

Cairo-Dock

3.0.2

Generated by Doxygen 1.8.1

Mon Jun 18 2012 20:14:11

Contents

1 Cairo-Dock's API documentation.	1
1.1 Introduction	2
1.2 Installation	2
1.3 Main structures	2
1.3.1 Containers	2
1.3.2 Icons	3
1.3.3 Dock	3
1.3.4 Desklet	3
1.3.5 Dialog	3
1.3.6 Flying Container	3
1.4 External Modules	3
1.4.1 First, what is a module ?	3
1.4.2 Let's start, how do I create an empty applet ?	3
1.4.3 Ok I have a generic applet, how do I define it ?	4
1.4.4 Great, I can see my applet in the dock ! Now, where should I continue ?	4
1.4.5 The notifications system.	5
1.4.6 Ok now I have several sections of code to fill. Are there any useful functions to do it ?	5
1.4.7 How can I take advantage of the OpenGL ?	6
1.4.8 How can I animate my applet to make it more lively ?	6
1.4.9 I have heavy treatments to do, how can I make them without slowing the dock ?	6
1.4.10 I need more than one icon, how can I easily get more ?	7
1.5 Advanced fonctionnalités	7
1.5.1 How can I make my own widgets in the config panel ?	7
1.5.2 How can my applet control the window of an application ?	7
1.5.3 How can I render some numerical values on my icon ?	7
1.5.4 How can I make my applet multi-instanciable ?	8
1.5.5 How can I draw anywhere on the dock, not only on my icon ?	8
2 Data Structure Index	9
2.1 Data Structures	9
3 File Index	11

3.1	File List	11
4	Data Structure Documentation	13
4.1	_CairoContainer Struct Reference	13
4.1.1	Detailed Description	14
4.2	_CairoDataRenderer Struct Reference	14
4.2.1	Detailed Description	15
4.3	_CairoDataRendererAttribute Struct Reference	15
4.3.1	Detailed Description	16
4.4	_CairoDataRendererInterface Struct Reference	16
4.4.1	Detailed Description	16
4.5	_CairoDesklet Struct Reference	16
4.5.1	Detailed Description	16
4.6	_CairoDeskletAttribute Struct Reference	17
4.6.1	Detailed Description	17
4.7	_CairoDeskletDecoration Struct Reference	17
4.7.1	Detailed Description	17
4.8	_CairoDeskletRenderer Struct Reference	17
4.8.1	Detailed Description	18
4.9	_CairoDialog Struct Reference	18
4.9.1	Detailed Description	18
4.10	_CairoDialogAttribute Struct Reference	18
4.10.1	Detailed Description	19
4.11	_CairoDialogDecorator Struct Reference	19
4.11.1	Detailed Description	19
4.12	_CairoDialogRenderer Struct Reference	19
4.12.1	Detailed Description	19
4.13	_CairoDock Struct Reference	19
4.13.1	Detailed Description	22
4.14	_CairoDockClassAppli Struct Reference	22
4.14.1	Detailed Description	22
4.15	_CairoDockDesktopBackground Struct Reference	22
4.15.1	Detailed Description	23
4.16	_CairoDockDesktopEnvBackend Struct Reference	23
4.16.1	Detailed Description	23
4.17	_CairoDockGLConfig Struct Reference	23
4.17.1	Detailed Description	23
4.18	_CairoDockGLFont Struct Reference	23
4.18.1	Detailed Description	23
4.19	_CairoDockGLPath Struct Reference	24

4.19.1 Detailed Description	24
4.20 _CairoDockGroupKeyWidget Struct Reference	24
4.20.1 Detailed Description	24
4.21 _CairoDockGuiBackend Struct Reference	24
4.21.1 Detailed Description	24
4.22 _CairoDockHidingEffect Struct Reference	24
4.22.1 Detailed Description	25
4.23 _CairoDockImageBuffer Struct Reference	25
4.23.1 Detailed Description	25
4.24 _CairoDockLabelDescription Struct Reference	25
4.24.1 Detailed Description	26
4.25 _CairoDockModule Struct Reference	26
4.25.1 Detailed Description	27
4.26 _CairoDockModuleInstance Struct Reference	27
4.26.1 Detailed Description	27
4.27 _CairoDockModuleInterface Struct Reference	27
4.27.1 Detailed Description	28
4.28 _CairoDockPackage Struct Reference	28
4.28.1 Detailed Description	28
4.29 _CairoDockRenderer Struct Reference	28
4.29.1 Detailed Description	29
4.30 _CairoDockTask Struct Reference	29
4.30.1 Detailed Description	30
4.31 _CairoDockTransition Struct Reference	30
4.31.1 Detailed Description	31
4.32 _CairoDockVisitCard Struct Reference	31
4.32.1 Detailed Description	31
4.33 _CairoDockWMBackend Struct Reference	31
4.33.1 Detailed Description	31
4.34 _CairoEmblem Struct Reference	31
4.34.1 Detailed Description	31
4.35 _CairoGraphAttribute Struct Reference	32
4.35.1 Detailed Description	32
4.36 _CairoIconContainerRenderer Struct Reference	32
4.36.1 Detailed Description	32
4.37 _CairoOverlay Struct Reference	32
4.37.1 Detailed Description	33
4.38 _CairoParticle Struct Reference	33
4.38.1 Detailed Description	34
4.39 _CairoParticleSystem Struct Reference	34

4.39.1	Detailed Description	34
4.40	_Icon Struct Reference	34
4.40.1	Detailed Description	35
4.41	_IconInterface Struct Reference	35
4.41.1	Detailed Description	35
5	File Documentation	37
5.1	cairo-dock-animations.h File Reference	37
5.1.1	Detailed Description	38
5.1.2	Macro Definition Documentation	38
5.1.2.1	cairo_dock_container_is_animating	38
5.1.2.2	cairo_dock_animation_will_be_visible	38
5.1.2.3	cairo_dock_stop_icon_animation	38
5.1.2.4	cairo_dock_get_animation_delta_t	38
5.1.2.5	cairo_dock_get_slow_animation_delta_t	39
5.1.2.6	cairo_dock_has_transition	39
5.1.2.7	cairo_dock_get_transition_count	39
5.1.2.8	cairo_dock_get_transition_elapsed_time	39
5.1.2.9	cairo_dock_get_transition_fraction	39
5.1.3	Function Documentation	40
5.1.3.1	cairo_dock_pop_up	40
5.1.3.2	cairo_dock_pop_down	40
5.1.3.3	cairo_dock_launch_animation	40
5.1.3.4	cairo_dock_start_icon_animation	40
5.1.3.5	cairo_dock_request_icon_animation	40
5.1.3.6	cairo_dock_trigger_icon_removal_from_dock	41
5.1.3.7	cairo_dock_set_transition_on_icon	41
5.1.3.8	cairo_dock_remove_transition_on_icon	41
5.2	cairo-dock-applet-canvas.h File Reference	41
5.2.1	Detailed Description	42
5.2.2	Macro Definition Documentation	43
5.2.2.1	CD_APPLET_DEFINE_ALL_BEGIN	43
5.2.2.2	CD_APPLET_DEFINE_END	43
5.2.2.3	CD_APPLET_DEFINITION	43
5.2.2.4	CD_APPLET_INIT_ALL_BEGIN	43
5.2.2.5	CD_APPLET_INIT_END	43
5.2.2.6	CD_APPLET_STOP_BEGIN	43
5.2.2.7	CD_APPLET_STOP_END	43
5.2.2.8	CD_APPLET_RELOAD_ALL_BEGIN	44
5.2.2.9	CD_APPLET_RELOAD_END	44

5.2.2.10	CD_APPLET_GET_CONFIG_ALL_BEGIN	44
5.2.2.11	CD_APPLET_GET_CONFIG_END	44
5.2.2.12	CD_APPLET_RESET_CONFIG_ALL_BEGIN	44
5.2.2.13	CD_APPLET_RESET_CONFIG_ALL_END	44
5.2.2.14	CD_APPLET_RESET_DATA_BEGIN	44
5.2.2.15	CD_APPLET_RESET_DATA_ALL_END	44
5.2.2.16	CD_APPLET_ON_CLICK_BEGIN	44
5.2.2.17	CD_APPLET_ON_CLICK_END	44
5.2.2.18	CD_APPLET_ON_BUILD_MENU_BEGIN	44
5.2.2.19	CD_APPLET_ON_BUILD_MENU_END	44
5.2.2.20	CD_APPLET_ON_MIDDLE_CLICK_BEGIN	45
5.2.2.21	CD_APPLET_ON_MIDDLE_CLICK_END	45
5.2.2.22	CD_APPLET_ON_DOUBLE_CLICK_BEGIN	45
5.2.2.23	CD_APPLET_ON_DOUBLE_CLICK_END	45
5.2.2.24	CD_APPLET_ON_DROP_DATA_BEGIN	45
5.2.2.25	CD_APPLET_ON_DROP_DATA_END	45
5.2.2.26	CD_APPLET_ON_SCROLL_BEGIN	45
5.2.2.27	CD_APPLET_ON_SCROLL_END	45
5.2.2.28	CD_APPLET_ON_UPDATE_ICON_BEGIN	45
5.2.2.29	CD_APPLET_ON_UPDATE_ICON_END	45
5.2.2.30	CD_APPLET_SKIP_UPDATE_ICON	45
5.2.2.31	CD_APPLET_STOP_UPDATE_ICON	45
5.2.2.32	CD_APPLET_PAUSE_UPDATE_ICON	46
5.2.2.33	CD_APPLET_REGISTER_FOR_CLICK_EVENT	46
5.2.2.34	CD_APPLET_UNREGISTER_FOR_CLICK_EVENT	46
5.2.2.35	CD_APPLET_REGISTER_FOR_BUILD_MENU_EVENT	46
5.2.2.36	CD_APPLET_UNREGISTER_FOR_BUILD_MENU_EVENT	46
5.2.2.37	CD_APPLET_REGISTER_FOR_MIDDLE_CLICK_EVENT	46
5.2.2.38	CD_APPLET_UNREGISTER_FOR_MIDDLE_CLICK_EVENT	46
5.2.2.39	CD_APPLET_REGISTER_FOR_DOUBLE_CLICK_EVENT	46
5.2.2.40	CD_APPLET_UNREGISTER_FOR_DOUBLE_CLICK_EVENT	46
5.2.2.41	CD_APPLET_REGISTER_FOR_DROP_DATA_EVENT	46
5.2.2.42	CD_APPLET_UNREGISTER_FOR_DROP_DATA_EVENT	46
5.2.2.43	CD_APPLET_REGISTER_FOR_SCROLL_EVENT	46
5.2.2.44	CD_APPLET_UNREGISTER_FOR_SCROLL_EVENT	47
5.2.2.45	CD_APPLET_REGISTER_FOR_UPDATE_ICON_SLOW_EVENT	47
5.2.2.46	CD_APPLET_UNREGISTER_FOR_UPDATE_ICON_SLOW_EVENT	47
5.2.2.47	CD_APPLET_REGISTER_FOR_UPDATE_ICON_EVENT	47
5.2.2.48	CD_APPLET_UNREGISTER_FOR_UPDATE_ICON_EVENT	47
5.3	cairo-dock-applet-facility.h File Reference	47

5.3.1	Detailed Description	49
5.3.2	Macro Definition Documentation	50
5.3.2.1	cairo_dock_set_icon_surface	50
5.3.2.2	CD_CONFIG_GET_BOOLEAN_WITH_DEFAULT	50
5.3.2.3	CD_CONFIG_GET_BOOLEAN	50
5.3.2.4	CD_CONFIG_GET_INTEGER_WITH_DEFAULT	50
5.3.2.5	CD_CONFIG_GET_INTEGER	51
5.3.2.6	CD_CONFIG_GET_DOUBLE_WITH_DEFAULT	51
5.3.2.7	CD_CONFIG_GET_DOUBLE	51
5.3.2.8	CD_CONFIG_GET_INTEGER_LIST	51
5.3.2.9	CD_CONFIG_GET_STRING_WITH_DEFAULT	52
5.3.2.10	CD_CONFIG_GET_STRING	52
5.3.2.11	CD_CONFIG_GET_FILE_PATH	52
5.3.2.12	CD_CONFIG_GET_STRING_LIST_WITH_DEFAULT	52
5.3.2.13	CD_CONFIG_GET_STRING_LIST	53
5.3.2.14	CD_CONFIG_GET_COLOR_WITH_DEFAULT	53
5.3.2.15	CD_CONFIG_GET_COLOR	53
5.3.2.16	CD_CONFIG_GET_COLOR_RVB_WITH_DEFAULT	53
5.3.2.17	CD_CONFIG_GET_COLOR_RVB	54
5.3.2.18	CD_CONFIG_GET_THEME_PATH	54
5.3.2.19	CD_CONFIG_GET_GAUGE_THEME	54
5.3.2.20	CD_CONFIG_RENAME_GROUP	54
5.3.2.21	CD_APPLET_ADD_SUB_MENU_WITH_IMAGE	54
5.3.2.22	CD_APPLET_ADD_SUB_MENU	55
5.3.2.23	CD_APPLET_ADD_IN_MENU_WITH_STOCK_AND_DATA	55
5.3.2.24	CD_APPLET_ADD_IN_MENU_WITH_DATA	55
5.3.2.25	CD_APPLET_ADD_IN_MENU	55
5.3.2.26	CD_APPLET_ADD_IN_MENU_WITH_STOCK	56
5.3.2.27	CD_APPLET_ADD_SEPARATOR_IN_MENU	56
5.3.2.28	CD_APPLET_ADD_FIRST_SEPARATOR_IN_MENU	56
5.3.2.29	CD_APPLET_POPUP_MENU_ON_MY_ICON	56
5.3.2.30	CD_APPLET_RELOAD_CONFIG_PANEL	56
5.3.2.31	CD_APPLET_RELOAD_CONFIG_PANEL_WITH_PAGE	56
5.3.2.32	CD_APPLET_MY_CONF_FILE	56
5.3.2.33	CD_APPLET_MY_KEY_FILE	56
5.3.2.34	CD_APPLET_MY_CONFIG_CHANGED	56
5.3.2.35	CD_APPLET_MY_CONTAINER_TYPE_CHANGED	57
5.3.2.36	CD_APPLET_MY_OLD_CONTAINER	57
5.3.2.37	CD_APPLET_CLICKED_ICON	57
5.3.2.38	CD_APPLET_CLICKED_CONTAINER	57

