchar iIdentifier)
- void cairo_dock_upgrade_conf_file_full (const gchar ∗cConfFilePath, GKeyFile ∗pKeyFile, const gchar ∗c↩ DefaultConfFilePath, gboolean bUpdateKeys)
- void cairo_dock_get_conf_file_version (GKeyFile ∗pKeyFile, gchar ∗∗cConfFileVersion)
- gboolean cairo_dock_conf_file_needs_update (GKeyFile ∗pKeyFile, const gchar ∗cVersion)
- void cairo_dock_add_remove_element_to_key (const gchar ∗cConfFilePath, const gchar ∗cGroupName, const gchar ∗cKeyName, gchar ∗cElementName, gboolean bAdd)
- void cairo_dock_add_group_key_to_conf_file (GKeyFile ∗pKeyFile, const gchar ∗cGroupName, const gchar ∗ckeyName, const gchar ∗cInitialValue, CairoDockGUIWidgetType iWidgetType, const gchar ∗cAuthorized↩ Values, const gchar ∗cDescription, const gchar ∗cTooltip)
- void cairo_dock_remove_group_key_from_conf_file (GKeyFile ∗pKeyFile, const gchar ∗cGroupName, const gchar ∗ckeyName)
- void cairo_dock_update_keyfile (const gchar ∗cConfFilePath, GType iFirstDataType,...)

5.41.1 Detailed Description

This class provides useful functions to manipulate the conf files of Cairo-Dock, which are classic group/key pair files.

5.41.2 Function Documentation

5.41.2.1 GKeyFile∗ cairo_dock_open_key_file (const gchar ∗ *cConfFilePath*)

Open a conf file to be read/written. Returns NULL if the file couldn't be found/opened/parsed. Free it with g_key_↩ file_free after you're done.

5.41.2.2 void cairo_dock_write_keys_to_file (GKeyFile ∗ *pKeyFile,* const gchar ∗ *cConfFilePath*)

Write a key file on the disk.

5.41.2.3 void cairo_dock_merge_conf_files (const gchar ∗ *cConfFilePath,* gchar ∗ *cReplacementConfFilePath,* gchar *iIdentifier*)

Merge the values of a conf-file into another one. Keys are filtered by an identifier on the original conf-file.

Parameters

cConfFilePath	an up-to-date conf-file with old values, that will be updated.
cReplacement↩ ConfFilePath	an old conf-file containing values we want to use
iIdentifier	a character to filter the keys, or 0.

5.41.2.4 void cairo_dock_upgrade_conf_file_full (const gchar ∗ *cConfFilePath,* GKeyFile ∗ *pKeyFile,* const gchar ∗ *cDefaultConfFilePath,* gboolean *bUpdateKeys*)

Update a conf-file, by merging values from a given key-file into a template conf-file.

Parameters

cConfFilePath	path to the conf-file to update.
pKeyFile	a key-file with correct values, but old comments and possibly missing or old keys. It is not modified by the function.
cDefaultConf↩ FilePath	a template conf-file.
bUpdateKeys	whether to remove old keys (hidden and persistent) or not.

5.41.2.5 void cairo_dock_get_conf_file_version (GKeyFile ∗ *pKeyFile,* gchar ∗∗ *cConfFileVersion*)

Get the version of a conf file. The version is written on the first line of the file, as a comment.

5.41.2.6 gboolean cairo_dock_conf_file_needs_update (GKeyFile ∗ *pKeyFile,* const gchar ∗ *cVersion*)

Say if a conf file's version mismatches a given version.

5.41.2.7 void cairo_dock_add_remove_element_to_key (const gchar ∗ *cConfFilePath,* const gchar ∗ *cGroupName,* const gchar ∗ *cKeyName,* gchar ∗ *cElementName,* gboolean *bAdd*)

Add or remove a value in a list of values to a given (group,key) pair of a conf file.

5.41.2.8 void cairo_dock_add_group_key_to_conf_file (GKeyFile ∗ *pKeyFile,* const gchar ∗ *cGroupName,* const gchar ∗ *ckeyName,* const gchar ∗ *cInitialValue,* CairoDockGUIWidgetType *iWidgetType,* const gchar ∗ *cAuthorizedValues,* const gchar ∗ *cDescription,* const gchar ∗ *cTooltip*)

Add a key to a conf file, so that it can be parsed by the GUI manager.

5.41.2.9 void cairo_dock_remove_group_key_from_conf_file (GKeyFile ∗ *pKeyFile,* const gchar ∗ *cGroupName,* const gchar ∗ *ckeyName*)

Remove a key from a conf file.

5.41.2.10 void cairo_dock_update_keyfile (const gchar ∗ *cConfFilePath,* GType *iFirstDataType,* ...)

Update a conf file with a list of values of the form : {type, name of the groupe, name of the key, value}. Must end with G_TYPE_INVALID.

Parameters

cConfFilePath	path to the conf file.
iFirstDataType	type of the first value.

5.42 cairo-dock-kwin-integration.h File Reference

5.42.1 Detailed Description

This class implements the integration of Kwin inside Cairo-Dock.

5.43 cairo-dock-launcher-manager.h File Reference

Macros

- #define GLDI_OBJECT_IS_LAUNCHER_ICON(obj)

5.43.1 Detailed Description

This class handles the Launcher Icons, which are user icons used to launch a program.

5.43.2 Macro Definition Documentation

5.43.2.1 #define GLDI_OBJECT_IS_LAUNCHER_ICON(*obj*)

Say if an object is a LauncherIcon.

Parameters

obj	the object.

Returns

TRUE if the object is a LauncherIcon.

5.44 cairo-dock-manager.h File Reference

Data Structures

- struct _GldiManager

    Definition of a Manager.

Macros

- #define GLDI_OBJECT_IS_MANAGER(obj)

5.44.1 Detailed Description

This class defines the Managers. A Manager is like an internal module: it has a classic module interface, manages a set of resources, and has its own configuration.

Each manager is initialized at the beginning. When loading the current theme, get_config and load are called. When unloading the current theme, unload and reset_config are called. When reloading a part of the current theme, reset_config, get_config and load are called.

5.44.2 Macro Definition Documentation

5.44.2.1 #define GLDI_OBJECT_IS_MANAGER(*obj*)

Say if an object is a Manager.

Parameters

obj	the object.

Returns

TRUE if the object is a Manager.

5.45 cairo-dock-menu.h File Reference

Macros

- #define gldi_submenu_new(...)
- #define gldi_menu_item_new(cLabel, cImage)
- #define gldi_menu_add_sub_menu(pMenu, cLabel, cImage)

Functions

- GtkWidget ∗ gldi_menu_new (Icon ∗pIcon)
- void gldi_menu_init (GtkWidget ∗pMenu, Icon ∗pIcon)
- void gldi_menu_popup (GtkWidget ∗menu)
- GtkWidget ∗ gldi_menu_item_new_full (const gchar ∗cLabel, const gchar ∗cImage, gboolean bUse←Mnemonic, GtkIconSize iSize)
- GtkWidget ∗ gldi_menu_item_new_with_action (const gchar ∗cLabel, const gchar ∗cImage, GCallback p←Function, gpointer pData)
- GtkWidget ∗ gldi_menu_item_new_with_submenu (const gchar ∗cLabel, const gchar ∗cImage, GtkWidget ∗∗pSubMenuPtr)

- void gldi_menu_item_set_image (GtkWidget ∗pMenuItem, GtkWidget ∗image)
- GtkWidget ∗ gldi_menu_item_get_image (GtkWidget ∗pMenuItem)
- GtkWidget ∗ gldi_menu_add_item (GtkWidget ∗pMenu, const gchar ∗cLabel, const gchar ∗cImage, G←Callback pFunction, gpointer pData)
- GtkWidget ∗ gldi_menu_add_sub_menu_full (GtkWidget ∗pMenu, const gchar ∗cLabel, const gchar ∗cImage, GtkWidget ∗∗pMenuItemPtr)
- void gldi_menu_add_separator (GtkWidget ∗pMenu)

5.45.1 Detailed Description

This class defines the Menu. They are classical menus, but with a custom looking.

5.45.2 Macro Definition Documentation

5.45.2.1 #define gldi_submenu_new(...)

Creates a new sub-menu. It's just a menu that doesn't point on an Icon/Container.

5.45.2.2 #define gldi_menu_item_new(*cLabel*, *cImage*)

A convenient function to create a menu-item with a label and an image.

Parameters

cLabel	the label, or NULL
cImage	the image path or name, or NULL

Returns

the new menu-item.

5.45.2.3 #define gldi_menu_add_sub_menu(*pMenu*, *cLabel*, *cImage*)

A convenient function to add a sub-menu to a given menu.

Parameters

pMenu	the menu
cLabel	the label, or NULL
cImage	the image path or name, or NULL

Returns

the new sub-menu that has been added.

5.45.3 Function Documentation

5.45.3.1 GtkWidget∗ gldi_menu_new (Icon ∗ *pIcon*)

Creates a new menu that will point on a given Icon. If the Icon is NULL, it will be placed under the mouse.

Parameters

pIcon	the icon, or NULL

Returns

the new menu.

5.45.3.2 void gldi_menu_init (GtkWidget ∗ *pMenu,* Icon ∗ *pIcon*)

Initialize a menu, so that it can be drawn and placed correctly. It's useful if the menu was created beforehand (like a DbusMenu).

Parameters

pIcon	the icon, or NULL

5.45.3.3 void gldi_menu_popup (GtkWidget ∗ *menu*)

Pop-up a menu. The menu is placed above the icon, or above the container, or above the mouse, depending on how it has been initialized.

Parameters

menu	the menu.

5.45.3.4 GtkWidget∗ gldi_menu_item_new_full (const gchar ∗ *cLabel,* const gchar ∗ *cImage,* gboolean *bUseMnemonic,* GtkIconSize *iSize*)

Creates a menu-item, with a label and an image. The child widget of the menu-item is a gtk-label. If the label is NULL, the child widget will be NULL too (this is useful if the menu-item will hold a custom widget).

Parameters

cLabel	the label, or NULL
cImage	the image path or name, or NULL
bUseMnemonic	whether to use the mnemonic inside the label or not
iSize	size of the image, or 0 to use the default size

Returns

the new menu-item.

5.45.3.5 GtkWidget∗ gldi_menu_item_new_with_action (const gchar ∗ *cLabel,* const gchar ∗ *cImage,* GCallback *pFunction,* gpointer *pData*)

A convenient function to create a menu-item with a label, an image, and an associated action.

Parameters

cLabel	the label, or NULL
cImage	the image path or name, or NULL

pFunction	the callback
pData	the data passed to the callback

Returns

the new menu-item.

5.45.3.6 GtkWidget∗ gldi_menu_item_new_with_submenu (const gchar ∗ *cLabel,* const gchar ∗ *cImage,* GtkWidget ∗∗ *pSubMenuPtr*)

A convenient function to create a menu-item with a label, an image, and an associated sub-menu.

Parameters

cLabel	the label
cImage	the image path or name, or NULL
pSubMenuPtr	pointer that will contain the new sub-menu, or NULL

Returns

the new menu-item.

5.45.3.7 void gldi_menu_item_set_image (GtkWidget ∗ *pMenuItem,* GtkWidget ∗ *image*)

Sets a gtk-image on a menu-item. This is useful if the image can't be given by a name or path (for instance, loaded from a cairo surface).

Parameters

pMenuItem	the menu-item
image	the image

5.45.3.8 GtkWidget∗ gldi_menu_item_get_image (GtkWidget ∗ *pMenuItem*)

Gets the image of a menu-item.

Parameters

pMenuItem	the menu-item

Returns

the gtk-image

5.45.3.9 GtkWidget∗ gldi_menu_add_item (GtkWidget ∗ *pMenu,* const gchar ∗ *cLabel,* const gchar ∗ *cImage,* GCallback *pFunction,* gpointer *pData*)

A convenient function to add an item to a given menu.

Parameters

pMenu	the menu
cLabel	the label, or NULL
cImage	the image path or name, or NULL
pFunction	the callback
pData	the data passed to the callback

Returns

the new menu-entry that has been added.