5.3.2.39	CD_APPLET_SHIFT_CLICK	57
5.3.2.40	CD_APPLET_CTRL_CLICK	57
5.3.2.41	CD_APPLET_ALT_CLICK	57
5.3.2.42	CD_APPLET_MY_MENU	57
5.3.2.43	CD_APPLET_RECEIVED_DATA	57
5.3.2.44	CD_APPLET_SCROLL_UP	57
5.3.2.45	CD_APPLET_SCROLL_DOWN	57
5.3.2.46	CD_APPLET_BIND_KEY	57
5.3.2.47	CD_APPLET_REDRAW_MY_ICON	58
5.3.2.48	CAIRO_DOCK_REDRAW_MY_CONTAINER	58
5.3.2.49	CD_APPLET_LOAD_SURFACE_FOR_MY_APPLET	58
5.3.2.50	CD_APPLET_LOAD_SURFACE_FOR_MY_APPLET_WITH_DEFAULT	58
5.3.2.51	CD_APPLET_SET_SURFACE_ON_MY_ICON	58
5.3.2.52	CD_APPLET_SET_SURFACE_ON_MY_ICON_WITH_ZOOM	59
5.3.2.53	CD_APPLET_SET_SURFACE_ON_MY_ICON_WITH_ALPHA	59
5.3.2.54	CD_APPLET_SET_SURFACE_ON_MY_ICON_WITH_BAR	59
5.3.2.55	CD_APPLET_SET_IMAGE_ON_MY_ICON	59
5.3.2.56	CD_APPLET_SET_USER_IMAGE_ON_MY_ICON	59
5.3.2.57	CD_APPLET_SET_DEFAULT_IMAGE_ON_MY_ICON_IF_NONE	60
5.3.2.58	CD_APPLET_SET_NAME_FOR_MY_ICON	60
5.3.2.59	CD_APPLET_SET_NAME_FOR_MY_ICON_PRINTF	60
5.3.2.60	CD_APPLET_SET_QUICK_INFO_ON_MY_ICON	60
5.3.2.61	CD_APPLET_SET_QUICK_INFO_ON_MY_ICON_PRINTF	60
5.3.2.62	CD_APPLET_SET_HOURS_MINUTES_AS_QUICK_INFO	60
5.3.2.63	CD_APPLET_SET_MINUTES_SECONDES_AS_QUICK_INFO	60
5.3.2.64	CD_APPLET_SET_SIZE_AS_QUICK_INFO	61
5.3.2.65	CD_APPLET_SET_STATIC_ICON	61
5.3.2.66	CD_APPLET_UNSET_STATIC_ICON	61
5.3.2.67	CD_APPLET_SET_ALWAYS_VISIBLE_ICON	61
5.3.2.68	CD_APPLET_ANIMATE_MY_ICON	61
5.3.2.69	CD_APPLET_STOP_ANIMATING_MY_ICON	61
5.3.2.70	CD_APPLET_DEMANDS_ATTENTION	61
5.3.2.71	CD_APPLET_STOP_DEMANDING_ATTENTION	62
5.3.2.72	CD_APPLET_GET_MY_ICON_EXTENT	62
5.3.2.73	CD_APPLET_START_DRAWING_MY_ICON	62
5.3.2.74	CD_APPLET_START_DRAWING_MY_ICON_OR_RETURN	62
5.3.2.75	CD_APPLET_FINISH_DRAWING_MY_ICON	62
5.3.2.76	CD_APPLET_ADD_OVERLAY_ON_MY_ICON	62
5.3.2.77	CD_APPLET_PRINT_OVERLAY_ON_MY_ICON	62
5.3.2.78	CD_APPLET_REMOVE_OVERLAY_ON_MY_ICON	63

5.3.2.79	CD_APPLET_MAKE_EMBLEM	63
5.3.2.80	CD_APPLET_DRAW_EMBLEM_ON_MY_ICON	63
5.3.2.81	CD_APPLET_ADD_DATA_RENDERER_ON_MY_ICON	63
5.3.2.82	CD_APPLET_RELOAD_MY_DATA_RENDERER	63
5.3.2.83	CD_APPLET_RENDER_NEW_DATA_ON_MY_ICON	64
5.3.2.84	CD_APPLET_REMOVE_MY_DATA_RENDERER	64
5.3.2.85	CD_APPLET_REFRESH_MY_DATA_RENDERER	64
5.3.2.86	CD_APPLET_SET_MY_DATA_RENDERER_HISTORY_TO_MAX	64
5.3.2.87	CD_APPLET_MY_CONTAINER_IS_OPENGL	64
5.3.2.88	CD_APPLET_SET_DESKLET_RENDERER_WITH_DATA	64
5.3.2.89	CD_APPLET_SET_DESKLET_RENDERER	64
5.3.2.90	CD_APPLET_SET_STATIC_DESKLET	64
5.3.2.91	CD_APPLET_ALLOW_NO_CLICKABLE_DESKLET	64
5.3.2.92	CD_APPLET_DELETE_MY_ICONS_LIST	65
5.3.2.93	CD_APPLET_REMOVE_ICON_FROM_MY_ICONS_LIST	65
5.3.2.94	CD_APPLET_DETACH_ICON_FROM_MY_ICONS_LIST	65
5.3.2.95	CD_APPLET_LOAD_MY_ICONS_LIST	65
5.3.2.96	CD_APPLET_ADD_ICON_IN_MY_ICONS_LIST	65
5.3.2.97	CD_APPLET_MY_ICONS_LIST	66
5.3.2.98	CD_APPLET_MY_ICONS_LIST_CONTAINER	66
5.3.2.99	CD_APPLET_MANAGE_APPLICATION	66
5.3.2.100	D_	66
5.3.3	Enumeration Type Documentation	66
5.3.3.1	CairoDockInfoDisplay	66
5.3.4	Function Documentation	66
5.3.4.1	cairo_dock_set_icon_surface_full	66
5.3.4.2	cairo_dock_draw_bar_on_icon	67
5.3.4.3	cairo_dock_set_icon_surface_with_reflect	67
5.3.4.4	cairo_dock_set_image_on_icon	67
5.3.4.5	cairo_dock_set_image_on_icon_with_default	67
5.3.4.6	cairo_dock_get_human_readable_size	68
5.3.4.7	cairo_dock_play_sound	68
5.4	cairo-dock-applications-manager.h File Reference	68
5.4.1	Detailed Description	68
5.4.2	Function Documentation	68
5.4.2.1	cairo_dock_start_applications_manager	68
5.4.2.2	cairo_dock_search_window_covering_dock	69
5.4.2.3	cairo_dock_search_window_overlapping_dock	69
5.4.2.4	cairo_dock_get_current_applis_list	69
5.4.2.5	cairo_dock_get_current_active_window	69

5.4.2.6	cairo_dock_get_current_active_icon	70
5.4.2.7	cairo_dock_get_icon_with_Xid	70
5.4.2.8	cairo_dock_foreach_applis	70
5.4.2.9	cairo_dock_foreach_applis_on_viewport	70
5.5	cairo-dock-class-manager.h File Reference	70
5.5.1	Detailed Description	71
5.5.2	Macro Definition Documentation	71
5.5.2.1	cairo_dock_register_class	71
5.5.3	Function Documentation	71
5.5.3.1	cairo_dock_set_data_from_class	71
5.6	cairo-dock-compiz-integration.h File Reference	71
5.6.1	Detailed Description	71
5.7	cairo-dock-config.h File Reference	71
5.7.1	Detailed Description	72
5.7.2	Macro Definition Documentation	72
5.7.2.1	cairo_dock_get_pango_weight_from_1_9	72
5.7.3	Function Documentation	72
5.7.3.1	cairo_dock_load_config	72
5.7.3.2	cairo_dock_is_loading	72
5.7.3.3	cairo_dock_get_version_from_string	72
5.7.3.4	cairo_dock_decrypt_string	73
5.7.3.5	cairo_dock_encrypt_string	73
5.7.3.6	cairo_dock_load_current_theme	73
5.8	cairo-dock-container.h File Reference	73
5.8.1	Detailed Description	74
5.8.2	Macro Definition Documentation	74
5.8.2.1	cairo_dock_init_container	75
5.8.2.2	cairo_dock_init_container_no_opengl	75
5.8.2.3	gldi_container_enable_drop	75
5.8.2.4	cairo_dock_get_max_scale	75
5.8.2.5	cairo_dock_popup_menu_on_container	75
5.8.3	Enumeration Type Documentation	76
5.8.3.1	CairoContainerNotifications	76
5.8.4	Function Documentation	76
5.8.4.1	cairo_dock_finish_container	76
5.8.4.2	cairo_dock_redraw_container	76
5.8.4.3	cairo_dock_redraw_container_area	77
5.8.4.4	cairo_dock_redraw_icon	77
5.8.4.5	cairo_dock_search_container_from_icon	77
5.8.4.6	cairo_dock_string_is_adress	77

5.8.4.7	cairo_dock_notify_drop_data	77
5.8.4.8	cairo_dock_popup_menu_on_icon	78
5.8.4.9	cairo_dock_add_in_menu_with_stock_and_data	78
5.8.4.10	cairo_dock_create_sub_menu	78
5.8.4.11	cairo_dock_build_menu	78
5.9	cairo-dock-core.h File Reference	79
5.9.1	Detailed Description	79
5.10	cairo-dock-data-renderer-manager.h File Reference	79
5.10.1	Detailed Description	79
5.10.2	Function Documentation	79
5.10.2.1	cairo_dock_get_default_data_renderer_font	79
5.11	cairo-dock-data-renderer.h File Reference	79
5.11.1	Detailed Description	80
5.11.2	Macro Definition Documentation	80
5.11.2.1	cairo_dock_get_icon_data_renderer	80
5.11.2.2	CAIRO_DATA_RENDERERER	80
5.11.2.3	cairo_data_renderer_get_data	81
5.11.2.4	CAIRO_DATA_RENDERERER_ATTRIBUTE	81
5.11.2.5	cairo_data_renderer_get_nb_values	81
5.11.2.6	cairo_data_renderer_get_min_value	81
5.11.2.7	cairo_data_renderer_get_max_value	82
5.11.2.8	cairo_data_renderer_get_value	82
5.11.2.9	cairo_data_renderer_get_current_value	82
5.11.2.10	cairo_data_renderer_get_previous_value	82
5.11.2.11	cairo_data_renderer_get_normalized_value	83
5.11.2.12	cairo_data_renderer_get_normalized_current_value	83
5.11.2.13	cairo_data_renderer_get_normalized_previous_value	83
5.11.2.14	cairo_data_renderer_get_normalized_current_value_with_latency	83
5.11.2.15	cairo_data_renderer_format_value_full	84
5.11.2.16	cairo_data_renderer_format_value	84
5.11.3	Function Documentation	84
5.11.3.1	cairo_dock_get_default_data_renderer_font	84
5.11.3.2	cairo_dock_add_new_data_renderer_on_icon	84
5.11.3.3	cairo_dock_render_new_data_on_icon	84
5.11.3.4	cairo_dock_remove_data_renderer_on_icon	85
5.11.3.5	cairo_dock_reload_data_renderer_on_icon	85
5.11.3.6	cairo_dock_resize_data_renderer_history	85
5.11.3.7	cairo_dock_refresh_data_renderer	85
5.12	cairo-dock-dbus.h File Reference	85
5.12.1	Detailed Description	86

5.12.2	Function Documentation	86
5.12.2.1	cairo_dock_get_session_connection	86
5.12.2.2	cairo_dock_register_service_name	86
5.12.2.3	cairo_dock_dbus_is_enabled	86
5.12.2.4	cairo_dock_create_new_session_proxy	86
5.12.2.5	cairo_dock_create_new_system_proxy	87
5.12.2.6	cairo_dock_dbus_detect_application	87
5.12.2.7	cairo_dock_dbus_detect_system_application	87
5.12.2.8	cairo_dock_dbus_get_boolean	87
5.12.2.9	cairo_dock_dbus_get_uinteger	88
5.12.2.10	cairo_dock_dbus_get_integer	88
5.12.2.11	cairo_dock_dbus_get_string	88
5.12.2.12	cairo_dock_dbus_get_string_list	88
5.12.2.13	cairo_dock_dbus_get_uchar	89
5.12.2.14	cairo_dock_dbus_call	89
5.13	cairo-dock-default-view.h File Reference	89
5.13.1	Detailed Description	89
5.14	cairo-dock-desklet-factory.h File Reference	89
5.14.1	Detailed Description	90
5.14.2	Macro Definition Documentation	91
5.14.2.1	CAIRO_DOCK_IS_DESKLET	91
5.14.2.2	CAIRO_DESKLET	91
5.14.2.3	cairo_dock_add_interactive_widget_to_desklet	91
5.14.3	Enumeration Type Documentation	91
5.14.3.1	CairoDeskletVisibility	91
5.14.4	Function Documentation	92
5.14.4.1	cairo_dock_new_desklet	92
5.14.4.2	cairo_dock_free_desklet	92
5.14.4.3	cairo_dock_configure_desklet	92
5.14.4.4	cairo_dock_add_interactive_widget_to_desklet_full	92
5.14.4.5	cairo_dock_set_desklet_margin	92
5.14.4.6	cairo_dock_steal_interactive_widget_from_desklet	93
5.14.4.7	cairo_dock_hide_desklet	93
5.14.4.8	cairo_dock_show_desklet	93
5.14.4.9	cairo_dock_zoom_out_desklet	93
5.14.4.10	cairo_dock_set_desklet_accessibility	93
5.14.4.11	cairo_dock_set_desklet_sticky	94
5.14.4.12	cairo_dock_lock_desklet_position	94
5.15	cairo-dock-desklet-manager.h File Reference	94
5.15.1	Detailed Description	94