5.45.3.10 GtkWidget∗ gldi_menu_add_sub_menu_full (GtkWidget ∗ *pMenu,* const gchar ∗ *cLabel,* const gchar ∗ *cImage,* GtkWidget ∗∗ *pMenuItemPtr*)

A convenient function to add a sub-menu to a given menu.

Parameters

pMenu	the menu
cLabel	the label, or NULL
cImage	the image path or name, or NULL
pMenuItemPtr	pointer that will contain the new menu-item, or NULL

Returns

the new sub-menu that has been added.

5.45.3.11 void gldi_menu_add_separator (GtkWidget ∗ *pMenu*)

A convenient function to add a separator to a given menu.

Parameters

pMenu	the menu

5.46 cairo-dock-module-instance-manager.h File Reference

Data Structures

- struct _GldiModuleInstance

  Definition of an instance of a module. A module can be instanciated several times.

Macros

- #define GLDI_OBJECT_IS_MODULE_INSTANCE(obj)

5.46.1 Detailed Description

This class defines the instances of modules.

A module-instance represents one instance of a module; it holds a set of data: the icon and its container, the config structure and its conf file, the data structure and a slot to plug datas into containers and icons. All these parameters are optionnal; a module-instance that has an icon is also called an applet.

5.46.2 Macro Definition Documentation

5.46.2.1 #define GLDI_OBJECT_IS_MODULE_INSTANCE(*obj*)

Say if an object is a Module-instance.

Parameters

obj	the object.

Returns

TRUE if the object is a Module-instance.

5.47 cairo-dock-module-manager.h File Reference

Data Structures

- struct _GldiVisitCard

  Definition of the visit card of a module. Contains everything that is statically defined for a module.
- struct _GldiModuleInterface

  Definition of the interface of a module.
- struct _GldiModule

  Definition of an external module.

Macros

- #define GLDI_OBJECT_IS_MODULE(obj)

Typedefs

- typedef gboolean(∗ GldiModulePreInit)(GldiVisitCard ∗pVisitCard, GldiModuleInterface ∗pInterface)

  Pre-init function of a module. Fills the visit card and the interface of a module.

Enumerations

- enum GldiModuleCategory

  Categories a module can be in.

Functions

- GldiModule ∗ gldi_module_new (GldiVisitCard ∗pVisitCard, GldiModuleInterface ∗pInterface)
- GldiModule ∗ gldi_module_new_from_so_file (const gchar ∗cSoFilePath)
- void gldi_modules_new_from_directory (const gchar ∗cModuleDirPath, GError ∗∗erreur)
- gchar ∗ gldi_module_get_config_dir (GldiModule ∗pModule)
- GldiModule ∗ gldi_module_get (const gchar ∗cModuleName)
- void gldi_module_activate (GldiModule ∗module)
- void gldi_module_deactivate (GldiModule ∗module)
- void gldi_module_add_instance (GldiModule ∗pModule)

  should maybe be in the module-instance too...

5.47.1 Detailed Description

This class manages the external modules of Cairo-Dock.

A module has an interface and a visit card :

- the visit card allows it to define itself (name, category, default icon, etc)
- the interface defines the entry points for init, stop, reload, read config, and reset data.

Modules can be instanciated several times; each time they are, an instance _GldiModuleInstance is created. Each instance holds a set of data: the icon and its container, the config structure and its conf file, the data structure and a slot to plug datas into containers and icons. All these data are optionnal; a module that has an icon is also called an applet.

5.47.2 Macro Definition Documentation

5.47.2.1 #define GLDI_OBJECT_IS_MODULE(*obj*)

Say if an object is a Module.

Parameters

obj	the object.

Returns

TRUE if the object is a Module.

5.47.3 Function Documentation

5.47.3.1 GldiModule∗ gldi_module_new (GldiVisitCard ∗ *pVisitCard*, GldiModuleInterface ∗ *pInterface*)

Create a new module. The module takes ownership of the 2 arguments, unless an error occured.

Parameters

pVisitCard	the visit card of the module
pInterface	the interface of the module

Returns

the new module, or NULL if the visit card is invalid.

5.47.3.2 GldiModule∗ gldi_module_new_from_so_file (const gchar ∗ *cSoFilePath*)

Create a new module from a .so file.

Parameters

cSoFilePath	path to the .so file

Returns

the new module, or NULL if an error occured.

5.47.3.3 void gldi_modules_new_from_directory (const gchar ∗ *cModuleDirPath*, GError ∗∗ *erreur*)

Create new modules from all the .so files contained in the given folder.

Parameters

cModuleDirPath	path to the folder
erreur	an error

Returns

the new module, or NULL if an error occured.

5.47.3.4 gchar∗ gldi_module_get_config_dir (GldiModule ∗ *pModule*)

Get the path to the folder containing the config files of a module (one file per instance). The folder is created if needed. If the module is not configurable, or if the folder couldn't be created, NULL is returned.

Parameters

pModule	the module

Returns

the path to the folder (free it after use).

5.47.3.5 GldiModule∗ gldi_module_get (const gchar ∗ *cModuleName*)

Get the module which has a given name.

Parameters

cModuleName	the unique name of the module.

Returns

the module, or NULL if not found.

5.47.3.6 void gldi_module_activate (GldiModule ∗ *module*)

Create and initialize all the instances of a module.

Parameters

module	the module to activate.

5.47.3.7 void gldi_module_deactivate (GldiModule ∗ *module*)

Stop and destroy all the instances of a module.

Parameters

module	the module to deactivate

5.48 cairo-dock-object.h File Reference

Data Structures

- struct _GldiObject

Definition of an Object.

- struct _GldiObjectManager

  Definition of an ObjectManager.

Macros

- #define GLDI_RUN_FIRST

  Use this in gldi_object_register_notification to be called before the core.
- #define GLDI_RUN_AFTER

  Use this in gldi_object_register_notification to be called after the core.
- #define GLDI_NOTIFICATION_INTERCEPT

  Return this in your callback to prevent the other callbacks from being called after you.
- #define GLDI_NOTIFICATION_LET_PASS

  Return this in your callback to let pass the notification to the other callbacks after you.
- #define gldi_object_notify(pObject, iNotifType,...)

Typedefs

- typedef gboolean(∗ GldiNotificationFunc)(gpointer pUserData,...)

  Generic prototype of a notification callback.

Enumerations

- enum GldiObjectNotifications {
  NOTIFICATION_NEW,
  NOTIFICATION_DESTROY }

  signals (any object has at least these ones)

Functions

- GldiObject ∗ gldi_object_new (GldiObjectManager ∗pMgr, gpointer attr)
- void gldi_object_ref (GldiObject ∗pObject)
- void gldi_object_unref (GldiObject ∗pObject)
- void gldi_object_delete (GldiObject ∗pObject)
- void gldi_object_reload (GldiObject ∗pObject, gboolean bReloadConfig)
- void gldi_object_register_notification (gpointer pObject, GldiNotificationType iNotifType, GldiNotificationFunc pFunction, gboolean bRunFirst, gpointer pUserData)
- void gldi_object_remove_notification (gpointer pObject, GldiNotificationType iNotifType, GldiNotificationFunc pFunction, gpointer pUserData)

5.48.1 Detailed Description

This class defines the Objects, the base class of libgldi. Every element in this library is an Object. An object is defined by an ObjectManager, which defines its capabilities and signals.

Any object is created with gldi_object_new and destroyed with gldi_object_unref. An object can be deleted from the current theme with gldi_object_delete. An object can be reloaded with gldi_object_reload.

You can listen for notifications on an object with gldi_object_register_notification and stop listening with gldi_↩object_remove_notification. To listen for notifications on any object of a given type, simply register yourself on its ObjectManager.

5.48.2 Macro Definition Documentation

5.48.2.1 #define gldi_object_notify(*pObject, iNotifType, ...*)

Broadcast a notification on a given object, and on all its managers.

Parameters

pObject	the object (Icon, Container, Manager, ...).
iNotifType	type of the notification.
...	parameters to be passed to the callbacks that have registered to this notification.

5.48.3 Enumeration Type Documentation

5.48.3.1 enum GldiObjectNotifications

signals (any object has at least these ones)

Enumerator

NOTIFICATION_NEW	notification called when an object has been created. data : the object
NOTIFICATION_DESTROY	notification called when the object is going to be destroyed. data : the object

5.48.4 Function Documentation

5.48.4.1 GldiObject∗ gldi_object_new (GldiObjectManager ∗ *pMgr*, gpointer *attr*)

Create a new object.

Parameters

pMgr	the ObjectManager
attr	the attributes of the object

Returns

the new object, with a reference of 1; use gldi_object_unref to destroy it

5.48.4.2 void gldi_object_ref (GldiObject ∗ *pObject*)

Take a reference on an object.

Parameters

pObject	the Object

5.48.4.3 void gldi_object_unref (GldiObject ∗ *pObject*)

Drop your reference on an object. If it's the last reference, the object is destroyed, otherwise nothing happens.

Parameters

pObject	the Object

5.48.4.4 void gldi_object_delete (GldiObject ∗ *pObject*)

Delete an object from the current theme. The object is unref'd, and won't be created again on next startup.

Parameters

pObject	the Object

5.48.4.5 void gldi_object_reload (GldiObject ∗ *pObject*, gboolean *bReloadConfig*)

Reload an object.

Parameters

pObject	the Object
bReloadConfig	TRUE to read its config file again (if the object has one)

5.48.4.6 void gldi_object_register_notification (gpointer *pObject*, GldiNotificationType *iNotifType*, GldiNotificationFunc *pFunction*, gboolean *bRunFirst*, gpointer *pUserData*)

Register an action to be called when a given notification is broadcasted from a given object.

Parameters

pObject	the object (Icon, Container, Manager).
iNotifType	type of the notification.
pFunction	callback.
bRunFirst	GLDI_RUN_FIRST to be called before Cairo-Dock, GLDI_RUN_AFTER to be called after.
pUserData	data to be passed as the first parameter of the callback.

5.48.4.7 void gldi_object_remove_notification (gpointer *pObject*, GldiNotificationType *iNotifType*, GldiNotificationFunc *pFunction*, gpointer *pUserData*)

Remove a callback from the list of callbacks of a given object for a given notification and a given data. Note: it is safe to remove the callback when it is called, but not another one.

Parameters

pObject	the object (Icon, Container, Manager) for which the action has been registered.
iNotifType	type of the notification.
pFunction	callback.
pUserData	data that was registerd with the callback.

5.49 cairo-dock-opengl-font.h File Reference

Data Structures

- struct _CairoDockGLFont

    Structure used to load a font for OpenGL text rendering.

Functions

- GLuint cairo_dock_create_texture_from_text_simple (const gchar ∗cText, const gchar ∗cFontDescription, cairo_t ∗pSourceContext, int ∗iWidth, int ∗iHeight)
- CairoDockGLFont ∗ cairo_dock_load_textured_font (const gchar ∗cFontDescription, int first, int count)
- CairoDockGLFont ∗ cairo_dock_load_textured_font_from_image (const gchar ∗cImagePath)
- void cairo_dock_free_gl_font (CairoDockGLFont ∗pFont)

- void cairo_dock_get_gl_text_extent (const gchar ∗cText, CairoDockGLFont ∗pFont, int ∗iWidth, int ∗iHeight)
- void cairo_dock_draw_gl_text (const guchar ∗cText, CairoDockGLFont ∗pFont)
- void cairo_dock_draw_gl_text_at_position (const guchar ∗cText, CairoDockGLFont ∗pFont, int x, int y)
- void cairo_dock_draw_gl_text_in_area (const guchar ∗cText, CairoDockGLFont ∗pFont, int iWidth, int iHeight, gboolean bCentered)
- void cairo_dock_draw_gl_text_at_position_in_area (const guchar ∗cText, CairoDockGLFont ∗pFont, int x, int y, int iWidth, int iHeight, gboolean bCentered)

5.49.1 Detailed Description

This class provides different ways to draw text directly in OpenGL. cairo_dock_create_texture_from_text_simple lets you draw any text in any font, by creating a texture from a Pango font description. This is a convenient function but not very fast. For a more efficient way, you load a font into a CairoDockGLFont with either : cairo_dock_load_↩
textured_font to load a subset of a Mono font into textures. You then use cairo_dock_draw_gl_text_at_position to draw the text.

5.49.2 Function Documentation

5.49.2.1 GLuint cairo_dock_create_texture_from_text_simple (const gchar ∗ *cText*, const gchar ∗ *cFontDescription*, cairo_t ∗ *pSourceContext*, int ∗ *iWidth*, int ∗ *iHeight*)

Create a texture from a text. The text is drawn in white, so that you can later colorize it with a mere glColor.

Parameters

cText	the text
cFontDescription	a description of the font, for instance "Monospace Bold 12"
pSourceContext	a cairo context, not altered by the function.
iWidth	a pointer that will be filled with the width of the texture.
iHeight	a pointer that will be filled with the height of the texture.

Returns

a newly allocated texture.

5.49.2.2 CairoDockGLFont∗ cairo_dock_load_textured_font (const gchar ∗ *cFontDescription*, int *first*, int *count*)

Load a font into textures. You can then render your text like a normal texture (zoom, etc). The drawback is that only a mono font can be used with this function.

Parameters

cFontDescription	a description of the font, for instance "Monospace Bold 12"
first	first character to load.
count	number of characters to load.

Returns

a newly allocated opengl font.

5.49.2.3 CairoDockGLFont∗ cairo_dock_load_textured_font_from_image (const gchar ∗ *cImagePath*)

Like the previous function, but loads the characters from an image. The image must be squared and contain the 256 extended ASCII characters in the alphabetic order.

Parameters

cImagePath	path to the image.

Returns

a newly allocated opengl font.

5.49.2.4 void cairo_dock_free_gl_font (CairoDockGLFont ∗ *pFont*)

Free an opengl font.

Parameters

pFont	the font.

5.49.2.5 void cairo_dock_get_gl_text_extent (const gchar ∗ *cText,* CairoDockGLFont ∗ *pFont,* int ∗ *iWidth,* int ∗ *iHeight*)

Compute the size a text will take for a given font.

Parameters

cText	the text
pFont	the font.
iWidth	a pointer that will be filled with the width of the text.
iHeight	a pointer that will be filled with the height of the text.

5.49.2.6 void cairo_dock_draw_gl_text (const guchar ∗ *cText,* CairoDockGLFont ∗ *pFont*)

Render a text for a given font. In the case of a bitmap font, the current raster position is used. In the case of a texture font, the current model view is used.

Parameters

cText	the text
pFont	the font.

5.49.2.7 void cairo_dock_draw_gl_text_at_position (const guchar ∗ *cText,* CairoDockGLFont ∗ *pFont,* int *x,* int *y*)

Like /ref cairo_dock_draw_gl_text but at a given position.

Parameters

cText	the text
pFont	the font.
x	x position of the left bottom corner of the text.
y	y position of the left bottom corner of the text.

5.49.2.8 void cairo_dock_draw_gl_text_in_area (const guchar ∗ *cText,* CairoDockGLFont ∗ *pFont,* int *iWidth,* int *iHeight,* gboolean *bCentered*)

Like /ref cairo_dock_draw_gl_text but resize the text so that it fits into a given area. Only works for a texture font.

Parameters

cText	the text
pFont	the font.
iWidth	iWidth of the area.
iHeight	iHeight of the area
bCentered	whether the text is centered on the current position or not.

5.49.2.9 void cairo_dock_draw_gl_text_at_position_in_area (const guchar ∗ *cText*, CairoDockGLFont ∗ *pFont*, int *x*, int *y*, int *iWidth*, int *iHeight*, gboolean *bCentered*)

Like /ref cairo_dock_draw_gl_text_in_area and /ref cairo_dock_draw_gl_text_at_position.

Parameters

cText	the text
pFont	the font.
x	x position of the left bottom corner of the text.
y	y position of the left bottom corner of the text.
iWidth	iWidth of the area.
iHeight	iHeight of the area
bCentered	whether the text is centered on the given position or not.

5.50 cairo-dock-opengl-path.h File Reference

Data Structures

- struct _CairoDockGLPath

    Definition of a CairoDockGLPath.

Functions

- CairoDockGLPath ∗ cairo_dock_new_gl_path (int iNbVertices, double x0, double y0, int iWidth, int iHeight)
- void cairo_dock_free_gl_path (CairoDockGLPath ∗pPath)
- void cairo_dock_gl_path_move_to (CairoDockGLPath ∗pPath, double x0, double y0)
- void cairo_dock_gl_path_set_extent (CairoDockGLPath ∗pPath, int iWidth, int iHeight)
- void cairo_dock_gl_path_line_to (CairoDockGLPath ∗pPath, GLfloat x, GLfloat y)
- void cairo_dock_gl_path_rel_line_to (CairoDockGLPath ∗pPath, GLfloat dx, GLfloat dy)
- void cairo_dock_gl_path_curve_to (CairoDockGLPath ∗pPath, int iNbPoints, GLfloat x1, GLfloat y1, GLfloat x2, GLfloat y2, GLfloat x3, GLfloat y3)
- void cairo_dock_gl_path_rel_curve_to (CairoDockGLPath ∗pPath, int iNbPoints, GLfloat dx1, GLfloat dy1, GLfloat dx2, GLfloat dy2, GLfloat dx3, GLfloat dy3)
- void cairo_dock_gl_path_simple_curve_to (CairoDockGLPath ∗pPath, int iNbPoints, GLfloat x1, GLfloat y1, GLfloat x2, GLfloat y2)
- void cairo_dock_gl_path_rel_simple_curve_to (CairoDockGLPath ∗pPath, int iNbPoints, GLfloat dx1, GLfloat dy1, GLfloat dx2, GLfloat dy2)
- void cairo_dock_gl_path_arc (CairoDockGLPath ∗pPath, int iNbPoints, GLfloat xc, GLfloat yc, double r, double teta0, double cone)
- void cairo_dock_stroke_gl_path (const CairoDockGLPath ∗pPath, gboolean bClosePath)
- void cairo_dock_fill_gl_path (const CairoDockGLPath ∗pPath, GLuint iTexture)
- void cairo_dock_draw_rounded_rectangle_opengl (double fFrameWidth, double fFrameHeight, double f←Radius, double fLineWidth, double ∗fLineColor)

5.50.1 Detailed Description

This class define OpenGL path, with similar functions as cairo. You create a path with cairo_dock_new_gl_path, then you add lines, curves or arcs to it. Once the path is defined, you can eigher stroke it with cairo_dock_stroke←_gl_path or fill it with cairo_dock_fill_gl_path. You can fill a path with the current color or with a texture, in this case you must provide the dimension of the husk. To destroy the path, use cairo_dock_free_gl_path.

5.50.2 Function Documentation

5.50.2.1 CairoDockGLPath∗ cairo_dock_new_gl_path (int *iNbVertices,* double *x0,* double *y0,* int *iWidth,* int *iHeight*)

Create a new path. It will start at the point (x0, y0). If you want to be abe to fill it with a texture, you can specify here the dimension of the path's husk.

Parameters

iNbVertices	maximum number of vertices the path will have
x0	x coordinate of the origin point
y0	y coordinate of the origin point
iWidth	width of the husk of the path.
iHeight	height of the husk of the path

Returns

a newly allocated path, with 1 point.

5.50.2.2 void cairo_dock_free_gl_path (CairoDockGLPath ∗ *pPath*)

Destroy a path and free its allocated ressources.

Parameters

pPath	the path.

5.50.2.3 void cairo_dock_gl_path_move_to (CairoDockGLPath ∗ *pPath,* double *x0,* double *y0*)

Rewind the path, defining its origin point. The path has only 1 point after a call to this function.

Parameters

pPath	the path.
x0	x coordinate of the origin point
y0	y coordinate of the origin point

5.50.2.4 void cairo_dock_gl_path_set_extent (CairoDockGLPath ∗ *pPath,* int *iWidth,* int *iHeight*)

Define the dimension of the hulk. This is needed if you intend to fill the path with a texture.

Parameters

pPath	the path.
iWidth	width of the hulk

iHeight	height of the hulk

5.50.2.5 void cairo_dock_gl_path_line_to (CairoDockGLPath ∗ *pPath,* GLfloat *x,* GLfloat *y*)

Add a line between the current point and a given point.

Parameters

pPath	the path.
x	x coordinate of the point
y	y coordinate of the point

5.50.2.6 void cairo_dock_gl_path_rel_line_to (CairoDockGLPath ∗ *pPath,* GLfloat *dx,* GLfloat *dy*)

Add a line defined relatively to the current point.

Parameters

pPath	the path.
dx	horizontal offset
dy	vertical offset

5.50.2.7 void cairo_dock_gl_path_curve_to (CairoDockGLPath ∗ *pPath,* int *iNbPoints,* GLfloat *x1,* GLfloat *y1,* GLfloat *x2,* GLfloat *y2,* GLfloat *x3,* GLfloat *y3*)

Add a Bezier cubic curve starting from the current point.

Parameters

pPath	the path.
iNbPoints	number of points used to discretize the curve
x1	first control point x
y1	first control point y
x2	second control point x
y2	second control point y
x3	terminal point of the curve x
y3	terminal point of the curve y

5.50.2.8 void cairo_dock_gl_path_rel_curve_to (CairoDockGLPath ∗ *pPath,* int *iNbPoints,* GLfloat *dx1,* GLfloat *dy1,* GLfloat *dx2,* GLfloat *dy2,* GLfloat *dx3,* GLfloat *dy3*)

Add a Bezier cubic curve starting from the current point. The control and terminal points are defined relatively to the current point.

Parameters

pPath	the path.
iNbPoints	number of points used to discretize the curve
dx1	first control point offset x
dy1	first control point offset y

dx2	second control point offset x
dy2	second control point offset y
dx3	terminal point of the curve offset x
dy3	terminal point of the curve offset y

5.50.2.9 void cairo_dock_gl_path_simple_curve_to (CairoDockGLPath ∗ *pPath*, int *iNbPoints*, GLfloat *x1*, GLfloat *y1*, GLfloat *x2*, GLfloat *y2*)

Add a Bezier bilinear curve starting from the current point

Parameters

pPath	the path.
iNbPoints	number of points used to discretize the curve
x1	control point x
y1	control point y
x2	terminal point of the curve x
y2	terminal point of the curve y

5.50.2.10 void cairo_dock_gl_path_rel_simple_curve_to (CairoDockGLPath ∗ *pPath*, int *iNbPoints*, GLfloat *dx1*, GLfloat *dy1*, GLfloat *dx2*, GLfloat *dy2*)

Add a Bezier bilinear curve starting from the current point. The control and terminal points are defined relatively to the current point.

Parameters

pPath	the path.
iNbPoints	number of points used to discretize the curve
dx1	control point offset x
dy1	control point offset y
dx2	terminal point of the curve offset x
dy2	terminal point of the curve offset y

5.50.2.11 void cairo_dock_gl_path_arc (CairoDockGLPath ∗ *pPath*, int *iNbPoints*, GLfloat *xc*, GLfloat *yc*, double *r*, double *teta0*, double *cone*)

Add an arc to the path, joining the current point to the beginning of the arc with a line.

Parameters

pPath	the path.
iNbPoints	number of points used to discretize the arc
xc	x coordinate of the center
yc	y coordinate of the center
r	radius
teta0	initial angle
cone	cone of the arc (a negative value means clockwise).

5.50.2.12 void cairo_dock_stroke_gl_path (const CairoDockGLPath ∗ *pPath*, gboolean *bClosePath*)

Stroke a path with the current color and with the current line width.

Parameters

pPath	the path.
bClosePath	whether to close the path (that is to say, join the last point with the first one) or not.

5.50.2.13 void cairo_dock_fill_gl_path (const CairoDockGLPath ∗ *pPath*, GLuint *iTexture*)

Fill a path with a texture, or with the current color if the texture is 0.

Parameters

pPath	the path.
iTexture	the texture, or 0 to fill the path with the current color. To fill the path with a gradation, use GL_COLOR_ARRAY and feed it with a table of colors that matches the vertices.