5.15.2	Enumeration Type Documentation	95
5.15.2.1	CairoDeskletNotifications	95
5.15.3	Function Documentation	95
5.15.3.1	cairo_dock_create_desklet	95
5.15.3.2	cairo_dock_destroy_desklet	95
5.15.3.3	cairo_dock_foreach_desklet	96
5.15.3.4	cairo_dock_foreach_icons_in_desklets	96
5.15.3.5	cairo_dock_reload_desklets_decorations	96
5.15.3.6	cairo_dock_set_all_desklets_visible	96
5.15.3.7	cairo_dock_set_desklets_visibility_to_default	96
5.15.3.8	cairo_dock_get_desklet_by_Xid	97
5.15.3.9	cairo_dock_find_clicked_icon_in_desklet	97
5.16	cairo-dock-desktop-file-factory.h File Reference	97
5.16.1	Detailed Description	97
5.16.2	Function Documentation	97
5.16.2.1	cairo_dock_remove_html_spaces	97
5.16.2.2	cairo_dock_add_desktop_file_from_uri	98
5.16.2.3	cairo_dock_add_desktop_file_from_type	98
5.17	cairo-dock-dialog-factory.h File Reference	98
5.17.1	Detailed Description	99
5.17.2	Macro Definition Documentation	99
5.17.2.1	CAIRO_DOCK_IS_DIALOG	99
5.17.2.2	CAIRO_DIALOG	99
5.17.3	Function Documentation	100
5.17.3.1	cairo_dock_new_dialog	100
5.17.3.2	cairo_dock_free_dialog	100
5.17.3.3	cairo_dock_steal_interactive_widget_from_dialog	100
5.18	cairo-dock-dialog-manager.h File Reference	100
5.18.1	Detailed Description	101
5.18.2	Macro Definition Documentation	102
5.18.2.1	cairo_dock_remove_dialog_if_any	102
5.18.3	Function Documentation	102
5.18.3.1	cairo_dock_dialog_reference	102
5.18.3.2	cairo_dock_dialog_unreference	102
5.18.3.3	cairo_dock_remove_dialog_if_any_full	103
5.18.3.4	cairo_dock_build_dialog	103
5.18.3.5	cairo_dock_show_dialog_full	103
5.18.3.6	cairo_dock_show_temporary_dialog_with_icon_printf	103
5.18.3.7	cairo_dock_show_temporary_dialog_with_icon	104
5.18.3.8	cairo_dock_show_temporary_dialog	104

5.18.3.9	cairo_dock_show_temporary_dialog_with_default_icon	104
5.18.3.10	cairo_dock_show_dialog_with_question	105
5.18.3.11	cairo_dock_show_dialog_with_entry	105
5.18.3.12	cairo_dock_show_dialog_with_value	106
5.18.3.13	cairo_dock_show_dialog_and_wait	106
5.18.3.14	cairo_dock_show_demand_and_wait	106
5.18.3.15	cairo_dock_show_value_and_wait	107
5.18.3.16	cairo_dock_ask_question_and_wait	107
5.18.3.17	cairo_dock_icon_has_dialog	107
5.18.3.18	cairo_dock_get_dialogless_icon_full	107
5.18.3.19	cairo_dock_show_general_message	108
5.18.3.20	cairo_dock_ask_general_question_and_wait	108
5.18.3.21	cairo_dock_hide_dialog	108
5.18.3.22	cairo_dock_unhide_dialog	108
5.18.3.23	cairo_dock_toggle_dialog_visibility	109
5.19	cairo-dock-dock-facility.h File Reference	109
5.19.1	Detailed Description	109
5.19.2	Function Documentation	109
5.19.2.1	cairo_dock_update_dock_size	109
5.19.2.2	cairo_dock_calculate_dock_icons	109
5.19.2.3	cairo_dock_show_subdock	110
5.19.2.4	cairo_dock_calculate_icons_positions_at_rest_linear	110
5.19.2.5	cairo_dock_apply_wave_effect_linear	110
5.19.2.6	cairo_dock_get_current_dock_width_linear	110
5.19.2.7	cairo_dock_check_if_mouse_inside_linear	110
5.19.2.8	cairo_dock_check_can_drop_linear	111
5.19.2.9	cairo_dock_get_first_drawn_element_linear	111
5.20	cairo-dock-dock-factory.h File Reference	111
5.20.1	Detailed Description	112
5.20.2	Macro Definition Documentation	112
5.20.2.1	CAIRO_DOCK_IS_DOCK	112
5.20.2.2	CAIRO_DOCK	112
5.20.2.3	cairo_dock_insert_icon_in_dock	112
5.20.2.4	cairo_dock_remove_one_icon_from_dock	113
5.20.2.5	cairo_dock_remove_icon_from_dock	113
5.20.2.6	cairo_dock_add_new_launcher_by_uri	113
5.20.2.7	cairo_dock_add_new_launcher_by_type	113
5.20.3	Function Documentation	114
5.20.3.1	cairo_dock_insert_icon_in_dock_full	114
5.20.3.2	cairo_dock_detach_icon_from_dock_full	114

5.20.3.3	cairo_dock_remove_automatic_separators	114
5.20.3.4	cairo_dock_insert_automatic_separators_in_dock	114
5.20.3.5	cairo_dock_remove_icons_from_dock	115
5.21	cairo-dock-dock-manager.h File Reference	115
5.21.1	Detailed Description	116
5.21.2	Enumeration Type Documentation	116
5.21.2.1	CairoDocksNotifications	116
5.21.3	Function Documentation	116
5.21.3.1	cairo_dock_create_dock	116
5.21.3.2	cairo_dock_create_subdock	116
5.21.3.3	cairo_dock_destroy_dock	117
5.21.3.4	cairo_dock_search_dock_name	117
5.21.3.5	cairo_dock_get_readable_name_for_fock	117
5.21.3.6	cairo_dock_search_dock_from_name	117
5.21.3.7	cairo_dock_search_icon_pointing_on_dock	117
5.21.3.8	cairo_dock_rename_dock	118
5.21.3.9	cairo_dock_foreach_docks	118
5.21.3.10	cairo_dock_foreach_root_docks	118
5.21.3.11	cairo_dock_foreach_icons_in_docks	118
5.21.3.12	cairo_dock_hide_parent_dock	118
5.21.3.13	cairo_dock_hide_child_docks	118
5.21.3.14	cairo_dock_reload_buffers_in_all_docks	119
5.21.3.15	cairo_dock_set_all_views_to_default	119
5.21.3.16	cairo_dock_reload_one_root_dock	119
5.21.3.17	cairo_dock_remove_root_dock_config	119
5.21.3.18	cairo_dock_add_root_dock_config_for_name	119
5.21.3.19	cairo_dock_add_root_dock_config	119
5.21.3.20	cairo_dock_set_dock_visibility	120
5.22	cairo-dock-draw-opengl.h File Reference	120
5.22.1	Detailed Description	120
5.22.2	Macro Definition Documentation	120
5.22.2.1	cairo_dock_create_texture_from_image	120
5.22.2.2	_cairo_dock_delete_texture	121
5.22.2.3	_cairo_dock_enable_texture	121
5.22.2.4	_cairo_dock_disable_texture	121
5.22.2.5	_cairo_dock_set_alpha	121
5.22.2.6	_cairo_dock_set_blend_source	121
5.22.2.7	_cairo_dock_set_blend_alpha	121
5.22.2.8	_cairo_dock_set_blend_over	121
5.22.2.9	_cairo_dock_set_blend_pbuffer	121

5.22.2.10	_cairo_dock_apply_texture_at_size	121
5.22.2.11	_cairo_dock_apply_texture	122
5.22.2.12	_cairo_dock_apply_texture_at_size_with_alpha	122
5.22.3	Function Documentation	122
5.22.3.1	cairo_dock_render_one_icon_opengl	122
5.22.3.2	cairo_dock_create_texture_from_surface	122
5.22.3.3	cairo_dock_create_texture_from_raw_data	123
5.22.3.4	cairo_dock_create_texture_from_image_full	123
5.22.3.5	cairo_dock_update_icon_texture	123
5.22.3.6	cairo_dock_update_label_texture	123
5.22.3.7	cairo_dock_update_quick_info_texture	123
5.23	cairo-dock-draw.h File Reference	124
5.23.1	Detailed Description	124
5.23.2	Macro Definition Documentation	124
5.23.2.1	cairo_dock_erase_cairo_context	124
5.23.3	Function Documentation	124
5.23.3.1	cairo_dock_create_drawing_context_generic	124
5.23.3.2	cairo_dock_create_drawing_context_on_container	125
5.23.3.3	cairo_dock_create_drawing_context_on_area	125
5.23.3.4	cairo_dock_draw_rounded_rectangle	125
5.23.3.5	cairo_dock_draw_icon_cairo	125
5.23.3.6	cairo_dock_render_one_icon	126
5.23.3.7	cairo_dock_draw_string	126
5.24	cairo-dock-emblem.h File Reference	126
5.24.1	Detailed Description	127
5.24.2	Macro Definition Documentation	127
5.24.2.1	cairo_dock_set_emblem_position	127
5.24.3	Function Documentation	127
5.24.3.1	cairo_dock_make_emblem	127
5.24.3.2	cairo_dock_make_emblem_from_surface	127
5.24.3.3	cairo_dock_make_emblem_from_texture	128
5.24.3.4	cairo_dock_free_emblem	128
5.24.3.5	cairo_dock_draw_emblem_on_icon	128
5.25	cairo-dock-file-manager.h File Reference	128
5.25.1	Detailed Description	129
5.25.2	Function Documentation	129
5.25.2.1	cairo_dock_fm_register_vfs_backend	129
5.25.2.2	cairo_dock_fm_list_directory	130
5.25.2.3	cairo_dock_fm_measure_directory	130
5.25.2.4	cairo_dock_fm_get_file_info	130

5.25.2.5	cairo_dock_fm_get_file_properties	130
5.25.2.6	cairo_dock_fm_launch_uri	130
5.25.2.7	cairo_dock_fm_add_monitor_full	130
5.25.2.8	cairo_dock_fm_remove_monitor_full	130
5.25.2.9	cairo_dock_fm_mount_full	130
5.25.2.10	cairo_dock_fm_unmount_full	130
5.25.2.11	cairo_dock_fm_is_mounted	130
5.25.2.12	cairo_dock_fm_can_eject	130
5.25.2.13	cairo_dock_fm_eject_drive	131
5.25.2.14	cairo_dock_fm_delete_file	131
5.25.2.15	cairo_dock_fm_rename_file	131
5.25.2.16	cairo_dock_fm_move_file	131
5.25.2.17	cairo_dock_fm_create_file	131
5.25.2.18	cairo_dock_fm_list_apps_for_file	131
5.25.2.19	cairo_dock_fm_empty_trash	131
5.25.2.20	cairo_dock_fm_get_trash_path	131
5.25.2.21	cairo_dock_fm_get_desktop_path	131
5.25.2.22	cairo_dock_fm_logout	131
5.25.2.23	cairo_dock_fm_shutdown	131
5.25.2.24	cairo_dock_fm_reboot	131
5.25.2.25	cairo_dock_fm_lock_screen	132
5.25.2.26	cairo_dock_fm_setup_time	132
5.25.2.27	cairo_dock_fm_show_system_monitor	132
5.25.2.28	cairo_dock_fm_create_icon_from_URI	132
5.25.2.29	cairo_dock_get_file_size	132
5.26	cairo-dock-gauge.h File Reference	132
5.26.1	Detailed Description	132
5.27	cairo-dock-graph.h File Reference	132
5.27.1	Detailed Description	133
5.27.2	Enumeration Type Documentation	133
5.27.2.1	CairoDockTypeGraph	133
5.28	cairo-dock-gui-factory.h File Reference	133
5.28.1	Detailed Description	135
5.28.2	Enumeration Type Documentation	135
5.28.2.1	CairoDockGUIWidgetType	135
5.29	cairo-dock-gui-manager.h File Reference	137
5.29.1	Detailed Description	137
5.29.2	Macro Definition Documentation	137
5.29.2.1	cairo_dock_reload_current_module_widget	137
5.29.2.2	cairo_dock_build_generic_gui	138