5.50.2.14 void cairo_dock_draw_rounded_rectangle_opengl (double *fFrameWidth*, double *fFrameHeight*, double *fRadius*, double *fLineWidth*, double ∗ *fLineColor*)

Draw a rectangle with rounded corners. The rectangle will be centered at the current point. The current matrix is not altered.

Parameters

fFrameWidth	width of the rectangle, without the corners.
fFrameHeight	height of the rectangle, including the corners.
fRadius	radius of the corners (can be 0).
fLineWidth	width of the line. If set to 0, the background will be filled with the provided color, otherwise the path will be stroke with this color.
fLineColor	color of the line if fLineWidth is non nul, or color of the background otherwise.

5.51 cairo-dock-opengl.h File Reference

Data Structures

- struct _CairoDockGLConfig

  This strucure summarizes the available OpenGL configuration on the system.

Macros

- #define gldi_gl_container_begin_draw(pContainer)

Functions

- gboolean gldi_gl_backend_init (gboolean bForceOpenGL)
- gboolean gldi_gl_container_make_current (GldiContainer ∗pContainer)
- gboolean gldi_gl_container_begin_draw_full (GldiContainer ∗pContainer, GdkRectangle ∗pArea, gboolean bClear)
- void gldi_gl_container_end_draw (GldiContainer ∗pContainer)
- void gldi_gl_container_set_perspective_view (GldiContainer ∗pContainer)
- void gldi_gl_container_set_perspective_view_for_icon (Icon ∗pIcon)
- void gldi_gl_container_set_ortho_view (GldiContainer ∗pContainer)
- void gldi_gl_container_set_ortho_view_for_icon (Icon ∗pIcon)
- void gldi_gl_container_init (GldiContainer ∗pContainer)

5.51.1 Detailed Description

This class manages the OpenGL backend and context.

5.51.2 Macro Definition Documentation

5.51.2.1 #define gldi_gl_container_begin_draw(*pContainer*)

Start drawing on a Container's OpenGL context (draw on the whole Container and clear buffers).

Parameters

pContainer	the container

5.51.3 Function Documentation

5.51.3.1 gboolean gldi_gl_backend_init (gboolean *bForceOpenGL*)

Initialize the OpenGL backend, by trying to get a suitable GLX configuration.

Parameters

bForceOpenGL	whether to force the use of OpenGL, or let the function decide.

Returns

TRUE if OpenGL is usable.

5.51.3.2 gboolean gldi_gl_container_make_current (GldiContainer ∗ *pContainer*)

Make a Container's OpenGL context the current one.

Parameters

pContainer	the container

Returns

TRUE if the Container's context is now the current one.

5.51.3.3 gboolean gldi_gl_container_begin_draw_full (GldiContainer ∗ *pContainer,* GdkRectangle ∗ *pArea,* gboolean *bClear*)

Start drawing on a Container's OpenGL context.

Parameters

pContainer	the container
pArea	optional area to clip the drawing (NULL to draw on the whole Container)
bClear	whether to clear the color buffer or not

5.51.3.4 void gldi_gl_container_end_draw (GldiContainer ∗ *pContainer*)

Ends the drawing on a Container's OpenGL context.

Parameters

pContainer	the container

5.51.3.5 void gldi_gl_container_set_perspective_view (GldiContainer ∗ *pContainer*)

Set a perspective view to the current GL context to fit a given Container. You may want to ensure the Container's context is really the current one.

Parameters

pContainer	the container

5.51.3.6 void gldi_gl_container_set_perspective_view_for_icon (Icon ∗ *pIcon*)

Set a perspective view to the current GL context to fit a given Icon (which must be inside a Container). You may want to ensure the Icon's Container's context is really the current one.

Parameters

pIcon	the icon

5.51.3.7 void gldi_gl_container_set_ortho_view (GldiContainer ∗ *pContainer*)

Set a orthogonal view to the current GL context to fit a given Container. You may want to ensure the Container's context is really the current one.

Parameters

pContainer	the container

5.51.3.8 void gldi_gl_container_set_ortho_view_for_icon (Icon ∗ *pIcon*)

Set a orthogonal view to the current GL context to fit a given Icon (which must be inside a Container). You may want to ensure the Icon's Container's context is really the current one.

Parameters

pIcon	the icon

5.51.3.9 void gldi_gl_container_init (GldiContainer ∗ *pContainer*)

Set a shared default-initialized GL context on a window.

Parameters

pContainer	the container, not yet realized.

5.52 cairo-dock-overlay.h File Reference

Data Structures

- struct _CairoOverlay
  Definition of an Icon Overlay.

Macros

- #define cairo_dock_set_overlay_scale(pOverlay, _fScale)
- #define cairo_dock_get_overlay_image_buffer(pOverlay)

Enumerations

- enum CairoOverlayPosition

  Available position of an overlay on an icon.

Functions

- CairoOverlay ∗ cairo_dock_add_overlay_from_image (Icon ∗pIcon, const gchar ∗cImageFile, CairoOverlay←Position iPosition, gpointer data)
- CairoOverlay ∗ cairo_dock_add_overlay_from_surface (Icon ∗pIcon, cairo_surface_t ∗pSurface, int iWidth, int iHeight, CairoOverlayPosition iPosition, gpointer data)
- CairoOverlay ∗ cairo_dock_add_overlay_from_texture (Icon ∗pIcon, GLuint iTexture, CairoOverlayPosition iPosition, gpointer data)
- void cairo_dock_remove_overlay_at_position (Icon ∗pIcon, CairoOverlayPosition iPosition, gpointer data)
- gboolean cairo_dock_print_overlay_on_icon_from_image (Icon ∗pIcon, const gchar ∗cImageFile, Cairo←OverlayPosition iPosition)
- void cairo_dock_print_overlay_on_icon_from_surface (Icon ∗pIcon, cairo_surface_t ∗pSurface, int iWidth, int iHeight, CairoOverlayPosition iPosition)

5.52.1 Detailed Description

This class defines Overlays, that are small images superimposed on the icon at a given position.

To add an overlay to an icon, use cairo_dock_add_overlay_from_image or cairo_dock_add_overlay_from_surface. The overlay can then be removed from the icon by simply destroying it with gldi_object_unref

A common feature is to have only 1 overlay at a given position. This can be achieved by passing a non-NULL data to the creation functions. This data will identify all of your overlays. You can then remove an overlay simply from its position with cairo_dock_remove_overlay_at_position, and adding an overlay at a position will automatically remove any previous overlay at this position with the same data.

If you're never going to update nor remove an overlay, you can choose to print it directly onto the icon with cairo←_dock_print_overlay_on_icon_from_image or cairo_dock_print_overlay_on_icon_from_surface, which is slightly faster.

Overlays are drawn at 1/2 of the icon size by default, but this can be set up with cairo_dock_set_overlay_scale. If you need to modify an overlay directly, you can get its image buffer with cairo_dock_get_overlay_image_buffer.

5.52.2 Macro Definition Documentation

5.52.2.1 #define cairo_dock_set_overlay_scale(*pOverlay,* *_fScale*)

Set the scale of an overlay; by default it's 0.5

Parameters

pOverlay	the overlay
_fScale	the scale

5.52.2.2 #define cairo_dock_get_overlay_image_buffer(*pOverlay*)

Get the image buffer of an overlay (only useful if you need to redraw the overlay).

Parameters

pOverlay	the overlay

5.52.3 Function Documentation

5.52.3.1 CairoOverlay∗ cairo_dock_add_overlay_from_image (Icon ∗ *pIcon,* const gchar ∗ *cImageFile,* CairoOverlayPosition *iPosition,* gpointer *data*)

Add an overlay on an icon from an image.

Parameters

pIcon	the icon
cImageFile	an image (if it's not a path, it is searched amongst the current theme's images)
iPosition	position where to display the overlay

Returns

the overlay, or NULL if the image couldn't be loaded.

Parameters

data	data that will be used to look for the overlay in cairo_dock_remove_overlay_at_position; if NULL, then this function can't be used

5.52.3.2 CairoOverlay∗ cairo_dock_add_overlay_from_surface (Icon ∗ *pIcon,* cairo_surface_t ∗ *pSurface,* int *iWidth,* int *iHeight,* CairoOverlayPosition *iPosition,* gpointer *data*)

Add an overlay on an icon from a surface.

Parameters

pIcon	the icon
pSurface	a cairo surface
iWidth	width of the surface
iHeight	height of the surface
iPosition	position where to display the overlay
data	data that will be used to look for the overlay in cairo_dock_remove_overlay_at_position; if NULL, then this function can't be used

Returns

the overlay.

5.52.3.3 CairoOverlay∗ cairo_dock_add_overlay_from_texture (Icon ∗ *pIcon,* GLuint *iTexture,* CairoOverlayPosition *iPosition,* gpointer *data*)

Add an overlay on an icon from a texture.

Parameters

pIcon	the icon

iTexture	a texture
iPosition	position where to display the overlay
data	data that will be used to look for the overlay in cairo_dock_remove_overlay_at_position; if NULL, then this function can't be used

Returns

the overlay.

5.52.3.4 void cairo_dock_remove_overlay_at_position (Icon ∗ *pIcon,* CairoOverlayPosition *iPosition,* gpointer *data*)

Remove an overlay from an icon, given its position and data.

Parameters

pIcon	the icon
iPosition	the position of the overlay
data	data that was set on the overlay when created; a NULL pointer is not valid.

5.52.3.5 gboolean cairo_dock_print_overlay_on_icon_from_image (Icon ∗ *pIcon,* const gchar ∗ *cImageFile,* CairoOverlayPosition *iPosition*)

Print an overlay onto an icon from an image at a given position. You can't remove/modify the overlay then. The overlay will be displayed until you modify the icon directly (for instance by setting a new image).

Parameters

pIcon	the icon
cImageFile	an image (if it's not a path, it is searched amongst the current theme's images)
iPosition	position where to display the overlay

Returns

TRUE if the overlay has been successfuly printed.

5.52.3.6 void cairo_dock_print_overlay_on_icon_from_surface (Icon ∗ *pIcon,* cairo_surface_t ∗ *pSurface,* int *iWidth,* int *iHeight,* CairoOverlayPosition *iPosition*)

Print an overlay onto an icon from a surface at a given position. You can't remove/modify the overlay then. The overlay will be displayed until you modify the icon directly (for instance by setting a new image).

Parameters

pIcon	the icon
pSurface	a cairo surface
iWidth	width of the surface
iHeight	height of the surface
iPosition	position where to display the overlay

Returns

TRUE if the overlay has been successfuly printed.

5.53 cairo-dock-packages.h File Reference

Data Structures

- struct _CairoDockPackage

  Definition of a generic package.

Macros

- #define cairo_dock_get_url_data(cURL, erreur)

Typedefs

- typedef void(∗ CairoDockGetPackagesFunc)(GHashTable ∗pPackagesTable, gpointer data)

  Prototype of the function called when the list of packages is available. Use g_hash_table_ref if you want to keep the table outside of this function.

Enumerations

- enum CairoDockPackageType {
  CAIRO_DOCK_LOCAL_PACKAGE,
  CAIRO_DOCK_USER_PACKAGE,
  CAIRO_DOCK_DISTANT_PACKAGE,
  CAIRO_DOCK_NEW_PACKAGE,
  CAIRO_DOCK_UPDATED_PACKAGE,
  CAIRO_DOCK_ANY_PACKAGE }

  Types of packagess.