5.29.3	Function Documentation	138
5.29.3.1	cairo_dock_get_group_key_widget_from_name	138
5.29.3.2	cairo_dock_get_widget_from_name	138
5.29.3.3	cairo_dock_set_status_message	139
5.29.3.4	cairo_dock_set_status_message_printf	139
5.29.3.5	cairo_dock_build_generic_gui_window	139
5.29.3.6	cairo_dock_reload_generic_gui	139
5.30	cairo-dock-hiding-effect.h File Reference	140
5.30.1	Detailed Description	140
5.31	cairo-dock-icon-container.h File Reference	140
5.31.1	Detailed Description	140
5.32	cairo-dock-icon-facility.h File Reference	140
5.32.1	Detailed Description	141
5.32.2	Macro Definition Documentation	141
5.32.2.1	cairo_dock_icon_is_being_inserted	141
5.32.2.2	cairo_dock_icon_is_being_removed	141
5.32.2.3	cairo_dock_get_icon_order	141
5.32.2.4	cairo_dock_get_next_element	141
5.32.2.5	cairo_dock_get_previous_element	141
5.32.2.6	cairo_dock_set_icon_static	142
5.32.2.7	cairo_dock_set_icon_always_visible	142
5.32.2.8	cairo_dock_remove_quick_info	142
5.32.3	Function Documentation	142
5.32.3.1	cairo_dock_get_icon_type	142
5.32.3.2	cairo_dock_compare_icons_order	142
5.32.3.3	cairo_dock_compare_icons_name	143
5.32.3.4	cairo_dock_compare_icons_extension	143
5.32.3.5	cairo_dock_sort_icons_by_order	143
5.32.3.6	cairo_dock_sort_icons_by_name	143
5.32.3.7	cairo_dock_get_first_icon	144
5.32.3.8	cairo_dock_get_last_icon	144
5.32.3.9	cairo_dock_get_first_icon_of_group	144
5.32.3.10	cairo_dock_get_last_icon_of_group	144
5.32.3.11	cairo_dock_get_first_icon_of_order	144
5.32.3.12	cairo_dock_get_last_icon_of_order	145
5.32.3.13	cairo_dock_get_pointed_icon	145
5.32.3.14	cairo_dock_get_next_icon	145
5.32.3.15	cairo_dock_get_previous_icon	145
5.32.3.16	cairo_dock_get_icon_with_command	146
5.32.3.17	cairo_dock_get_icon_with_base_uri	146

5.32.3.18	<code>cairo_dock_get_icon_with_name</code>	146
5.32.3.19	<code>cairo_dock_get_icon_with_subdock</code>	146
5.32.3.20	<code>cairo_dock_get_icon_with_module</code>	147
5.32.3.21	<code>cairo_dock_get_icon_extent</code>	147
5.32.3.22	<code>cairo_dock_get_current_icon_size</code>	147
5.32.3.23	<code>cairo_dock_compute_icon_area</code>	147
5.32.3.24	<code>cairo_dock_update_icon_s_container_name</code>	148
5.32.3.25	<code>cairo_dock_set_icon_name</code>	148
5.32.3.26	<code>cairo_dock_set_icon_name_printf</code>	148
5.32.3.27	<code>cairo_dock_set_quick_info</code>	148
5.32.3.28	<code>cairo_dock_set_quick_info_printf</code>	148
5.33	<code>cairo-dock-icon-factory.h</code> File Reference	149
5.33.1	Detailed Description	149
5.33.2	Macro Definition Documentation	150
5.33.2.1	<code>CAIRO_DOCK_IS_APPLI</code>	150
5.33.2.2	<code>CAIRO_DOCK_IS_APPLET</code>	150
5.33.2.3	<code>CAIRO_DOCK_IS_MULTI_APPLI</code>	150
5.33.2.4	<code>CAIRO_DOCK_IS_AUTOMATIC_SEPARATOR</code>	150
5.33.2.5	<code>CAIRO_DOCK_IS_USER_SEPARATOR</code>	150
5.33.2.6	<code>CAIRO_DOCK_IS_NORMAL_APPLI</code>	151
5.33.2.7	<code>CAIRO_DOCK_IS_DETACHABLE_APPLET</code>	151
5.33.3	Function Documentation	151
5.33.3.1	<code>cairo_dock_new_icon</code>	151
5.33.3.2	<code>cairo_dock_load_icon_image</code>	151
5.33.3.3	<code>cairo_dock_load_icon_text</code>	151
5.33.3.4	<code>cairo_dock_load_icon_quickinfo</code>	151
5.33.3.5	<code>cairo_dock_load_icon_buffers</code>	152
5.34	<code>cairo-dock-icon-manager.h</code> File Reference	152
5.34.1	Detailed Description	152
5.34.2	Enumeration Type Documentation	152
5.34.2.1	<code>CairoIconNotifications</code>	152
5.34.3	Function Documentation	153
5.34.3.1	<code>cairo_dock_free_icon</code>	153
5.34.3.2	<code>cairo_dock_foreach_icons</code>	153
5.34.3.3	<code>cairo_dock_search_icon_size</code>	153
5.34.3.4	<code>cairo_dock_search_icon_s_path</code>	153
5.35	<code>cairo-dock-image-buffer.h</code> File Reference	153
5.35.1	Detailed Description	154
5.35.2	Macro Definition Documentation	154
5.35.2.1	<code>cairo_dock_strings_differ</code>	154

5.35.2.2	cairo_dock_colors_rvb_differ	154
5.35.2.3	cairo_dock_colors_differ	154
5.35.2.4	cairo_dock_load_image_buffer	154
5.35.3	Function Documentation	155
5.35.3.1	cairo_dock_search_image_s_path	155
5.35.3.2	cairo_dock_load_image_buffer_full	155
5.35.3.3	cairo_dock_load_image_buffer_from_surface	155
5.35.3.4	cairo_dock_create_image_buffer	155
5.35.3.5	cairo_dock_unload_image_buffer	156
5.35.3.6	cairo_dock_free_image_buffer	156
5.36	cairo-dock-indicator-manager.h File Reference	156
5.36.1	Detailed Description	156
5.37	cairo-dock-keybinder.h File Reference	156
5.37.1	Detailed Description	157
5.37.2	Macro Definition Documentation	157
5.37.2.1	cd_keybinder_could_grab	157
5.37.3	Function Documentation	157
5.37.3.1	cd_keybinder_bind	157
5.37.3.2	cd_keybinder_unbind	157
5.37.3.3	cd_keybinder_rebind	158
5.37.3.4	cairo_dock_trigger_shortkey	158
5.38	cairo-dock-keyfile-utilities.h File Reference	158
5.38.1	Detailed Description	158
5.38.2	Function Documentation	159
5.38.2.1	cairo_dock_open_key_file	159
5.38.2.2	cairo_dock_write_keys_to_file	159
5.38.2.3	cairo_dock_merge_conf_files	159
5.38.2.4	cairo_dock_upgrade_conf_file_full	159
5.38.2.5	cairo_dock_get_conf_file_version	159
5.38.2.6	cairo_dock_conf_file_needs_update	159
5.38.2.7	cairo_dock_add_remove_element_to_key	159
5.38.2.8	cairo_dock_add_group_key_to_conf_file	160
5.38.2.9	cairo_dock_remove_group_key_from_conf_file	160
5.38.2.10	cairo_dock_update_conf_file	160
5.39	cairo-dock-kwin-integration.h File Reference	160
5.39.1	Detailed Description	160
5.40	cairo-dock-launcher-factory.h File Reference	160
5.40.1	Detailed Description	160
5.40.2	Function Documentation	160
5.40.2.1	cairo_dock_set_launcher_class	160

5.40.2.2	cairo_dock_load_icon_info_from_desktop_file	161
5.40.2.3	cairo_dock_new_launcher_icon	161
5.41	cairo-dock-launcher-manager.h File Reference	161
5.41.1	Detailed Description	161
5.41.2	Function Documentation	162
5.41.2.1	cairo_dock_create_icon_from_desktop_file	162
5.41.2.2	cairo_dock_create_dummy_launcher	162
5.41.2.3	cairo_dock_load_launchers_from_dir	162
5.41.2.4	cairo_dock_reload_launcher	162
5.42	cairo-dock-manager.h File Reference	163
5.42.1	Detailed Description	163
5.43	cairo-dock-module-factory.h File Reference	163
5.43.1	Detailed Description	163
5.43.2	Function Documentation	164
5.43.2.1	cairo_dock_deinstanciate_module	164
5.43.2.2	cairo_dock_reload_module_instance	164
5.43.2.3	cairo_dock_activate_module	164
5.43.2.4	cairo_dock_deactivate_module	164
5.43.2.5	cairo_dock_reload_module	164
5.44	cairo-dock-module-manager.h File Reference	165
5.44.1	Detailed Description	165
5.44.2	Function Documentation	165
5.44.2.1	cairo_dock_find_module_from_name	165
5.44.2.2	cairo_dock_load_module	165
5.44.2.3	cairo_dock_load_modules_in_directory	166
5.45	cairo-dock-notifications.h File Reference	166
5.45.1	Detailed Description	166
5.45.2	Macro Definition Documentation	166
5.45.2.1	cairo_dock_notify_on_object	166
5.45.3	Function Documentation	167
5.45.3.1	cairo_dock_register_notification_on_object	167
5.45.3.2	cairo_dock_remove_notification_func_on_object	167
5.46	cairo-dock-object.h File Reference	167
5.46.1	Detailed Description	167
5.46.2	Enumeration Type Documentation	167
5.46.2.1	CairoObjectNotifications	167
5.47	cairo-dock-opengl-font.h File Reference	167
5.47.1	Detailed Description	168
5.47.2	Function Documentation	168
5.47.2.1	cairo_dock_create_texture_from_text_simple	168

5.47.2.2	cairo_dock_load_bitmap_font	168
5.47.2.3	cairo_dock_load_textured_font	169
5.47.2.4	cairo_dock_load_textured_font_from_image	169
5.47.2.5	cairo_dock_free_gl_font	169
5.47.2.6	cairo_dock_get_gl_text_extent	169
5.47.2.7	cairo_dock_draw_gl_text	170
5.47.2.8	cairo_dock_draw_gl_text_at_position	170
5.47.2.9	cairo_dock_draw_gl_text_in_area	170
5.47.2.10	cairo_dock_draw_gl_text_at_position_in_area	170
5.48	cairo-dock-opengl-path.h File Reference	171
5.48.1	Detailed Description	171
5.48.2	Function Documentation	171
5.48.2.1	cairo_dock_new_gl_path	171
5.48.2.2	cairo_dock_free_gl_path	172
5.48.2.3	cairo_dock_gl_path_move_to	172
5.48.2.4	cairo_dock_gl_path_set_extent	172
5.48.2.5	cairo_dock_gl_path_line_to	172
5.48.2.6	cairo_dock_gl_path_rel_line_to	172
5.48.2.7	cairo_dock_gl_path_curve_to	173
5.48.2.8	cairo_dock_gl_path_rel_curve_to	173
5.48.2.9	cairo_dock_gl_path_simple_curve_to	173
5.48.2.10	cairo_dock_gl_path_rel_simple_curve_to	173
5.48.2.11	cairo_dock_gl_path_arc	174
5.48.2.12	cairo_dock_stroke_gl_path	174
5.48.2.13	cairo_dock_fill_gl_path	174
5.48.2.14	cairo_dock_draw_rounded_rectangle_opengl	174
5.49	cairo-dock-opengl.h File Reference	175
5.49.1	Detailed Description	175
5.49.2	Function Documentation	175
5.49.2.1	cairo_dock_initialize_opengl_backend	175
5.49.2.2	cairo_dock_create_icon_fbo	175
5.49.2.3	cairo_dock_destroy_icon_fbo	175
5.49.2.4	cairo_dock_begin_draw_icon	176
5.49.2.5	cairo_dock_end_draw_icon	176
5.49.2.6	cairo_dock_set_perspective_view	176
5.49.2.7	cairo_dock_set_ortho_view	176
5.49.2.8	gldi_glx_apply_desktop_background	176
5.49.2.9	gldi_glx_init_container	177
5.50	cairo-dock-overlay.h File Reference	177
5.50.1	Detailed Description	177

5.50.2	Function Documentation	178
5.50.2.1	cairo_dock_add_overlay_to_icon	178
5.50.2.2	cairo_dock_add_overlay_from_image	178
5.50.2.3	cairo_dock_add_overlay_from_surface	178
5.50.2.4	cairo_dock_add_overlay_from_texture	178
5.50.2.5	cairo_dock_remove_overlay_at_position	179
5.50.2.6	cairo_dock_get_overlay_buffer_at_position	179
5.50.2.7	cairo_dock_print_overlay_on_icon_from_image	179
5.50.2.8	cairo_dock_print_overlay_on_icon_from_surface	179
5.51	cairo-dock-packages.h File Reference	180
5.51.1	Detailed Description	181
5.51.2	Macro Definition Documentation	181
5.51.2.1	cairo_dock_get_url_data	181
5.51.3	Enumeration Type Documentation	181
5.51.3.1	CairoDockPackageType	181
5.51.4	Function Documentation	181
5.51.4.1	cairo_dock_download_file	181
5.51.4.2	cairo_dock_download_file_in_tmp	182
5.51.4.3	cairo_dock_download_archive	182
5.51.4.4	cairo_dock_download_file_async	182
5.51.4.5	cairo_dock_get_url_data_with_post	182
5.51.4.6	cairo_dock_get_url_data_async	183
5.51.4.7	cairo_dock_free_package	183
5.51.4.8	cairo_dock_list_packages	183
5.51.4.9	cairo_dock_list_packages_async	184
5.51.4.10	cairo_dock_get_package_path	184
5.52	cairo-dock-particle-system.h File Reference	184
5.52.1	Detailed Description	185
5.52.2	Macro Definition Documentation	185
5.52.2.1	cairo_dock_render_particles	185
5.52.3	Function Documentation	185
5.52.3.1	cairo_dock_render_particles_full	185
5.52.3.2	cairo_dock_create_particle_system	186
5.52.3.3	cairo_dock_free_particle_system	186
5.52.3.4	cairo_dock_update_default_particle_system	186
5.53	cairo-dock-surface-factory.h File Reference	186
5.53.1	Detailed Description	187
5.53.2	Macro Definition Documentation	188
5.53.2.1	cairo_dock_create_surface_for_square_icon	188
5.53.2.2	cairo_dock_create_surface_from_text	188