Functions

- gboolean cairo_dock_download_file (const gchar ∗cURL, const gchar ∗cLocalPath)
- gchar ∗ cairo_dock_download_file_in_tmp (const gchar ∗cURL)
- gchar ∗ cairo_dock_download_archive (const gchar ∗cURL, const gchar ∗cExtractTo)
- GldiTask ∗ cairo_dock_download_file_async (const gchar ∗cURL, const gchar ∗cLocalPath, GFunc p↩ Callback, gpointer data)
- gchar ∗ cairo_dock_get_url_data_with_post (const gchar ∗cURL, gboolean bGetOutputHeaders, GError ∗∗erreur, const gchar ∗cFirstProperty,...)
- GldiTask ∗ cairo_dock_get_url_data_async (const gchar ∗cURL, GFunc pCallback, gpointer data)
- void cairo_dock_free_package (CairoDockPackage ∗pPackage)
- GHashTable ∗ cairo_dock_list_packages (const gchar ∗cSharePackagesDir, const gchar ∗cUserPackages↩ Dir, const gchar ∗cDistantPackagesDir, GHashTable ∗pTable)
- GldiTask ∗ cairo_dock_list_packages_async (const gchar ∗cSharePackagesDir, const gchar ∗cUser↩ PackagesDir, const gchar ∗cDistantPackagesDir, CairoDockGetPackagesFunc pCallback, gpointer data, GHashTable ∗pTable)
- gchar ∗ cairo_dock_get_package_path (const gchar ∗cPackageName, const gchar ∗cSharePackagesDir, const gchar ∗cUserPackagesDir, const gchar ∗cDistantPackagesDir, CairoDockPackageType iGivenType)

5.53.1 Detailed Description

This class provides a convenient way to deal with packages. A Package is a tarball (tar.gz) of a folder, located on a distant server, that can be installed locally. Packages are listed on the server in a file named "list.conf". It's a group-key file starting with "#!CD" on the first line; each package is described in its own group. Packages are stored on the server in a folder that has the same name, and contains the tarball, a "readme" file, and a "preview" file.

The class offers a high level of abstraction that allows to manipulate packages without having to care their location, version, etc. It also provides convenient utility functions to download a file or make a request to a server.

To get the list of available packages, use cairo_dock_list_packages, or its asynchronous version cairo_dock_list_↩ packages_async. To access a package, use cairo_dock_get_package_path.

5.53.2 Macro Definition Documentation

5.53.2.1 #define cairo_dock_get_url_data(*cURL,* *erreur*)

Retrieve the data of a distant URL.

Parameters

cURL	distant adress to get data from.
erreur	an error.

Returns

the data (NULL if failed). It's an array of chars, possibly containing nul chars. Free it after using.

5.53.3 Enumeration Type Documentation

5.53.3.1 enum **CairoDockPackageType**

Types of packagess.

Enumerator

CAIRO_DOCK_LOCAL_PACKAGE package installed as root on the machine (in a sub-folder /usr).

CAIRO_DOCK_USER_PACKAGE package located in the user's home

CAIRO_DOCK_DISTANT_PACKAGE package present on the server

CAIRO_DOCK_NEW_PACKAGE package newly present on the server (for less than 1 month)

CAIRO_DOCK_UPDATED_PACKAGE package present locally but with a more recent version on the server, or distant package that has been updated in the past month.

CAIRO_DOCK_ANY_PACKAGE joker (the search path function will search locally first, and on the server then).

5.53.4 Function Documentation

5.53.4.1 gboolean cairo_dock_download_file (const gchar ∗ *cURL,* const gchar ∗ *cLocalPath*)

Download a distant file into a given location.

Parameters

cURL	adress of the file.
cLocalPath	a local path where to store the file.

Returns

TRUE on success, else FALSE..

5.53.4.2 gchar∗ cairo_dock_download_file_in_tmp (const gchar ∗ *cURL*)

Download a distant file as a temporary file.

Parameters

cURL	adress of the file.

Returns

the local path of the file on success, else NULL. Free the string after using it.

5.53.4.3 gchar∗ cairo_dock_download_archive (const gchar ∗ *cURL*, const gchar ∗ *cExtractTo*)

Download an archive and extract it into a given folder.

Parameters

cURL	adress of the file.
cExtractTo	folder where to extract the archive (the archive is deleted then).

Returns

the local path of the file on success, else NULL. Free the string after using it.

5.53.4.4 GldiTask∗ cairo_dock_download_file_async (const gchar ∗ *cURL*, const gchar ∗ *cLocalPath*, GFunc *pCallback*, gpointer *data*)

Asynchronously download a distant file into a given location. This function is non-blocking, you'll get a CairoTask that you can discard at any time, and you'll get the path of the downloaded file as the first argument of the callback (the second being the data you passed to this function).

Parameters

cURL	adress of the file.
cLocalPath	a local path where to store the file, or NULL for a temporary file.
pCallback	function called when the download is finished. It takes the path of the downloaded file (it belongs to the task so don't free it) and the data you've set here.
data	data to be passed to the callback.

Returns

the Task that is doing the job. Keep it and use cairo_dock_discard_task whenever you want to discard the download (for instance if the user cancels it), or cairo_dock_free_task inside your callback.

5.53.4.5 gchar∗ cairo_dock_get_url_data_with_post (const gchar ∗ *cURL*, gboolean *bGetOutputHeaders*, GError ∗∗ *erreur*, const gchar ∗ *cFirstProperty*, ...)

Retrieve the response of a POST request to a server.

Parameters

cURL	the URL request
bGetOutput↩ *Headers*	whether to retrieve the page's header.

erreur	an error.
cFirstProperty	first property of the POST data.
...	tuples of property and data to insert in POST data; the POST data will be formed with a=urlencode(b)&c=urlencode(d)&... End it with NULL.

Returns

the data (NULL if failed). It's an array of chars, possibly containing nul chars. Free it after using.

5.53.4.6 GldiTask∗ cairo_dock_get_url_data_async (const gchar ∗ *cURL*, GFunc *pCallback*, gpointer *data*)

Asynchronously retrieve the content of a distant URL. This function is non-blocking, you'll get a CairoTask that you can discard at any time, and you'll get the content of the downloaded file as the first argument of the callback (the second being the data you passed to this function).

Parameters

cURL	distant adress to get data from.
pCallback	function called when the download is finished. It takes the content of the downloaded file (it belongs to the task so don't free it) and the data you've set here.
data	data to be passed to the callback.

Returns

the Task that is doing the job. Keep it and use cairo_dock_discard_task whenever you want to discard the download (for instance if the user cancels it), or cairo_dock_free_task inside your callback.

5.53.4.7 void cairo_dock_free_package (CairoDockPackage ∗ *pPackage*)

Destroy a package and free all its allocated memory.

Parameters

pPackage	the package.

5.53.4.8 GHashTable∗ cairo_dock_list_packages (const gchar ∗ *cSharePackagesDir*, const gchar ∗ *cUserPackagesDir*, const gchar ∗ *cDistantPackagesDir*, GHashTable ∗ *pTable*)

Get a list of packages from differente sources.

Parameters

cShare↔PackagesDir	path of a local folder containg packages or NULL.
cUser↔PackagesDir	path of a user folder containg packages or NULL.
cDistant↔PackagesDir	path of a distant folder containg packages or NULL.
pTable	a table of packages previously retrieved, or NULL.

Returns

a hash table of (name, _CairoDockPackage). Free it with g_hash_table_destroy when you're done with it.

5.53.4.9 GldiTask∗ cairo_dock_list_packages_async (const gchar ∗ *cSharePackagesDir,* const gchar ∗ *cUserPackagesDir,* const gchar ∗ *cDistantPackagesDir,* CairoDockGetPackagesFunc *pCallback,* gpointer *data,* GHashTable ∗ *pTable*)

Asynchronously get a list of packages from differente sources. This function is non-blocking, you'll get a CairoTask that you can discard at any time, and you'll get a hash-table of the packages as the first argument of the callback (the second being the data you passed to this function).

Parameters

cShare←PackagesDir	path of a local folder containg packages or NULL.
cUser←PackagesDir	path of a user folder containg packages or NULL.
cDistant←PackagesDir	path of a distant folder containg packages or NULL.
pCallback	function called when the listing is finished. It takes the hash-table of the found packages (it belongs to the task so don't free it) and the data you've set here.
data	data to be passed to the callback.
pTable	a table of packages previously retrieved, or NULL.

Returns

the Task that is doing the job. Keep it and use cairo_dock_discard_task whenever you want to discard the download (for instance if the user cancels it), or cairo_dock_free_task inside your callback.

5.53.4.10 gchar∗ cairo_dock_get_package_path (const gchar ∗ *cPackageName,* const gchar ∗ *cSharePackagesDir,* const gchar ∗ *cUserPackagesDir,* const gchar ∗ *cDistantPackagesDir,* CairoDockPackageType *iGivenType*)

Look for a package with a given name into differente sources. If the package is found on the server and is not present on the disk, or is not up to date, then it is downloaded and the local path is returned.

Parameters

cPackageName	name of the package.
cShare←PackagesDir	path of a local folder containing packages or NULL.
cUser←PackagesDir	path of a user folder containing packages or NULL.
cDistant←PackagesDir	path of a distant folder containg packages or NULL.
iGivenType	type of package, or CAIRO_DOCK_ANY_PACKAGE if any type of package should be considered.

Returns

a newly allocated string containing the complete local path of the package. If the package is distant, it is downloaded and extracted into this folder.

5.54 cairo-dock-particle-system.h File Reference

Data Structures

- struct _CairoParticle

  A particle of a particle system.
- struct _CairoParticleSystem

  A particle system.

Macros

- #define cairo_dock_render_particles(pParticleSystem)

Typedefs

- typedef struct _CairoParticle CairoParticle

    A particle of a particle system.
- typedef struct _CairoParticleSystem CairoParticleSystem

    A particle system.
- typedef void(CairoDockRewindParticleFunc)(CairoParticle ∗pParticle, double dt)

    Function that re-initializes a particle when its life is over.

Functions

- void cairo_dock_render_particles_full (CairoParticleSystem ∗pParticleSystem, int iDepth)
- CairoParticleSystem ∗ cairo_dock_create_particle_system (int iNbParticles, GLuint iTexture, double fWidth, double fHeight)
- void cairo_dock_free_particle_system (CairoParticleSystem ∗pParticleSystem)
- gboolean cairo_dock_update_default_particle_system (CairoParticleSystem ∗pParticleSystem, CairoDock←RewindParticleFunc pRewindParticle)

5.54.1 Detailed Description

A Particle System is a set of particles that evolve according to a given model. Each particle will see its parameters change with time : direction, speed, oscillation, color, size, etc. Particle Systems fully take advantage of OpenGL and are able to render many thousands of particles at a high frequency refresh.

5.54.2 Macro Definition Documentation

5.54.2.1 #define cairo_dock_render_particles(*pParticleSystem*)

Render all the particles of a particle system.

Parameters

pParticleSystem	the particle system.

5.54.3 Function Documentation

5.54.3.1 void cairo_dock_render_particles_full (CairoParticleSystem ∗ *pParticleSystem,* int *iDepth*)

Render all the particles of a particle system with a given depth.

Parameters

pParticleSystem	the particle system.
iDepth	depth of the particles that will be rendered. If set to -1, only particles with a negative z will be rendered, if set to 1, only particles with a positive z will be rendered, if set to 0, all the particles will be rendered.

5.54.3.2 CairoParticleSystem∗ cairo_dock_create_particle_system (int *iNbParticles,* GLuint *iTexture,* double *fWidth,* double *fHeight*)

Create a particle system.

Parameters

iNbParticles	number of particles of the system.
iTexture	texture to map on each particle.
fWidth	width of the system.
fHeight	height of the system.

Returns

a newly allocated particle system.

5.54.3.3 void cairo_dock_free_particle_system (CairoParticleSystem ∗ *pParticleSystem*)

Destroy a particle system, freeing all the ressources it was using.

Parameters

pParticleSystem	the particle system.

5.54.3.4 gboolean cairo_dock_update_default_particle_system (CairoParticleSystem ∗ *pParticleSystem*, CairoDockRewindParticleFunc *pRewindParticle*)

Update a particle system to the next step with a generic particle behavior model. You can write your own model depending on your needs.

Parameters

pParticleSystem	the particle system.
pRewindParticle	function called on a particle when its life is over.

Returns

TRUE if some particles are still alive.

5.55 cairo-dock-progressbar.h File Reference

Data Structures

- struct _CairoProgressBarAttribute

  Attributes of a PgrogressBar.

5.55.1 Detailed Description

This class defines the ProgressBar, which derives from the DataRenderer. All you need to know is the attributes that define a ProgressBar, the API to use is the common API for DataRenderer, defined in cairo-dock-data-renderer.h.

5.56 cairo-dock-separator-manager.h File Reference

Macros

- #define GLDI_OBJECT_IS_SEPARATOR_ICON(obj)

5.56.1 Detailed Description

This class handles the Separator Icons, which are user icons doing nothing.