5.53.3	Enumeration Type Documentation	188
5.53.3.1	CairoDockLoadImageModifier	188
5.53.4	Function Documentation	189
5.53.4.1	cairo_dock_calculate_constrained_size	189
5.53.4.2	cairo_dock_create_surface_from_xicon_buffer	189
5.53.4.3	cairo_dock_create_surface_from_pixbuf	189
5.53.4.4	cairo_dock_create_blank_surface	190
5.53.4.5	cairo_dock_create_surface_from_image	190
5.53.4.6	cairo_dock_create_surface_from_image_simple	191
5.53.4.7	cairo_dock_create_surface_from_icon	191
5.53.4.8	cairo_dock_create_surface_from_pattern	191
5.53.4.9	cairo_dock_rotate_surface	191
5.53.4.10	cairo_dock_create_surface_from_text_full	192
5.53.4.11	cairo_dock_duplicate_surface	192
5.54	cairo-dock-task.h File Reference	192
5.54.1	Detailed Description	193
5.54.2	Macro Definition Documentation	194
5.54.2.1	cairo_dock_new_task	194
5.54.2.2	cairo_dock_get_task_elapsed_time	194
5.54.3	Function Documentation	194
5.54.3.1	cairo_dock_launch_task	194
5.54.3.2	cairo_dock_launch_task_delayed	195
5.54.3.3	cairo_dock_new_task_full	195
5.54.3.4	cairo_dock_stop_task	195
5.54.3.5	cairo_dock_discard_task	195
5.54.3.6	cairo_dock_free_task	195
5.54.3.7	cairo_dock_task_is_active	196
5.54.3.8	cairo_dock_task_is_running	196
5.54.3.9	cairo_dock_change_task_frequency	196
5.54.3.10	cairo_dock_relaunch_task_immediately	196
5.54.3.11	cairo_dock_downgrade_task_frequency	196
5.54.3.12	cairo_dock_set_normal_task_frequency	197
5.55	cairo-dock-themes-manager.h File Reference	197
5.55.1	Detailed Description	197
5.55.2	Function Documentation	197
5.55.2.1	cairo_dock_export_current_theme	197
5.55.2.2	cairo_dock_package_current_theme	198
5.55.2.3	cairo_dock_depackage_theme	198
5.55.2.4	cairo_dock_delete_themes	198
5.55.2.5	cairo_dock_import_theme	198

5.55.2.6	cairo_dock_import_theme_async	199
5.56	cairo-dock-X-manager.h File Reference	199
5.56.1	Detailed Description	200
5.56.2	Enumeration Type Documentation	200
5.56.2.1	CairoDesktopNotifications	200
5.56.3	Function Documentation	200
5.56.3.1	cairo_dock_wm_register_backend	200
5.56.3.2	cairo_dock_wm_present_class	200
5.56.3.3	cairo_dock_wm_present_windows	201
5.56.3.4	cairo_dock_wm_present_desktops	201
5.56.3.5	cairo_dock_wm_show_widget_layer	201
5.56.3.6	cairo_dock_wm_set_on_widget_layer	201
5.56.3.7	cairo_dock_get_current_desktop_and_viewport	201
5.57	cairo-dock-X-utilities.h File Reference	202
5.57.1	Detailed Description	202
5.57.2	Function Documentation	202
5.57.2.1	cairo_dock_remove_version_from_string	202

Chapter 1

Cairo-Dock's API documentation.

[Introduction](#)

[Installation](#)

[Main structures](#)

- [Containers](#)
- [Icons](#)
- [Dock](#)
- [Desklet](#)
- [Dialog](#)
- [Flying Container](#)

[External Modules](#)

- [First, what is a module ?](#)
- [Let's start, how do I create an empty applet ?](#)
- [Ok I have a generic applet, how do I define it ?](#)
- [Great, I can see my applet in the dock ! Now, where should I continue ?](#)
- [The notifications system.](#)
- [Ok now I have several sections of code to fill. Are there any useful functions to do it ?](#)
- [How can I take advantage of the OpenGL ?](#)
- [How can I animate my applet to make it more lively ?](#)
- [I have heavy treatments to do, how can I make them without slowing the dock ?](#)
- [I need more than one icon, how can I easily get more ?](#)

[Advanced fonctionnalités](#)

- [How can I make my own widgets in the config panel ?](#)
- [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 is divided into 3 parts :

- the definition of the main classes (dock, icon, etc)
- utilities functions (interaction with X, GUI, etc)
- plug-ins framework.

It is useful if you want to write a complex plug-in or add new features in the core (or if you just love C); to write simple applets in any language, see <http://doc.glx-dock.org>.

Cairo-Dock has a **decentralized conception** : it has a minimalistic core, and lets external modules extend its functionalities.

This is a strong design, because it allows to extend fonctionnalities easily without having to hack into the core, which makes the project more stable and allows developpers to use high-level functions only, that are very tested and optimized.

Thus, Cairo-Dock itself has no animation, but has a convenient notification system that allows external plug-ins to animate icons when they want.

The core itself is a library made of several modules. Each module is made on the same model:

- the "factory" defines the structures/enums/interfaces of a class and creation/modification/destruction functions
- the "manager" manages all the ressources of the module and all the instances of the class
- the "facility" or "utilities" are a collection of helper functions to related to the class.

In this document, we will focus on the notification system and the plug-ins framework. Part 1 will be seen briefly, and part 2 will be let to your curiosity. This should be enough to quickly be able to write a lot of applets.

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
bzip2 -d cairo-dock-core.tar.gz
### 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 ..
bzip2 -d cairo-dock-plug-ins.tar.gz
### compil the stable plug-ins and install them
cd cairo-dock-plug-ins
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 Containers

See [_CairoContainer](#) for the definition of a Container, and [cairo-dock-container.h](#) for a complete description of the Container class.

1.3.2 Icons

See [_Icon](#) for the definition of an Icon, and [cairo-dock-icon-factory.h](#) for a complete description of the Icon class.

1.3.3 Dock

See [_CairoDock](#) for the definition of a Dock, and [cairo-dock-dock-factory.h](#) for a complete description of the Dock class.

1.3.4 Desklet

See [_CairoDesklet](#) for the definition of a Desklet, and [cairo-dock-desklet.h](#) for a complete description of the Desklet class.

1.3.5 Dialog

See [_CairoDialog](#) for the definition of a Dialog, and [cairo-dock-dialogs.h](#) for a complete description of the Dialog class.

1.3.6 Flying Container

See [_CairoFlyingContainer](#) for the definition of a Flying Container, and [cairo-dock-flying-container.h](#) for a complete description of the FlyingContainer class.

1.4 External Modules

1.4.1 First, what is a module ?

Modules are compiled .so files (that is to say, library) that are plugged into the dock at run-time. Therefore, they can use any function used by the dock, and have a total interaction freedom on the dock. The advantage is that applets can do anything, in fact they are extensions of the dock itself. The drawback is that a buggy applet can make the dock unstable.

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 config/data.

Modules can be instantiated several times; each time they are, an **instance** is created. This instance will hold all the data used by the module: the icon and its container, the config structure and its conf file, the data structure and a slot to plug data into containers and icons. All these parameters are optional; a module that has an icon is also called an **applet**.

When instantiating a module, CD will check the presence of an "Icon" group in the conf file. If there is one, it will create an icon accordingly and insert it into its container. If there is a "Desklet" group, the module is considered as detachable, and can be placed into a desklet. Here we will focus on applets, that is to say, we will have an icon and a container (dock or desklet).

1.4.2 Let's start, how do I create an empty applet ?

Easy ! just go to the "plug-ins" folder, and run the *generate-applet.sh* script. Answer the few questions, and you're done ! Don't forget to install the plug-in each time you modify it (*sudo make install* in your applet's folder). You can see that the script has created for you the architecture of your applet :

- in the **plug-ins** parent folder, you have the "CMakeLists.txt", where you can set the version number of your applet, the dependencies, etc
- in the **src** folder, you have the sources of your applet. It is common to put the init/stop/reload in `applet-init.c`, the `get_config/reset_config/reset_data` in `applet-config.c`, the notifications in `applet-notifications.c`, and the structures in `applet-struct.h`. Of course, you can add as many files as you want, just don't forget to specify them in the `CMakeLists.txt`.
- in the **data** folder, you have the config file, the default icon, and a preview. You will have to choose a default icon that fits your applet, and make a preview that makes users want to try it ;-). If you have other files to install, it's here you will do it. If you change the name of the default icon (for instance you use an SVG file), don't forget to modify the `data/CMakeLists.txt` and also the `src/CMakeLists.txt`.

1.4.3 Ok I have a generic applet, how do I define it ?

As we saw, a module must fill a visit card and an interface, to be accepted by the dock. This is done very easily by the `CD_APPLET_DEFINITION` macro. All you have to give is the name of the applet, its category, a brief description/manual (very important !), and your name.

Once you have finished your applet, don't forget to make a nice preview (~200x200 pixels) and a nice default icon, and place them in the `data` folder.

1.4.4 Great, I can see my applet in the dock ! Now, where should I continue ?

We saw that when our applet is activated, an instance is created. It is called **myApplet**, and it will hold the following :

- **myIcon** : this is your icon ! It will act as a drawing surface to represent whatever you want.
- **myDrawContext** : a cairo context, to draw on your icon with the `libcairo`.
- **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 conf file. You have to define it in `applet-struct.h`.
- **myData** : the structure holding all the resources loaded at run-time. You have to define it in `applet-struct.h`.

The framework defines different **sections**, and all you have to do is to fill them :

- First of all you will have to get your config parameters. This is done in the `CD_APPLET_GET_CONFIG_BEGIN/CD_APPLET_GET_CONFIG_END` section, in `applet-config.c`.
- Each time you add a parameter, think of freeing it if it's a dynamic resource like a string; this is done in the `CD_APPLET_RESET_CONFIG_BEGIN/CD_APPLET_RESET_CONFIG_END` section.
- In a similar way, you will free all the resources you allocated by `myData` in the `CD_APPLET_RESET_DATA_BEGIN/CD_APPLET_RESET_DATA_END` section.
- After the instance is created, the dock lets you start. This is done in the `CD_APPLET_INIT_BEGIN/CD_APPLET_INIT_END` section. At this point, `myApplet` is already fully defined, and `myConfig` has been filled. Therefore you can already draw on your icon, launch timers, register to notifications, etc.
- Each time the user changes something in its config, or the desklet is resized, your applet is reloaded. This is done in the `CD_APPLET_RELOAD_BEGIN/CD_APPLET_RELOAD_END` section. The macro `CD_APPLET_MY_CONFIG_CHANGED` tells you if something has changed in your config or if it's just a resizing.
- Last, when your applet is stopped, you have to stop everything you set up in the init (timers, notifications, etc) in the `CD_APPLET_STOP_BEGIN/CD_APPLET_STOP_END` section.

1.4.5 The notifications system.

When something happens, Cairo-Dock notifies everybody about it, including itself. An applet can register to any notification (see [cairo-dock-notifications.h](#)) before or after the dock, to be notified of the event of its choice. When you are notified, the function you registered for this event will be called; it must match the notification prototype as defined in [cairo-dock-notifications.h](#).

For instance if you want to know when the user clicks on your icon, you will register to the [NOTIFICATION_CLICK_ICON](#) notification.

To register to a notification, you have the [cairo_dock_register_notification_on_object](#) function. Always unregister when your applet is stopped, to avoid being notified when you shouldn't, with the function [cairo_dock_remove_notification_func_on_object](#).

For convenience, there are macros to register to the most common events:

- [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](#)

Then you just have to fill the corresponding sections:

- [CD_APPLET_ON_CLICK_BEGIN/CD_APPLET_ON_CLICK_END](#) for the actions on right click on your icon or one of its sub-dock.
- [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.

1.4.6 Ok now I have several sections of code to fill. Are there any useful functions to do it ?

A lot of useful macros are provided in [cairo-dock-applet-facility.h](#) to make your life easier :

- 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 **create a surface** that fits your icon from an image, you can use the following :
[CD_APPLET_LOAD_SURFACE_FOR_MY_APPLET](#) & cie
- To directly **set an image on your icon**, you can use the following :
[CD_APPLET_SET_IMAGE_ON_MY_ICON](#) & 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](#)
- 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

1.4.7 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_DRAWING_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.

1.4.8 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 frequency 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](#).

1.4.9 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 are 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](#).

1.4.10 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_MY_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_ICONS_LIST_CONTAINER](#).

1.5 Advanced fonctionnalités

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 `plInterface->load_custom_widget` and `plInterface->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_get_widget_from_name](#). 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_MANAGE_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 applet 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](#), [CairoDockGraph](#), [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_APPLET_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_APPLET_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 each 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:

_CairoContainer	Definition of a Container, whom derive Dock, Desklet, Dialog and FlyingContainer	13
_CairoDataRenderer	Generic DataRenderer. Any implementation of a DataRenderer will derive from this class . . .	14
_CairoDataRendererAttribute	Generic DataRenderer attributes structure. The attributes of any implementation of a Data-Renderer will derive from this class	15
_CairoDataRendererInterface	Interface of a DataRenderer	16
_CairoDesklet	Definition of a Desklet, which derives from a Container	16
_CairoDeskletAttribute	Configuration attributes of a Desklet	17
_CairoDeskletDecoration	Decoration of a Desklet	17
_CairoDeskletRenderer	Definition of a Desklet's renderer	17
_CairoDialog	Definition of a Dialog	18
_CairoDialogAttribute	Configuration attributes of a Dialog	18
_CairoDialogDecorator	Definition of a Dialog decorator. It draws the frame of the Dialog	19
_CairoDialogRenderer	Definition of a Dialog renderer. It draws the inside of the Dialog	19
_CairoDock	Definition of a Dock, which derives from a Container	19
_CairoDockClassAppli	Definition of a Class of application	22
_CairoDockDesktopBackground	Definition of a Desktop Background Buffer. It has a reference count so that it can be shared across all the lib	22
_CairoDockDesktopEnvBackend	Definition of the Desktop Environment backend	23
_CairoDockGLConfig	This structure summarizes the available OpenGL configuration on the system	23
_CairoDockGLFont	Structure used to load a font for OpenGL text rendering	23

_CairoDockGLPath	Definition of a CairoDockGLPath	24
_CairoDockGroupKeyWidget	Definition of a widget corresponding to a given (group;key) pair	24
_CairoDockGuiBackend	Definition of the GUI interface for modules	24
_CairoDockHidingEffect	Definition of a Hiding Effect backend (used to provide an animation when the docks hides/shows itself)	24
_CairoDockImageBuffer	Definition of an Image Buffer. It provides an unified interface for a cairo/opengl image buffer	25
_CairoDockLabelDescription	Description of the rendering of a text	25
_CairoDockModule	Definition of an external module	26
_CairoDockModuleInstance	Definition of an instance of a module. A module can be instanciated several times	27
_CairoDockModuleInterface	Definition of the interface of a module	27
_CairoDockPackage	Definition of a generic package	28
_CairoDockRenderer	Dock's renderer, also known as 'view'	28
_CairoDockTask	Definition of a periodic and asynchronous Task	29
_CairoDockTransition	Transitions are an easy way to set an animation on an Icon to make it change from a state to another	30
_CairoDockVisitCard	Definition of the visit card of a module. Contains everything that is statically defined for a module	31
_CairoDockWMBackend	Definition of the Window Manager backend	31
_CairoEmblem	Definition of an Emblem. You shouldn't access any of its fields directly	31
_CairoGraphAttribute	Attributes of a Graph	32
_CairoIconContainerRenderer	Definition of an Icon container (= an icon holding a sub-dock) renderer	32
_CairoOverlay	Definition of an Icon Overlay	32
_CairoParticle	A particle of a particle system	33
_CairoParticleSystem	A particle system	34
_Icon	Definition of an Icon	34
_IconInterface	Icon's interface	35

Chapter 3

File Index

3.1 File List

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

cairo-dock-animations.h	37
cairo-dock-applet-canvas.h	41
cairo-dock-applet-facility.h	47
cairo-dock-applications-manager.h	68
cairo-dock-class-manager.h	70
cairo-dock-compiz-integration.h	71
cairo-dock-config.h	71
cairo-dock-container.h	73
cairo-dock-core.h	79
cairo-dock-data-renderer-manager.h	79
cairo-dock-data-renderer.h	79
cairo-dock-dbus.h	85
cairo-dock-default-view.h	89
cairo-dock-desklet-factory.h	89
cairo-dock-desklet-manager.h	94
cairo-dock-desktop-file-factory.h	97
cairo-dock-dialog-factory.h	98
cairo-dock-dialog-manager.h	100
cairo-dock-dock-facility.h	109
cairo-dock-dock-factory.h	111
cairo-dock-dock-manager.h	115
cairo-dock-draw-opengl.h	120
cairo-dock-draw.h	124
cairo-dock-emblem.h	126
cairo-dock-file-manager.h	128
cairo-dock-gauge.h	132
cairo-dock-graph.h	132
cairo-dock-gui-factory.h	133
cairo-dock-gui-manager.h	137
cairo-dock-hiding-effect.h	140
cairo-dock-icon-container.h	140
cairo-dock-icon-facility.h	140
cairo-dock-icon-factory.h	149
cairo-dock-icon-manager.h	152
cairo-dock-image-buffer.h	153
cairo-dock-indicator-manager.h	156
cairo-dock-keybinder.h	156
cairo-dock-keyfile-utilities.h	158

cairo-dock-kwin-integration.h	160
cairo-dock-launcher-factory.h	160
cairo-dock-launcher-manager.h	161
cairo-dock-manager.h	163
cairo-dock-module-factory.h	163
cairo-dock-module-manager.h	165
cairo-dock-notifications.h	166
cairo-dock-object.h	167
cairo-dock-opengl-font.h	167
cairo-dock-opengl-path.h	171
cairo-dock-opengl.h	175
cairo-dock-overlay.h	177
cairo-dock-packages.h	180
cairo-dock-particle-system.h	184
cairo-dock-surface-factory.h	186
cairo-dock-task.h	192
cairo-dock-themes-manager.h	197
cairo-dock-X-manager.h	199
cairo-dock-X-utilities.h	202

Chapter 4

Data Structure Documentation

4.1 `_CairoContainer` Struct Reference

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

Data Fields

- GdiObject `object`
object.
- gpointer `pDataSlot` [CAIRO_DOCK_NB_DATA_SLOT]
External data.
- `CairoDockTypeContainer` `iType`
type of container.
- GtkWidget * `pWidget`
window of the container.
- gint `iWidth`
size of the container.
- gint `iWindowPositionX`
position of the container.
- gboolean `bInside`
TRUE is the mouse is inside the container (including the possible sub-widgets).
- `CairoDockTypeHorizontal` `blsHorizontal`
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.

4.1.1 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](#)

4.2 _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.it
- 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` \Leftrightarrow the theme images are rotated 90° clockwise.
- 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)

4.2.1 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](#)

4.3 _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.

4.3.1 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](#)

4.4 `_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.

4.4.1 Detailed Description

Interface of a DataRenderer.

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

- [cairo-dock-data-renderer.h](#)

4.5 `_CairoDesklet` Struct Reference

Definition of a Desklet, which derives from a Container.

4.5.1 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](#)

4.6 `_CairoDeskletAttribute` Struct Reference

Configuration attributes of a Desklet.

4.6.1 Detailed Description

Configuration attributes of a Desklet.

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

- [cairo-dock-desklet-factory.h](#)

4.7 `_CairoDeskletDecoration` Struct Reference

Decoration of a Desklet.

4.7.1 Detailed Description

Decoration of a Desklet.

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

- [cairo-dock-desklet-factory.h](#)

4.8 `_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.

4.8.1 Detailed Description

Definition of a Desklet's renderer.

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

- [cairo-dock-desklet-factory.h](#)

4.9 _CairoDialog Struct Reference

Definition of a Dialog.

Data Fields

- [CairoContainer](#) `container`
container.

4.9.1 Detailed Description

Definition of a Dialog.

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

- [cairo-dock-dialog-factory.h](#)

4.10 _CairoDialogAttribute Struct Reference

Configuration attributes of a Dialog.

Data Fields

- const gchar * [cImagePath](#)
path to an image to display in the left margin, or NULL.
- gint [iIconSize](#)
size of the icon in the left margin, or 0 to use the default one.
- gint [iNbFrames](#)
number of frames of the image, if it's an animated image, otherwise 0.
- const gchar * [cText](#)
text of the message, or NULL.
- gint [iMaxTextWidth](#)
maximum authorized width of the message; it will scroll if it is too large. Set 0 to not limit it.
- [CairoDockLabelDescription](#) * [pTextDescription](#)
A text rendering description of the message, or NULL to use the default one.
- GtkWidget * [pInteractiveWidget](#)
a widget to interact with the user, or NULL.
- const gchar ** [cButtonsImage](#)
a NULL-terminated list of images for buttons, or NULL. "ok" and "cancel" are key word to load the default "ok" and "cancel" buttons.
- [CairoDockActionOnAnswerFunc](#) [pActionFunc](#)
function that will be called when the user click on a button, or NULL.

- gpointer [pUserData](#)
data passed as a parameter of the callback, or NULL.
- GFreeFunc [pFreeDataFunc](#)
a function to free the data when the dialog is destroyed, or NULL.
- gint [iTimeLength](#)
life time of the dialog (in ms), or 0 for an unlimited dialog.
- const gchar * [cDecoratorName](#)
name of a decorator, or NULL to use the default one.
- gboolean [bNoInput](#)
whether the dialog should be transparent to mouse input.
- gboolean [bForceAbove](#)
whether to pop-up the dialog in front of all other windows, including fullscreen windows.
- gboolean [bHideOnClick](#)
for a dialog with no buttons, clicking on it will close it, or hide if this boolean is TRUE.

4.10.1 Detailed Description

Configuration attributes of a Dialog.

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

- [cairo-dock-dialog-factory.h](#)

4.11 `_CairoDialogDecorator` Struct Reference

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

4.11.1 Detailed Description

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

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

- [cairo-dock-dialog-factory.h](#)

4.12 `_CairoDialogRenderer` Struct Reference

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

4.12.1 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](#)

4.13 `_CairoDock` Struct Reference

Definition of a Dock, which derives from a Container.

Data Fields

- [CairoContainer](#) `container`
container.
- [GList](#) * `icons`
the list of icons.
- [gboolean](#) `blsMainDock`
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).
- [CairoDockVisibility](#) `iVisibility`
visibility.
- [gint](#) `iScreenOffsetX`
Horizontal offset of the screen where the dock lives, according to Xinerama.
- [gint](#) `iScreenOffsetY`
Vertical offset of the screen where the dock lives, according to Xinerama.
- [gint](#) `iNumScreen`
number of the screen the dock is placed on (Xinerama).
- [gdouble](#) `iMaxIconHeight`
maximum height of the icons.
- [gdouble](#) `fFlatDockWidth`
width of the dock, only taking into account an alignment of the icons.
- [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](#) `blsBelow`
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](#) `blsDragging`
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](#) `blsShrinkingDown`
whether the dock is shrinking down.
- [gboolean](#) `blsGrowingUp`
whether the dock is growing up.
- [gboolean](#) `blsHiding`
whether the dock is hiding.
- [gboolean](#) `blsShowing`
whether the dock is showing.
- [gboolean](#) `blconIsFlyingAway`
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).
- [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.*

 - [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 edge, 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).*

 - [GldiShape](#) * [pShapeBitmap](#)
- input shape of the window when the dock is at rest.*

 - [GldiShape](#) * [pHiddenShapeBitmap](#)
- input shape of the window when the dock is hidden.*

 - [GldiShape](#) * [pActiveShapeBitmap](#)
- input shape of the window when the dock is active (NULL to cover all dock).*

 - 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.
- `gdouble fBgColorBright [4]`
first color of the gradation
- `gdouble fBgColorDark [4]`
second color of the gradation
- `CairoDockImageBuffer backgroundBuffer`
Background image buffer of the dock.
- `gint iIconSize`
icon size, as specified in the config of the dock
- `guint iSidUpdateDockSize`
Source ID for updating the dock's size and icons layout.
- `gboolean bGlobalIconSize`
whether the dock should use the global icons size parameters.

4.13.1 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](#)

4.14 _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 together with its class.
- `GList * pIconsOfClass`
List of the inhibitors of the class.
- `GList * pAppliOfClass`
List of the appli icons of this class.

4.14.1 Detailed Description

Definition of a Class of application.

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

- [cairo-dock-class-manager.h](#)

4.15 _CairoDockDesktopBackground Struct Reference

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

4.15.1 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-X-manager.h](#)

4.16 `_CairoDockDesktopEnvBackend` Struct Reference

Definition of the Desktop Environment backend.

4.16.1 Detailed Description

Definition of the Desktop Environment backend.

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

- [cairo-dock-file-manager.h](#)

4.17 `_CairoDockGLConfig` Struct Reference

This structure summarizes the available OpenGL configuration on the system.

Data Fields

- GLXContext [context](#)

*GdkGLConfig *pGLConfig;*

4.17.1 Detailed Description

This structure summarizes the available OpenGL configuration on the system.

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

- [cairo-dock-opengl.h](#)

4.18 `_CairoDockGLFont` Struct Reference

Structure used to load a font for OpenGL text rendering.

4.18.1 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](#)

4.19 `_CairoDockGLPath` Struct Reference

Definition of a `CairoDockGLPath`.

4.19.1 Detailed Description

Definition of a `CairoDockGLPath`.

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

- [cairo-dock-opengl-path.h](#)

4.20 `_CairoDockGroupKeyWidget` Struct Reference

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

4.20.1 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](#)

4.21 `_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](#))([CairoDockModuleInstance](#) *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](#))([CairoDockModuleInstance](#) *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.

4.21.1 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](#)

4.22 `_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 after the icons are drawn (cairo)
- void(* `post_render_opengl`)(`CairoDock` *pDock, double fOffset)
function called after the icons are drawn (opengl)
- void(* `init`)(`CairoDock` *pDock)
function called when the animation is started.

4.22.1 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](#)

4.23 `_CairoDockImageBuffer` Struct Reference

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

4.23.1 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](#)

4.24 `_CairoDockLabelDescription` Struct Reference

Description of the rendering of a text.

Data Fields

- gint `iSize`
font size (also approximately the resulting size in pixels)
- gchar * `cFont`
font.
- PangoWeight `iWeight`
text weight. The higher, the thicker the strokes are.
- PangoStyle `iStyle`

- text style (italic or normal).*
- gdouble [fColorStart](#) [3]
 - first color of the characters.*
- gdouble [fColorStop](#) [3]
 - second color of the characters. If different from the first one, it will make a gradation.*
- gboolean [bVerticalPattern](#)
 - TRUE if the gradation is vertical (from top to bottom).*
- gdouble [fBackgroundColor](#) [4]
 - frame background color. Set the alpha channel to 0 to not draw a frame in the background.*
- 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 "&" -> "&")*
- gdouble [fMaxRelativeWidth](#)
 - maximum width allowed, in ratio of the screen's width. Carriage returns will be inserted if necessary. 0 means no limit.*

4.24.1 Detailed Description

Description of the rendering of a text.