5.56.2 Macro Definition Documentation

5.56.2.1 #define GLDI_OBJECT_IS_SEPARATOR_ICON(*obj*)

Say if an object is a SeparatorIcon.

Parameters

obj	the object.

Returns

TRUE if the object is a SeparatorIcon.

5.57 cairo-dock-stack-icon-manager.h File Reference

Macros

- #define GLDI_OBJECT_IS_STACK_ICON(obj)

5.57.1 Detailed Description

This class handles the Stack Icons, which are user icons pointing to a sub-dock.

5.57.2 Macro Definition Documentation

5.57.2.1 #define GLDI_OBJECT_IS_STACK_ICON(*obj*)

Say if an object is a StackIcon.

Parameters

obj	the object.

Returns

TRUE if the object is a StackIcon.

5.58 cairo-dock-style-facility.h File Reference

Data Structures

- struct _GldiTextDescription

  Description of the rendering of a text.

Macros

- #define GLDI_COLOR_SHADE_LIGHT

A light shade level (dock background, ...)
- #define GLDI_COLOR_SHADE_MEDIUM

  A medium shade level (selected menu-item, widget inside a dialog/menu, separator, ...)
- #define GLDI_COLOR_SHADE_STRONG

  A strong shade level (child widget inside a dialog/menu, ...)

Enumerations

- enum GldiStyleColors

  Available types of color.

Functions

- void gldi_style_color_shade (GldiColor ∗icolor, double shade, GldiColor ∗ocolor)

5.58.1 Detailed Description

This file provides a few functions dealing with style elements like colors and text.

5.58.2 Function Documentation

5.58.2.1 void gldi_style_color_shade (GldiColor ∗ *icolor,* double *shade,* GldiColor ∗ *ocolor*)

Shade a color, making it darker if it's light, and lighter if it's dark. Note that the opposite behavior can be obtained by passing a negative shade value. Alpha is copied unchanged. Both pointers can be the same.

Parameters

icolor	input color
shade	amount of light to add/remove, <= 1.
ocolor	output color

5.59 cairo-dock-style-manager.h File Reference

Macros

- #define gldi_style_colors_set_bg_color(pCairoContext)

Enumerations

- enum GldiStyleNotifications { NOTIFICATION_STYLE_CHANGED }

  signals

Functions

- void gldi_style_color_get (GldiStyleColors iColorType, GldiColor ∗pColor)
- void gldi_style_colors_set_bg_color_full (cairo_t ∗pCairoContext, gboolean bUseAlpha)
- void gldi_style_colors_set_selected_bg_color (cairo_t ∗pCairoContext)
- void gldi_style_colors_set_line_color (cairo_t ∗pCairoContext)
- void gldi_style_colors_set_text_color (cairo_t ∗pCairoContext)

- void gldi_style_colors_set_separator_color (cairo_t ∗pCairoContext)
- void gldi_style_colors_set_child_color (cairo_t ∗pCairoContext)
- void gldi_style_colors_paint_bg_color_with_alpha (cairo_t ∗pCairoContext, int iWidth, double fAlpha)

5.59.1 Detailed Description

This class defines the global style used by all widgets (Docks, Dialogs, Desklets, Menus, Icons). This includes background color, outline color, text color, linewidth, corner radius.

5.59.2 Macro Definition Documentation

5.59.2.1 #define gldi_style_colors_set_bg_color(*pCairoContext*)

Set the global background color on a context.

Parameters

pCairoContext	a context

5.59.3 Enumeration Type Documentation

5.59.3.1 enum GldiStyleNotifications

signals

Enumerator

NOTIFICATION_STYLE_CHANGED	notification called when the global style has changed

5.59.4 Function Documentation

5.59.4.1 void gldi_style_color_get (GldiStyleColors *iColorType*, GldiColor ∗ *pColor*)

Get the value of a color. In case the color is actually a pattern, it gives its dominant color. This function is really only useful when you need to have a color for sure (rather than potentially a pattern/texture), or when you need to apply the color with some transformation. Most of the time, you only want to use the gldi_style_colors_set_∗ functions.

Parameters

iColorType	type of the color
pColor	output color

5.59.4.2 void gldi_style_colors_set_bg_color_full (cairo_t ∗ *pCairoContext*, gboolean *bUseAlpha*)

Set the global background color on a context, with or without the alpha component.

Parameters

pCairoContext	a context
bUseAlpha	TRUE to use the alpha, FALSE to set it fully opaque

5.59.4.3 void gldi_style_colors_set_selected_bg_color (cairo_t ∗ *pCairoContext*)

Set the global selected color on a context.

Parameters

pCairoContext	a context

5.59.4.4 void gldi_style_colors_set_line_color (cairo_t ∗ *pCairoContext*)

Set the global line color on a context.

Parameters

pCairoContext	a context

5.59.4.5 void gldi_style_colors_set_text_color (cairo_t ∗ *pCairoContext*)

Set the global text color on a context.

Parameters

pCairoContext	a context

5.59.4.6 void gldi_style_colors_set_separator_color (cairo_t ∗ *pCairoContext*)

Set the global separator color on a context.

Parameters

pCairoContext	a context

5.59.4.7 void gldi_style_colors_set_child_color (cairo_t ∗ *pCairoContext*)

Set the global child color on a context.

Parameters

pCairoContext	a context

5.59.4.8 void gldi_style_colors_paint_bg_color_with_alpha (cairo_t ∗ *pCairoContext,* int *iWidth,* double *fAlpha*)

Paint a context with a horizontal alpha gradation. If the alpha is negative, the global style is used to find the alpha.

Parameters

pCairoContext	a context
iWidth	width of the gradation
fAlpha	alpha to use

5.60 cairo-dock-surface-factory.h File Reference

Macros

- #define CAIRO_DOCK_ORIENTATION_MASK

    mask to get the orientation from a CairoDockLoadImageModifier.
- #define cairo_dock_create_surface_for_square_icon(cImagePath, fImageSize)

- #define cairo_dock_create_surface_from_text(cText, pLabelDescription, iTextWidthPtr, iTextHeightPtr)

Enumerations

- enum CairoDockLoadImageModifier {
  CAIRO_DOCK_FILL_SPACE,
  CAIRO_DOCK_KEEP_RATIO,
  CAIRO_DOCK_DONT_ZOOM_IN,
  CAIRO_DOCK_ORIENTATION_HFLIP,
  CAIRO_DOCK_ORIENTATION_ROT_180,
  CAIRO_DOCK_ORIENTATION_VFLIP,
  CAIRO_DOCK_ORIENTATION_ROT_90_HFLIP,
  CAIRO_DOCK_ORIENTATION_ROT_90,
  CAIRO_DOCK_ORIENTATION_ROT_90_VFLIP,
  CAIRO_DOCK_ORIENTATION_ROT_270,
  CAIRO_DOCK_ANIMATED_IMAGE }

  Types of image loading modifiers.

Functions

- cairo_surface_t ∗ cairo_dock_create_surface_from_xicon_buffer (gulong ∗pXIconBuffer, int iBufferNb←Elements, int iWidth, int iHeight)
- cairo_surface_t ∗ cairo_dock_create_surface_from_pixbuf (GdkPixbuf ∗pixbuf, double fMaxScale, int iWidth←Constraint, int iHeightConstraint, CairoDockLoadImageModifier iLoadingModifier, double ∗fImageWidth, double ∗fImageHeight, double ∗fZoomX, double ∗fZoomY)
- cairo_surface_t ∗ cairo_dock_create_blank_surface (int iWidth, int iHeight)
- cairo_surface_t ∗ cairo_dock_create_surface_from_image (const gchar ∗cImagePath, double fMaxScale, int iWidthConstraint, int iHeightConstraint, CairoDockLoadImageModifier iLoadingModifier, double ∗fImage←Width, double ∗fImageHeight, double ∗fZoomX, double ∗fZoomY)
- cairo_surface_t ∗ cairo_dock_create_surface_from_image_simple (const gchar ∗cImageFile, double f←ImageWidth, double fImageHeight)
- cairo_surface_t ∗ cairo_dock_create_surface_from_icon (const gchar ∗cImagePath, double fImageWidth, double fImageHeight)
- cairo_surface_t ∗ cairo_dock_create_surface_from_pattern (const gchar ∗cImageFile, double fImageWidth, double fImageHeight, double fAlpha)
- cairo_surface_t ∗ cairo_dock_rotate_surface (cairo_surface_t ∗pSurface, double fImageWidth, double f←ImageHeight, double fRotationAngle)
- cairo_surface_t ∗ cairo_dock_create_surface_from_text_full (const gchar ∗cText, GldiTextDescription ∗p←LabelDescription, double fMaxScale, int iMaxWidth, int ∗iTextWidth, int ∗iTextHeight)
- cairo_surface_t ∗ cairo_dock_duplicate_surface (cairo_surface_t ∗pSurface, double fWidth, double fHeight, double fDesiredWidth, double fDesiredHeight)

5.60.1 Detailed Description

This class contains functions to load any image/X buffer/GdkPixbuf/text into a cairo-surface. The loading of an image can be modified by a mask, to take into account the ratio, zoom, orientation, etc.

The general way to load an image is by using cairo_dock_create_surface_from_image.

If you just want to load an image at a given size, use cairo_dock_create_surface_from_image_simple, or cairo_←dock_create_surface_from_icon.

To load a text into a surface, describe your text look with a _GldiTextDescription, and pass it to cairo_dock_create←_surface_from_text.

Note: if you also need to load the image into a texture, it's easier to use the higher level ImageBuffer API (see cairo_dock_create_image_buffer).

5.60.2 Macro Definition Documentation

5.60.2.1 #define cairo_dock_create_surface_for_square_icon(*cImagePath, fImageSize*)

Create a square surface from any image, at a given size. If the image is given by its sole name, it is searched inside the icons themes known by Cairo-Dock.

Parameters

cImagePath	path or name of an image.
fImageSize	the desired surface size.

Returns

the newly allocated surface.

5.60.2.2 #define cairo_dock_create_surface_from_text(*cText, pLabelDescription, iTextWidthPtr, iTextHeightPtr*)

Create a surface representing a text, according to a given text description.

Parameters

cText	the text.
pLabel↔ Description	description of the text rendering.
iTextWidthPtr	will be filled the width of the resulting surface.
iTextHeightPtr	will be filled the height of the resulting surface.

Returns

the newly allocated surface.

5.60.3 Enumeration Type Documentation

5.60.3.1 enum CairoDockLoadImageModifier

Types of image loading modifiers.

Enumerator

CAIRO_DOCK_FILL_SPACE fill the space, with transparency if necessary.

CAIRO_DOCK_KEEP_RATIO keep the ratio of the original image.

CAIRO_DOCK_DONT_ZOOM_IN don't zoom in the image if the final surface is larger than the original image.

CAIRO_DOCK_ORIENTATION_HFLIP orientation horizontal flip

CAIRO_DOCK_ORIENTATION_ROT_180 orientation 180° rotation

CAIRO_DOCK_ORIENTATION_VFLIP orientation vertical flip

CAIRO_DOCK_ORIENTATION_ROT_90_HFLIP orientation 90° rotation + horizontal flip

CAIRO_DOCK_ORIENTATION_ROT_90 orientation 90° rotation

CAIRO_DOCK_ORIENTATION_ROT_90_VFLIP orientation 90° rotation + vertical flip

CAIRO_DOCK_ORIENTATION_ROT_270 orientation 270° rotation

CAIRO_DOCK_ANIMATED_IMAGE load the image as a strip if possible.

5.60.4 Function Documentation

5.60.4.1 cairo_surface_t∗ cairo_dock_create_surface_from_xicon_buffer (gulong ∗ *pXIconBuffer,* int *iBufferNbElements,* int *iWidth,* int *iHeight*)

Create a surface from raw data of an X icon. The biggest icon possible is taken. The ratio is kept, and the surface will fill the space with transparency if necessary.

Parameters

pXIconBuffer	raw data of the icon.
iBufferNb←↩ Elements	number of elements in the buffer.
iWidth	will be filled with the resulting width of the surface.
iHeight	will be filled with the resulting height of the surface.

Returns

the newly allocated surface.

5.60.4.2 cairo_surface_t∗ cairo_dock_create_surface_from_pixbuf (GdkPixbuf ∗ *pixbuf,* double *fMaxScale,* int *iWidthConstraint,* int *iHeightConstraint,* CairoDockLoadImageModifier *iLoadingModifier,* double ∗ *fImageWidth,* double ∗ *fImageHeight,* double ∗ *fZoomX,* double ∗ *fZoomY*)

Create a surface from a GdkPixbuf.

Parameters

pixbuf	the pixbuf.
fMaxScale	maximum zoom of the icon.
iWidthConstraint	constraint on the width, or 0 to not constraint it.
iHeight←↩ Constraint	constraint on the height, or 0 to not constraint it.
iLoadingModifier	a mask of different loading modifiers.
fImageWidth	will be filled with the resulting width of the surface (hors zoom).
fImageHeight	will be filled with the resulting height of the surface (hors zoom).
fZoomX	if non NULL, will be filled with the zoom that has been applied on width.
fZoomY	if non NULL, will be filled with the zoom that has been applied on width.

Returns

the newly allocated surface.

5.60.4.3 cairo_surface_t∗ cairo_dock_create_blank_surface (int *iWidth,* int *iHeight*)

Create an empty surface (transparent) of a given size. In OpenGL mode, this surface can act as a buffer to generate a texture.

Parameters

iWidth	width of the surface.
iHeight	height of the surface.

Returns

the newly allocated surface.

5.60.4.4 cairo_surface_t∗ cairo_dock_create_surface_from_image (const gchar ∗ *cImagePath,* double *fMaxScale,* int *iWidthConstraint,* int *iHeightConstraint,* CairoDockLoadImageModifier *iLoadingModifier,* double ∗ *fImageWidth,* double ∗ *fImageHeight,* double ∗ *fZoomX,* double ∗ *fZoomY*)

Create a surface from any image.

Parameters

cImagePath	complete path to the image.
fMaxScale	maximum zoom of the icon.
iWidthConstraint	constraint on the width, or 0 to not constraint it.
iHeight←↩ Constraint	constraint on the height, or 0 to not constraint it.
iLoadingModifier	a mask of different loading modifiers.
fImageWidth	will be filled with the resulting width of the surface (hors zoom).
fImageHeight	will be filled with the resulting height of the surface (hors zoom).
fZoomX	if non NULL, will be filled with the zoom that has been applied on width.
fZoomY	if non NULL, will be filled with the zoom that has been applied on width.

Returns

the newly allocated surface.

5.60.4.5 cairo_surface_t∗ cairo_dock_create_surface_from_image_simple (const gchar ∗ *cImageFile,* double *fImageWidth,* double *fImageHeight*)

Create a surface from any image, at a given size. If the image is given by its sole name, it is searched inside the current theme root folder.

Parameters

cImageFile	path or name of an image.
fImageWidth	the desired surface width.
fImageHeight	the desired surface height.

Returns

the newly allocated surface.

5.60.4.6 cairo_surface_t∗ cairo_dock_create_surface_from_icon (const gchar ∗ *cImagePath,* double *fImageWidth,* double *fImageHeight*)

Create a surface from any image, at a given size. If the image is given by its sole name, it is searched inside the icons themes known by Cairo-Dock.

Parameters

cImagePath	path or name of an image.
fImageWidth	the desired surface width.
fImageHeight	the desired surface height.

Returns

the newly allocated surface.

5.60.4.7 cairo_surface_t∗ cairo_dock_create_surface_from_pattern (const gchar ∗ *cImageFile,* double *fImageWidth,* double *fImageHeight,* double *fAlpha*)

Create a surface at a given size, and fill it with a pattern. If the pattern image is given by its sole name, it is searched inside the current theme root folder.

Parameters

cImageFile	path or name of an image that will be repeated to fill the surface.
fImageWidth	the desired surface width.
fImageHeight	the desired surface height.
fAlpha	transparency of the pattern (1 means opaque).

Returns

the newly allocated surface.

5.60.4.8 cairo_surface_t∗ cairo_dock_rotate_surface (cairo_surface_t ∗ *pSurface,* double *fImageWidth,* double *fImageHeight,* double *fRotationAngle*)

Create a surface by rotating another. Only works for 1/4 of rounds.

Parameters

pSurface	surface to rotate.
fImageWidth	the width of the surface.
fImageHeight	the height of the surface.
fRotationAngle	rotation angle to apply, in radians.

Returns

the newly allocated surface.

5.60.4.9 cairo_surface_t∗ cairo_dock_create_surface_from_text_full (const gchar ∗ *cText,* GldiTextDescription ∗ *pLabelDescription,* double *fMaxScale,* int *iMaxWidth,* int ∗ *iTextWidth,* int ∗ *iTextHeight*)

Create a surface representing a text, according to a given text description.

Parameters

cText	the text.
pLabel←Description	description of the text rendering.
fMaxScale	maximum zoom of the text.
iMaxWidth	maximum authorized width for the surface; it will be zoomed in to fits this limit. 0 for no limit.
iTextWidth	will be filled the width of the resulting surface.
iTextHeight	will be filled the height of the resulting surface.

Returns

the newly allocated surface.

5.60.4.10 cairo_surface_t∗ cairo_dock_duplicate_surface (cairo_surface_t ∗ *pSurface,* double *fWidth,* double *fHeight,* double *fDesiredWidth,* double *fDesiredHeight*)

Create a surface identical to another, possibly resizing it.

Parameters

pSurface	surface to duplicate.
fWidth	the width of the surface.
fHeight	the height of the surface.
fDesiredWidth	desired width of the copy (0 to keep the same size).
fDesiredHeight	desired height of the copy (0 to keep the same size).

Returns

the newly allocated surface.

5.61 cairo-dock-task.h File Reference

Data Structures

- struct _GldiTask

  Definition of a periodic and/or asynchronous Task.

Macros

- #define gldi_task_new(iPeriod, get_data, update, pSharedMemory)
- #define gldi_task_get_elapsed_time(pTask)

Typedefs

- typedef void(∗ GldiGetDataAsyncFunc)(gpointer pSharedMemory)

  Definition of the asynchronous job, that does the heavy part.
- typedef gboolean(∗ GldiUpdateSyncFunc)(gpointer pSharedMemory)

  Definition of the synchronous job, that update the dock with the results of the previous job. Returns TRUE to continue, FALSE to stop.

Functions

- void gldi_task_launch (GldiTask ∗pTask)
- void gldi_task_launch_delayed (GldiTask ∗pTask, double fDelay)
- GldiTask ∗ gldi_task_new_full (int iPeriod, GldiGetDataAsyncFunc get_data, GldiUpdateSyncFunc update, GFreeFunc free_data, gpointer pSharedMemory)
- void gldi_task_stop (GldiTask ∗pTask)
- void gldi_task_discard (GldiTask ∗pTask)
- void gldi_task_free (GldiTask ∗pTask)
- gboolean gldi_task_is_active (GldiTask ∗pTask)
- gboolean gldi_task_is_running (GldiTask ∗pTask)
- void gldi_task_change_frequency (GldiTask ∗pTask, int iNewPeriod)
- void gldi_task_change_frequency_and_relaunch (GldiTask ∗pTask, int iNewPeriod)
- void gldi_task_downgrade_frequency (GldiTask ∗pTask)
- void gldi_task_set_normal_frequency (GldiTask ∗pTask)

5.61.1 Detailed Description

An easy way to define periodic and asynchronous tasks, that can perform heavy jobs without blocking the dock.

A Task is divided in 2 phases :

- the asynchronous phase will be executed in another thread, while the dock continues to run on its own thread, in parallel. During this phase you will do all the heavy job (like downloading a file or computing something) but you can't interact on the dock.
- the synchronous phase will be executed after the first one has finished. There you will update your applet with the result of the first phase.

Attention

A data buffer is used to communicate between the 2 phases. It is important that these datas are never accessed outside the task, and vice versa that the asynchronous thread never accesses other data than this buffer.
If you want to access these datas outside the task, you have to copy them in a safe place during the 2nd phase, or to stop the task before (beware that stopping the task means waiting for the 1st phase to finish, which can take some time).

You create a Task with gldi_task_new, launch it with gldi_task_launch, and destroy it with gldi_task_free or gldi_↩
task_discard.

A Task can be periodic if you specify a period, otherwise it will be executed once. It also can also be fully synchronous if you don't specify an asynchronous function.

5.61.2 Macro Definition Documentation

5.61.2.1 #define gldi_task_new(*iPeriod, get_data, update, pSharedMemory*)

Create a periodic Task.

Parameters

iPeriod	time between 2 iterations, possibly nul for a Task to be executed once only.
get_data	asynchonous function, which carries out the heavy job parallel to the dock; stores the results in the shared memory.
update	synchonous function, which carries out the update of the dock from the result of the previous function. Returns TRUE to continue, FALSE to stop.
pSharedMemory	structure passed as a parameter of the get_data and update functions. Must not be accessed outside of these functions !

Returns

the newly allocated Task, ready to be launched with gldi_task_launch. Free it with gldi_task_free or gldi_↩
task_discard.

5.61.2.2 #define gldi_task_get_elapsed_time(*pTask*)

Get the time elapsed since the last time the Task has run.

Parameters

pTask	the periodic Task.

5.61.3 Function Documentation

5.61.3.1 void gldi_task_launch (**GldiTask** ∗ *pTask*)

Launch a periodic Task, beforehand prepared with gldi_task_new. The first iteration is executed immediately. The frequency returns to its normal state.

Parameters

pTask	the periodic Task.

5.61.3.2 void gldi_task_launch_delayed (**GldiTask** ∗ *pTask,* double *fDelay*)

Same as above but after a delay. If the delay is 0, the task will be launched as soon as the main loop becomes idle.

Parameters

pTask	the periodic Task.
fDelay	delay in ms.

5.61.3.3 **GldiTask**∗ gldi_task_new_full (int *iPeriod,* **GldiGetDataAsyncFunc** *get_data,* **GldiUpdateSyncFunc** *update,* GFreeFunc *free_data,* gpointer *pSharedMemory*)

Create a periodic Task.

Parameters

iPeriod	time between 2 iterations, possibly nul for a Task to be executed once only.
get_data	asynchonous function, which carries out the heavy job parallel to the dock; stores the results in the shared memory.
update	synchonous function, which carries out the update of the dock from the result of the previous function. Returns TRUE to continue, FALSE to stop.
free_data	function called when the Task is destroyed, to free the shared memory (optionnal).
pSharedMemory	structure passed as a parameter of the get_data and update functions. Must not be accessed outside of these functions !

Returns

the newly allocated Task, ready to be launched with gldi_task_launch. Free it with gldi_task_free or gldi_↩
task_discard.

5.61.3.4 void gldi_task_stop (**GldiTask** ∗ *pTask*)

Stop a periodic Task. If the Task is running, it will wait until the asynchronous thread has finished, and skip the update. The Task can be launched again with a call to gldi_task_launch.

Parameters

pTask	the periodic Task.

5.61.3.5 void gldi_task_discard (**GldiTask** ∗ *pTask*)

Discard a periodic Task. The asynchronous thread will continue, and the Task will be freed when it ends. The Task should be considered as destroyed after a call to this function. This function can be used inside the 'update' callback to destroy the Task.

Parameters

pTask	the periodic Task.

5.61.3.6 void gldi_task_free (**GldiTask** ∗ *pTask*)

Stop and destroy a periodic Task, freeing all the allocated ressources. Unlike gldi_task_discard, the task is stopped before being freeed, so this is a blocking call. If you want to destroy the task inside the update callback, don't use this function; use gldi_task_discard instead.

Parameters

pTask	the periodic Task.

5.61.3.7 gboolean gldi_task_is_active (**GldiTask** ∗ *pTask*)

Tell if a Task is active, that is to say is periodically called.

Parameters

pTask	the periodic Task.

Returns

TRUE if the Task is active.

5.61.3.8 gboolean gldi_task_is_running (**GldiTask** ∗ *pTask*)

Tell if a Task is running, that is to say it is either in the thread or waiting for the update.