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

- [cairo-dock-surface-factory.h](#)

4.25 _CairoDockModule Struct Reference

Definition of an external module.

Data Fields

- gchar * [cSoFilePath](#)
 - path to the .so file.*
- gpointer [handle](#)
 - internal structure of the .so file, once it has been opened.*
- [CairoDockModuleInterface](#) * [pInterface](#)
 - interface of the module.*
- [CairoDockVisitCard](#) * [pVisitCard](#)
 - visit card of the module.*
- gchar * [cConfFilePath](#)
 - conf file of the module.*
- gboolean [bCanDetach](#)
 - TRUE if the appet can be detached from a dock (desklet mode).*
- GList * [pInstancesList](#)
 - List of instances of the module.*

4.25.1 Detailed Description

Definition of an external module.

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

- [cairo-dock-module-factory.h](#)

4.26 `_CairoDockModuleInstance` Struct Reference

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

Data Fields

- `CairoDockModule` * `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.
- `CairoContainer` * `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.

4.26.1 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-factory.h](#)

4.27 `_CairoDockModuleInterface` Struct Reference

Definition of the interface of a module.

4.27.1 Detailed Description

Definition of the interface of a module.

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

- [cairo-dock-module-factory.h](#)

4.28 _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.

4.28.1 Detailed Description

Definition of a generic package.

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

- [cairo-dock-packages.h](#)

4.29 _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.

4.29.1 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](#)

4.30 `_CairoDockTask` Struct Reference

Definition of a periodic and asynchronous Task.

Data Fields

- gint [iSidTimer](#)
ID of the timer of the Task.
- gint [iSidTimerUpdate](#)
ID of the timer to check the end of the thread.
- gint [iThreadIsRunning](#)

- Atomic value, set to 1 when the thread is running.*
- [CairoDockGetDataAsyncFunc get_data](#)
function carrying out the heavy job.
- [CairoDockUpdateSyncFunc update](#)
function carrying out the update of the dock. Returns TRUE to continue, FALSE to stop.
- [guint iPeriod](#)
interval of time in seconds, 0 to run the Task once.
- [CairoDockFrequencyState iFrequencyState](#)
state of the frequency of the Task.
- [gpointer pSharedMemory](#)
structure passed as parameter of the 'get_data' and 'update' functions. Must not be accessed outside of these 2 functions !
- [GTimer * pClock](#)
timer to get the accurate amount of time since last update.
- [double fElapsedTime](#)
time elapsed since last update.
- [GFreeFunc free_data](#)
function called when the task is destroyed to free the shared memory (optionnal).
- [gboolean bDiscard](#)
TRUE when the task has been discarded.

4.30.1 Detailed Description

Definition of a periodic and asynchronous Task.

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

- [cairo-dock-task.h](#)

4.31 _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 steps since the beginning of the transition, in ms.
- `CairoContainer * pContainer`
Container of the Icon.

4.31.1 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](#)

4.32 `_CairoDockVisitCard` Struct Reference

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

4.32.1 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-factory.h](#)

4.33 `_CairoDockWMBackend` Struct Reference

Definition of the Window Manager backend.

4.33.1 Detailed Description

Definition of the Window Manager backend.

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

- [cairo-dock-X-manager.h](#)

4.34 `_CairoEmblem` Struct Reference

Definition of an Emblem. You shouldn't access any of its fields directly.

4.34.1 Detailed Description

Definition of an Emblem. You shouldn't access any of its fields directly.

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

- [cairo-dock-emblem.h](#)

4.35 `_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.

4.35.1 Detailed Description

Attributes of a Graph.

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

- [cairo-dock-graph.h](#)

4.36 `_CairoIconContainerRenderer` Struct Reference

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

4.36.1 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](#)

4.37 `_CairoOverlay` Struct Reference

Definition of an Icon Overlay.

Data Fields

- [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).

4.37.1 Detailed Description

Definition of an Icon Overlay.

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

- [cairo-dock-overlay.h](#)

4.38 `_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.

4.38.1 Detailed Description

A particle of a particle system.

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

- [cairo-dock-particle-system.h](#)

4.39 _CairoParticleSystem Struct Reference

A particle system.

4.39.1 Detailed Description

A particle system.

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

- [cairo-dock-particle-system.h](#)

4.40 _Icon Struct Reference

Definition of an Icon.

Data Fields

- GdiObject [object](#)
object
- CairoDockIconTrueType [iTrueType](#)
type of the icon.
- 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 with the dock is hidden.
- gboolean `bPointed`
Whether the icon is currently pointed or not.
- `CairoContainer` * `pContainer`
container where the icon is currently.

4.40.1 Detailed Description

Definition of an `Icon`.

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

- [cairo-dock-icon-factory.h](#)

4.41 `_IconInterface` Struct Reference

`Icon`'s interface.

Data Fields

- void(* `load_image`)(`Icon` *`icon`)
function that loads the icon surface (and optionnally texture).
- gboolean(* `on_delete`)(`Icon` *`icon`)
function called when the icon is deleted from the current theme.
- void(* `action_on_drag_hover`)(`Icon` *`icon`)
function called when the user drag something over the icon for more than 500ms.

4.41.1 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 [cairo_dock_stop_icon_animation](#)(plcon)
- #define [cairo_dock_get_animation_delta_t](#)(pContainer)
- #define [cairo_dock_get_slow_animation_delta_t](#)(pContainer)
- #define [cairo_dock_has_transition](#)(plcon)
- #define [cairo_dock_get_transition_count](#)(plcon)
- #define [cairo_dock_get_transition_elapsed_time](#)(plcon)
- #define [cairo_dock_get_transition_fraction](#)(plcon)

Typedefs

- typedef gboolean(* [CairoDockTransitionRenderFunc](#))(Icon *plcon, gpointer pUserData)
callback to render the icon with libcairo at each step of the Transition.
- typedef gboolean(* [CairoDockTransitionGLRenderFunc](#))(Icon *plcon, 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](#) (CairoContainer *pContainer)
- void [cairo_dock_start_icon_animation](#) (Icon *icon, CairoDock *pDock)
- void [cairo_dock_request_icon_animation](#) (Icon *plcon, CairoContainer *pContainer, const gchar *c-Animation, int iNbRounds)
- void [cairo_dock_trigger_icon_removal_from_dock](#) (Icon *plcon)

- void `cairo_dock_set_transition_on_icon` (`Icon *pIcon`, `CairoContainer *pContainer`, `CairoDockTransitionRenderFunc` `render_step_cairo`, `CairoDockTransitionGLRenderFunc` `render_step_opengl`, `gboolean` `bFastPace`, `gint` `iDuration`, `gboolean` `bRemoveWhenFinished`, `gpointer` `pUserData`, `GFreeFunc` `pFreeUserDataFunc`)
- 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

<code>pContainer</code>	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

<code>pDock</code>	the Dock to animate.
--------------------	----------------------

5.1.2.3 `#define cairo_dock_stop_icon_animation(pIcon)`

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

Parameters

<code>pIcon</code>	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

<code>pContainer</code>	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

<i>pContainer</i>	the container.
-------------------	----------------

5.1.2.6 #define cairo_dock_has_transition(*plcon*)

Say if an Icon has a Transition.

Parameters

<i>plcon</i>	the icon.
--------------	-----------

Returns

TRUE if the icon has a Transition.

5.1.2.7 #define cairo_dock_get_transition_count(*plcon*)

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

Parameters

<i>plcon</i>	the icon.
--------------	-----------

Returns

the elapsed number of steps.

5.1.2.8 #define cairo_dock_get_transition_elapsed_time(*plcon*)

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

Parameters

<i>plcon</i>	the icon.
--------------	-----------

Returns

the elapsed time.

5.1.2.9 #define cairo_dock_get_transition_fraction(*plcon*)

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

<i>plcon</i>	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

<i>pDock</i>	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

<i>pDock</i>	the dock.
--------------	-----------

5.1.3.3 void cairo_dock_launch_animation (CairoContainer * *pContainer*)

Launch the animation of a Container.

Parameters

<i>pContainer</i>	the container to animate.
-------------------	---------------------------

5.1.3.4 void cairo_dock_start_icon_animation (Icon * *icon*, CairoDock * *pDock*)

Launch the animation of an Icon. Do nothing if the icon will not be animated or if the icon is at rest.

Parameters

<i>icon</i>	the icon to animate.
<i>pDock</i>	the dock containing the icon.

5.1.3.5 void cairo_dock_request_icon_animation (Icon * *pIcon*, CairoContainer * *pContainer*, 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

<i>pIcon</i>	the icon to animate.
<i>pContainer</i>	the container containing the icon.
<i>cAnimation</i>	name of the animation.
<i>iNbRounds</i>	number of rounds the animation will be played.

5.1.3.6 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

<i>pIcon</i>	the icon to remove
--------------	--------------------

5.1.3.7 void `cairo_dock_set_transition_on_icon (Icon * pIcon, CairoContainer * 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

<i>pIcon</i>	the icon.
<i>pContainer</i>	the Container of the Icon. It will be shared with the transition.
<i>render_step_cairo</i>	the cairo rendering function.
<i>render_step_opengl</i>	the openGL rendering function (can be NULL, in which case the texture mapping from the cairo drawing is done automatically).
<i>bFastPace</i>	TRUE for a high frequency refresh (this uses of course more CPU).
<i>iDuration</i>	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 .
<i>bRemoveWhenFinished</i>	TRUE to destroy and remove the transition when it is finished.
<i>pUserData</i>	data passed to the rendering functions.
<i>pFreeUserDataFunc</i>	function called to free the user data when the transition is destroyed (optional).

5.1.3.8 void `cairo_dock_remove_transition_on_icon (Icon * pIcon)`

Stop and remove the Transition of an Icon.

Parameters

<i>pIcon</i>	the icon.
--------------	-----------

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

Macros

- #define [CD_APPLET_DEFINE_ALL_BEGIN](#)(*cName*, *iMajorVersion*, *iMinorVersion*, *iMicroVersion*, *iAppletCategory*, *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

<code>_cName</code>	nom de sous lequel l'applet sera enregistree par Cairo-Dock.
<code>_iMajorVersion</code>	version majeure du dock necessaire au bon fonctionnement de l'applet.
<code>_iMinorVersion</code>	version mineure du dock necessaire au bon fonctionnement de l'applet.
<code>_iMicroVersion</code>	version micro du dock necessaire au bon fonctionnement de l'applet.
<code>_iApplet-Category</code>	Categorie de l'applet (CAIRO_DOCK_CATEGORY_ACCESSORY, CAIRO_DOCK_CATEGORY_DESKTOP, CAIRO_DOCK_CATEGORY_CONTROLLER)
<code>_cDescription</code>	description et mode d'emploi succinct de l'applet.
<code>_cAuthor</code>	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

<code>pApplet</code>	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 default 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 default 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 default 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 default 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 default 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 default 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, cDefaultValues)
- #define [CD_CONFIG_GET_STRING_LIST](#)(cGroupName, cKeyName, length)
- #define [CD_CONFIG_GET_COLOR_WITH_DEFAULT](#)(cGroupName, cKeyName, pColorBuffer, pDefaultColor)
- #define [CD_CONFIG_GET_COLOR](#)(cGroupName, cKeyName, pColorBuffer)
- #define [CD_CONFIG_GET_COLOR_RVB_WITH_DEFAULT](#)(cGroupName, cKeyName, pColorBuffer, pDefaultColor)
- #define [CD_CONFIG_GET_COLOR_RVB](#)(cGroupName, cKeyName, pColorBuffer)
- #define [CD_CONFIG_GET_THEME_PATH](#)(cGroupName, cKeyName, cThemeDirName, cDefaultThemeName)
- #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_ADD_FIRST_SEPARATOR_IN_MENU`
- #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_SURFACE_ON_MY_ICON_WITH_ZOOM`(pSurface, fScale)
- #define `CD_APPLET_SET_SURFACE_ON_MY_ICON_WITH_ALPHA`(pSurface, fAlpha)
- #define `CD_APPLET_SET_SURFACE_ON_MY_ICON_WITH_BAR`(pSurface, fValue)
- #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_OR_RETURN`(...)
- #define `CD_APPLET_FINISH_DRAWING_MY_ICON`
- #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_MAKE_EMBLEM`(cImageFile)
- #define `CD_APPLET_DRAW_EMBLEM_ON_MY_ICON`(pEmblem)
- #define `CD_APPLET_ADD_DATA_RENDERER_ON_MY_ICON`(pAttr)
- #define `CD_APPLET_RELOAD_MY_DATA_RENDERER`(pAttr)
- #define `CD_APPLET_RENDER_NEW_DATA_ON_MY_ICON`(pValues)
- #define `CD_APPLET_REMOVE_MY_DATA_RENDERER`
- #define `CD_APPLET_REFRESH_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`(plcon)
- #define `CD_APPLET_DETACH_ICON_FROM_MY_ICONS_LIST`(plcon)
- #define `CD_APPLET_LOAD_MY_ICONS_LIST`(plconList, cDockRendererName, cDeskletRendererName, pDeskletRendererConfig)
- #define `CD_APPLET_ADD_ICON_IN_MY_ICONS_LIST`(plcon)
- #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 *plconContext, cairo_surface_t *pSurface, double fScale, double fAlpha, Icon *plcon)
- void `cairo_dock_draw_bar_on_icon` (cairo_t *plconContext, double fValue, Icon *plcon)
- void `cairo_dock_set_icon_surface_with_reflect` (cairo_t *plconContext, cairo_surface_t *pSurface, Icon *plcon, CairoContainer *pContainer)
- gboolean `cairo_dock_set_image_on_icon` (cairo_t *plconContext, const gchar *clconName, Icon *plcon, CairoContainer *pContainer)
- void `cairo_dock_set_image_on_icon_with_default` (cairo_t *plconContext, const gchar *cImage, Icon *plcon, CairoContainer *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