Parameters

pTask	the periodic Task.

Returns

TRUE if the Task is running.

5.61.3.9 void gldi_task_change_frequency (**GldiTask** ∗ *pTask,* int *iNewPeriod*)

Change the frequency of a Task. The next iteration is re-scheduled according to the new period.

Parameters

pTask	the periodic Task.
iNewPeriod	the new period between 2 iterations of the Task, in s.

5.61.3.10 void gldi_task_change_frequency_and_relaunch (**GldiTask** ∗ *pTask,* int *iNewPeriod*)

Change the frequency of a Task and relaunch it immediately. The next iteration is therefore immediately executed.

Parameters

pTask	the periodic Task.
iNewPeriod	the new period between 2 iterations of the Task, in s, or -1 to let it unchanged.

5.61.3.11 void gldi_task_downgrade_frequency (**GldiTask** ∗ *pTask*)

Downgrade the frequency of a Task. The Task will be executed less often (this is typically useful to put on stand-by a periodic measure).

Parameters

pTask	the periodic Task.

5.61.3.12 void gldi_task_set_normal_frequency (**GldiTask** ∗ *pTask*)

Set the frequency of the Task to its normal state. This is also done automatically when launching the Task.

Parameters

pTask	the periodic Task.

5.62 cairo-dock-themes-manager.h File Reference

Functions

- void cairo_dock_update_conf_file (const gchar ∗cConfFilePath, GType iFirstDataType,...)
- void cairo_dock_write_keys_to_conf_file (GKeyFile ∗pKeyFile, const gchar ∗cConfFilePath)
- gboolean cairo_dock_export_current_theme (const gchar ∗cNewThemeName, gboolean bSaveBehavior, gboolean bSaveLaunchers)
- gboolean cairo_dock_package_current_theme (const gchar ∗cThemeName, const gchar ∗cDirPath)
- gchar ∗ cairo_dock_depackage_theme (const gchar ∗cPackagePath)
- gboolean cairo_dock_delete_themes (gchar ∗∗cThemesList)
- gboolean cairo_dock_import_theme (const gchar ∗cThemeName, gboolean bLoadBehavior, gboolean b←LoadLaunchers)
- GldiTask ∗ cairo_dock_import_theme_async (const gchar ∗cThemeName, gboolean bLoadBehavior, gboolean bLoadLaunchers, GFunc pCallback, gpointer data)
- void cairo_dock_set_paths (gchar ∗cRootDataDirPath, gchar ∗cExtraDirPath, gchar ∗cThemesDirPath, gchar ∗cCurrentThemeDirPath, gchar ∗cLocalThemeDirPath, gchar ∗cDistantThemeDirName, gchar ∗cTheme←ServerAdress)

5.62.1 Detailed Description

This class defines the structure of the global theme (launchers, icons, plug-ins, configuration files, etc). It also provides methods to manage the themes, like exporting the current theme, importing new themes, deleting themes, etc.

5.62.2 Function Documentation

5.62.2.1 void cairo_dock_update_conf_file (const gchar ∗ *cConfFilePath*, GType *iFirstDataType*, ...)

Update a conf file with a list of values of the form : {type, name of the groupe, name of the key, value}. Must end with G_TYPE_INVALID.

Parameters

cConfFilePath	path to the conf file.
iFirstDataType	type of the first value.

5.62.2.2 void cairo_dock_write_keys_to_conf_file (GKeyFile ∗ *pKeyFile*, const gchar ∗ *cConfFilePath*)

Write a key file on the disk.

Parameters

pKeyFile	the key-file
cConfFilePath	its path on the disk

5.62.2.3 gboolean cairo_dock_export_current_theme (const gchar ∗ *cNewThemeName*, gboolean *bSaveBehavior*, gboolean *bSaveLaunchers*)

Export the current theme to a given name. Exported themes can be imported directly from the Theme Manager.

Parameters

cNewTheme←Name	name to export the theme to.
bSaveBehavior	whether to save the behavior parameters too.
bSaveLaunchers	whether to save the launchers too.

Returns

TRUE if the theme could be exported succefuly.

5.62.2.4 gboolean cairo_dock_package_current_theme (const gchar ∗ *cThemeName*, const gchar ∗ *cDirPath*)

Create a package of the current theme. Packages can be distributed easily, and imported into the dock by a mere drag and drop into the Theme Manager. The package is placed in the cDirPath directory (or $HOME if cDirPath is wrong).

Parameters

cThemeName	name of the package.

cDirPath	path to the directory

Returns

TRUE if the theme could be packaged succefuly.

5.62.2.5 gchar∗ cairo_dock_depackage_theme (const gchar ∗ *cPackagePath*)

Extract a package into the themes folder. Does not load it.

Parameters

cPackagePath	path of a package. If the package is distant, it is first downoladed.

Returns

the path of the theme folder, or NULL if anerror occured.

5.62.2.6 gboolean cairo_dock_delete_themes (gchar ∗∗ *cThemesList*)

Remove some exported themes from the hard-disk.

Parameters

cThemesList	a list of theme names, NULL-terminated.

Returns

TRUE if the themes has been succefuly deleted.

5.62.2.7 gboolean cairo_dock_import_theme (const gchar ∗ *cThemeName,* gboolean *bLoadBehavior,* gboolean *bLoadLaunchers*)

Import a theme, which can be : a local theme, a user theme, a distant theme, or even the path to a packaged theme.

Parameters

cThemeName	name of the theme to import.
bLoadBehavior	whether to import the behavior parameters too.
bLoadLaunchers	whether to import the launchers too.

Returns

TRUE if the theme could be imported succefuly.

5.62.2.8 GldiTask∗ cairo_dock_import_theme_async (const gchar ∗ *cThemeName,* gboolean *bLoadBehavior,* gboolean *bLoadLaunchers,* GFunc *pCallback,* gpointer *data*)

Asynchronously import a theme, which can be : a local theme, a user theme, a distant theme, or even the path to a packaged theme. This function is non-blocking, you'll get a CairoTask that you can discard at any time, and you'll get the result of the import as the first argument of the callback (the second being the data you passed to this function). Note that only downloading or unpacking the theme is done asynchronously, actually copying the files in the current theme folder is not (because it couldn't be cancelled without first making a backup).

Parameters

cThemeName	name of the theme to import.
bLoadBehavior	whether to import the behavior parameters too.
bLoadLaunchers	whether to import the launchers too.
pCallback	function called when the download is finished. It takes the result of the import (TRUE for a successful import) and the data you've set here.
data	data to be passed to the callback.

Returns

the Task that is doing the job. Keep it and use cairo_dock_discard_task if you want to discard the download before it's completed (for instance if the user cancels it), or cairo_dock_free_task inside your callback.

5.62.2.9 void cairo_dock_set_paths (gchar ∗ *cRootDataDirPath,* gchar ∗ *cExtraDirPath,* gchar ∗ *cThemesDirPath,* gchar ∗ *cCurrentThemeDirPath,* gchar ∗ *cLocalThemeDirPath,* gchar ∗ *cDistantThemeDirName,* gchar ∗ *cThemeServerAdress*)

Define the paths of themes. Do it just after 'gldi_init'.

Parameters

cRootDataDir←Path	path to the root folder of libgldi
cExtraDirPath	path to the extras themes (plug-in themes)
cThemesDirPath	path to the user themes
cCurrent←ThemeDirPath	path to the current theme
cLocalTheme←DirPath	path to the installed themes (default themes)
cDistant←ThemeDirName	folder of the themes on the server
cThemeServer←Adress	adress of the themes server

5.63 cairo-dock-user-icon-manager.h File Reference

Macros

- #define GLDI_OBJECT_IS_USER_ICON(obj)

5.63.1 Detailed Description

This class handles the User Icons. These are Icons belonging to the user (like launchers, stack-icons, separators), and that have a config file. The config file contains at least the dock the icon belongs to and the position inside the dock.

5.63.2 Macro Definition Documentation

5.63.2.1 #define GLDI_OBJECT_IS_USER_ICON(*obj*)

Say if an object is a UserIcon.

Parameters

obj	the object.

Returns

TRUE if the object is a UserIcon.

5.64 cairo-dock-utils.h File Reference

Functions

- gboolean cairo_dock_remove_version_from_string (gchar ∗cString)
- void cairo_dock_remove_html_spaces (gchar ∗cString)
- void cairo_dock_get_version_from_string (const gchar ∗cVersionString, int ∗iMajorVersion, int ∗iMinor↩ Version, int ∗iMicroVersion)
- gboolean cairo_dock_string_is_address (const gchar ∗cString)
- const gchar ∗ cairo_dock_get_default_terminal (void)
- gchar ∗ cairo_dock_get_command_with_right_terminal (const gchar ∗cCommand)

5.64.1 Detailed Description

Some helper functions.

5.64.2 Function Documentation

5.64.2.1 gboolean cairo_dock_remove_version_from_string (gchar ∗ *cString*)

Remove the version number from a string. Directly modifies the string.

Parameters

cString	a string.

Returns

TRUE if a version has been removed.

5.64.2.2 void cairo_dock_remove_html_spaces (gchar ∗ *cString*)

Replace the %20 by normal spaces into the string. The string is directly modified.

Parameters

cString	the string (it can't be a constant string)

5.64.2.3 void cairo_dock_get_version_from_string (const gchar ∗ *cVersionString,* int ∗ *iMajorVersion,* int ∗ *iMinorVersion,* int ∗ *iMicroVersion*)

Get the 3 version numbers of a string.

Parameters

cVersionString	the string of the form "x.y.z".
iMajorVersion	pointer to the major version.
iMinorVersion	pointer to the minor version.
iMicroVersion	pointer to the micro version.

5.64.2.4 gboolean cairo_dock_string_is_address (const gchar ∗ *cString*)

Say if a string is an adress (`file://xxx, http://xxx, ftp://xxx,` etc).

Parameters

cString	a string.

Returns

TRUE if it's an address.

5.64.2.5 const gchar∗ cairo_dock_get_default_terminal (void)

Get the command to launch the default terminal

5.64.2.6 gchar∗ cairo_dock_get_command_with_right_terminal (const gchar ∗ *cCommand*)

Get the command to launch another one from a terminal

Parameters

cCommand	command to launch from a terminal

5.65 cairo-dock-windows-manager.h File Reference

Data Structures

- struct _GldiWindowManagerBackend

  Definition of the Windows Manager backend.
- struct _GldiWindowActor

  Definition of a window actor.

Enumerations

- enum GldiWindowNotifications

  signals

Functions

- void gldi_windows_manager_register_backend (GldiWindowManagerBackend ∗pBackend)
- void gldi_windows_foreach (gboolean bOrderedByZ, GFunc callback, gpointer data)
- GldiWindowActor ∗ gldi_windows_find (gboolean(∗callback)(GldiWindowActor ∗, gpointer), gpointer data)
- GldiWindowActor ∗ gldi_windows_get_active (void)

5.65.1 Detailed Description

This class manages the windows actors and notifies for any change on them.

5.65.2 Function Documentation

5.65.2.1 void gldi_windows_manager_register_backend (GldiWindowManagerBackend ∗ *pBackend*)

Register a Window Manager backend. NULL functions are simply ignored.

Parameters

pBackend	a Window Manager backend

5.65.2.2 void gldi_windows_foreach (gboolean *bOrderedByZ,* GFunc *callback,* gpointer *data*)

Run a function on each window actor.

Parameters

bOrderedByZ	TRUE to sort by z-order, FALSE to sort by age
callback	the callback
data	user data

5.65.2.3 GldiWindowActor∗ gldi_windows_find (gboolean(∗)(GldiWindowActor ∗, gpointer) *callback,* gpointer *data*)

Run a function on each window actor.

Parameters

callback	the callback (takes the actor and the data, returns TRUE to stop)
data	user data

Returns

the found actor, or NULL

5.65.2.4 GldiWindowActor∗ gldi_windows_get_active (void)

Get the current active window actor.

Returns

the actor, or NULL if no window is currently active

5.66 gldi-icon-names.h File Reference

5.66.1 Detailed Description

This file lists the common named icons; these are generic icons that any icon-theme should provide, and they replace gtk-stock icons.

Index