<i>pIconContext</i>	the drawing context; is not altered by the function.
<i>pSurface</i>	the surface to apply.
<i>pIcon</i>	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

<i>cGroupName</i>	name of the group in the conf file.
<i>cKeyName</i>	name of the key in the conf file.
<i>bDefaultValue</i>	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

<i>cGroupName</i>	name of the group in the conf file.
<i>cKeyName</i>	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

<i>cGroupName</i>	name of the group in the conf file.
<i>cKeyName</i>	name of the key in the conf file.
<i>iDefaultValue</i>	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

<i>cGroupName</i>	name of the group in the conf file.
<i>cKeyName</i>	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

<i>cGroupName</i>	name of the group in the conf file.
<i>cKeyName</i>	name of the key in the conf file.
<i>fDefaultValue</i>	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

<i>cGroupName</i>	name of the group in the conf file.
<i>cKeyName</i>	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

<i>cGroupName</i>	name of the group in the conf file.
<i>cKeyName</i>	name of the key in the conf file.
<i>iNbElements</i>	number of elements to get from the conf file.
<i>iValueBuffer</i>	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

<i>cGroupName</i>	name of the group in the conf file.
<i>cKeyName</i>	name of the key in the conf file.
<i>cDefaultValue</i>	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

<i>cGroupName</i>	name of the group in the conf file.
<i>cKeyName</i>	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

<i>cGroupName</i>	name of the group in the conf file.
<i>cKeyName</i>	name of the key in the conf file.
<i>cDefaultFileName</i>	default file 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

<i>cGroupName</i>	name of the group in the conf file.
<i>cKeyName</i>	name of the key in the conf file.
<i>length</i>	pointer to the number of strings that were extracted from the conf file.
<i>cDefaultValues</i>	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 freed 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

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

Returns

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

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

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

Parameters

<i>cGroupName</i>	name of the group in the conf file.
<i>cKeyName</i>	name of the key in the conf file.
<i>pColorBuffer</i>	a table of 4 'double' already allocated, that will be filled with the color components.
<i>pDefaultColor</i>	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(*cGroupName*, *cKeyName*, *pColorBuffer*)

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

Parameters

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

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

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

Parameters

<i>cGroupName</i>	name of the group in the conf file.
<i>cKeyName</i>	name of the key in the conf file.
<i>pColorBuffer</i>	a table of 3 'double' already allocated, that will be filled with the color components.
<i>pDefaultColor</i>	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_RVB(*cGroupName*, *cKeyName*, *pColorBuffer*)

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

Parameters

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

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

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

Parameters

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

Returns

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

5.3.2.19 #define CD_CONFIG_GET_GAUGE_THEME(*cGroupName*, *cKeyName*)

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

Parameters

<i>cGroupName</i>	name of the group (in the conf file).
<i>cKeyName</i>	name of the key (in the conf file).

5.3.2.20 #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

<i>cGroupName</i>	name of the group.
<i>cNewGroupName</i>	new name of the group.

5.3.2.21 #define CD_APPLET_ADD_SUB_MENU_WITH_IMAGE(*cLabel*, *pMenu*, *clmage*)

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

Parameters

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

Returns

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

5.3.2.22 #define CD_APPLET_ADD_SUB_MENU(*cLabel*, *pMenu*)

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

Parameters

<i>cLabel</i>	name of the sub-menu.
<i>pMenu</i>	GtkWidget of the menu we will add the sub-menu to..

Returns

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

5.3.2.23 #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

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

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

Create and add an entry to a menu.

Parameters

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

5.3.2.25 #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

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

5.3.2.26 `#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

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

5.3.2.27 `#define CD_APPLET_ADD_SEPARATOR_IN_MENU(pMenu)`

Create and add a separator to a menu.

5.3.2.28 `#define CD_APPLET_ADD_FIRST_SEPARATOR_IN_MENU`

Create and add the first separator to the main menu.

5.3.2.29 `#define CD_APPLET_POPUP_MENU_ON_MY_ICON(pMenu)`

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

Parameters

<i>pMenu</i>	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, available 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 occurred (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 shortcutkey to an action. Unbind it when you don't want it anymore. myApplet is passed as the callback data.

Parameters

<i>cShortcut</i>	a shortcut.
<i>cDescription</i>	a short description of the action
<i>cGroupName</i>	group name where it's stored in the applet's conf file
<i>cKeyName</i>	key name where it's stored in the applet's conf file
<i>handler</i>	function called when the shortcut is pressed by the user

Returns

the key binding

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

<i>cImagePath</i>	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

<i>cUserImage-Name</i>	name or path of an user image.
<i>cDefaultLocal-ImageName</i>	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

<i>pSurface</i>	the surface to draw on your icon.
-----------------	-----------------------------------

5.3.2.52 #define CD_APPLET_SET_SURFACE_ON_MY_ICON_WITH_ZOOM(*pSurface*, *fScale*)

Apply a surface on the applet's icon, with a zoom factor and centered, and redraw it.

Parameters

<i>pSurface</i>	the surface to draw on your icon.
<i>fScale</i>	zoom factor (at 1 the surface will fill all the icon).

5.3.2.53 #define CD_APPLET_SET_SURFACE_ON_MY_ICON_WITH_ALPHA(*pSurface*, *fAlpha*)

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

Parameters

<i>pSurface</i>	the surface to draw on your icon.
<i>fAlpha</i>	transparency (in [0,1]).

5.3.2.54 #define CD_APPLET_SET_SURFACE_ON_MY_ICON_WITH_BAR(*pSurface*, *fValue*)

Apply a surface on the applet's icon with add a bar at the bottom, and redraw it. The bar is drawn at the bottom of the icon with a gradation from red to green and a given length.

Parameters

<i>pSurface</i>	the surface to draw on your icon.
<i>fValue</i>	the value representing a percentage, in [-1,1]. If negative, the gradation is inverted, and the absolute value is used.

5.3.2.55 #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

<i>cIconName</i>	name of an icon or path to an image.
------------------	--------------------------------------

5.3.2.56 #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

<i>cIconName</i>	name of an icon or path to an image.
<i>cDefaultLocalImageName</i>	name of an image to use as a fallback (taken in the applet's installation folder).

5.3.2.57 #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.

5.3.2.58 #define CD_APPLET_SET_NAME_FOR_MY_ICON(*clconName*)

Set a new label on the applet's icon.

Parameters

<i>clconName</i>	the label.
------------------	------------

5.3.2.59 #define CD_APPLET_SET_NAME_FOR_MY_ICON_PRINTF(*clconNameFormat*, ...)

Set a new label on the applet's icon.

Parameters

<i>clconName-Format</i>	the label, in a 'printf'-like format.
...	values to be written in the string.

5.3.2.60 #define CD_APPLET_SET_QUICK_INFO_ON_MY_ICON(*cQuickInfo*)

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

Parameters

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

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

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

Parameters

<i>cQuickInfo-Format</i>	the label, in a 'printf'-like format.
...	values to be written in the string.

5.3.2.62 #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

<i>iTimeInSeconds</i>	the time in seconds.
-----------------------	----------------------

5.3.2.63 #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

<i>iTimeInSeconds</i>	the time in seconds.
-----------------------	----------------------

5.3.2.64 #define CD_APPLET_SET_SIZE_AS_QUICK_INFO(*iSizeInBytes*)

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

Parameters

<i>iSizeInBytes</i>	the size in bytes, converted into a readable format.
---------------------	--

5.3.2.65 #define CD_APPLET_SET_STATIC_ICON

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

5.3.2.66 #define CD_APPLET_UNSET_STATIC_ICON

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

5.3.2.67 #define CD_APPLET_SET_ALWAYS_VISIBLE_ICON(*bAlwaysVisible*)

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

5.3.2.68 #define CD_APPLET_ANIMATE_MY_ICON(*cAnimationName*, *iAnimationLength*)

Launch an animation on the applet's icon.

Parameters

<i>cAnimation-Name</i>	name of the animation.
<i>iAnimation-Length</i>	number of rounds the animation should be played.

5.3.2.69 #define CD_APPLET_STOP_ANIMATING_MY_ICON

Stop any animation on the applet's icon.

5.3.2.70 #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

<i>cAnimation-Name</i>	name of the animation.
<i>iAnimation-Length</i>	number of rounds the animation should be played, or 0 for an endless animation.

5.3.2.71 `#define CD_APPLET_STOP_DEMANDING_ATTENTION`

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

5.3.2.72 `#define CD_APPLET_GET_MY_ICON_EXTENT(iWidthPtr, iHeightPtr)`

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

Parameters

<i>iWidthPtr</i>	pointer to the width.
<i>iHeightPtr</i>	pointer to the height.

5.3.2.73 `#define CD_APPLET_START_DRAWING_MY_ICON`

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

5.3.2.74 `#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.
-----	-------------------------------------

5.3.2.75 `#define CD_APPLET_FINISH_DRAWING_MY_ICON`

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

5.3.2.76 `#define CD_APPLET_ADD_OVERLAY_ON_MY_ICON(clmageFile, iPosition)`

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

Parameters

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

Returns

TRUE if the overlay has been successfully added.

5.3.2.77 `#define CD_APPLET_PRINT_OVERLAY_ON_MY_ICON(clmageFile, iPosition)`

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

Parameters

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

Returns

TRUE if the overlay has been successfully added.

5.3.2.78 #define CD_APPLET_REMOVE_OVERLAY_ON_MY_ICON(*iPosition*)

Remove an overlay the applet's icon, given its position (there is only one overlay at a given position).

Parameters

<i>iPosition</i>	position of the overlay
------------------	-------------------------

5.3.2.79 #define CD_APPLET_MAKE_EMBLEM(*cImageFile*)

Make an emblem from an image. If the image is given by its sole name, it is looked up inside the root theme folder. Free it with `cairo_dock_free_emblem`.

Parameters

<i>cImageFile</i>	name of an image file.
-------------------	------------------------

Returns

a newly allocated CairoEmblem.

5.3.2.80 #define CD_APPLET_DRAW_EMBLEM_ON_MY_ICON(*pEmblem*)

Draw an emblem on the applet's icon. The emblem is drawn directly on the icon, and is erased if the icon is redrawn.

Parameters

<i>pEmblem</i>	an emblem.
----------------	------------

5.3.2.81 #define CD_APPLET_ADD_DATA_RENDERER_ON_MY_ICON(*pAttr*)

Add a Data Renderer the applet's icon.

Parameters

<i>pAttr</i>	the attributes of the Data Renderer. They allow you to define its properties.
--------------	---

5.3.2.82 #define CD_APPLET_RELOAD_MY_DATA_RENDERER(*pAttr*)

Reload the Data Renderer of the applet's icon. Pass NULL as the attributes to simply reload the current data renderer without changing any of its parameters. Previous values are kept.

Parameters

<i>pAttr</i>	the attributes of the Data Renderer, or NULL to simply reload the Data Renderer as it is.
--------------	---

5.3.2.83 #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

<i>pValues</i>	the values, a table of double of the correct size.
----------------	--

5.3.2.84 #define CD_APPLET_REMOVE_MY_DATA_RENDERER

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

5.3.2.85 #define CD_APPLET_REFRESH_MY_DATA_RENDERER

Refresh the Data Renderer of the applet's icon, to redraw it when the applet's size has changed.

5.3.2.86 #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.

5.3.2.87 #define CD_APPLET_MY_CONTAINER_IS_OPENGL

Say if the applet's container currently supports OpenGL.

5.3.2.88 #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

<i>cRendererName</i>	name of the renderer.
<i>pConfig</i>	configuration data for the renderer, or NULL.

5.3.2.89 #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

<i>cRendererName</i>	name of the renderer.
----------------------	-----------------------

5.3.2.90 #define CD_APPLET_SET_STATIC_DESKLET

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

5.3.2.91 #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.

5.3.2.92 #define CD_APPLET_DELETE_MY_ICONS_LIST

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

5.3.2.93 #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

<i>pIcon</i>	the icon to remove.
--------------	---------------------

Returns

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

5.3.2.94 #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

<i>pIcon</i>	the icon to remove.
--------------	---------------------

Returns

whether the icon has been removed or not.

5.3.2.95 #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

<i>pIconList</i>	a list of icons. It will belong to the applet's container after that.
<i>cDockRendererName</i>	name of a renderer in case the applet is in dock mode.
<i>cDeskletRendererName</i>	name of a renderer in case the applet is in desklet mode.
<i>pDeskletRendererConfig</i>	possible configuration parameters for the desklet renderer.

5.3.2.96 #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

<i>pIcon</i>	an icon.
--------------	----------

5.3.2.97 #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.

5.3.2.98 #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.

5.3.2.99 #define CD_APPLET_MANAGE_APPLICATION(*cApplicationClass*)

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

Parameters

<i>cApplicationClass</i>	the class of the application you wish to control (in lower case), or NULL to stop controlling any appli.
--------------------------	--

5.3.2.100 #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

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

5.3.4.2 void `cairo_dock_draw_bar_on_icon` (`cairo_t * plconContext`, `double fValue`, `Icon * plcon`)

Draw a bar at the bottom of an Icon, with a gradation from red to green and a given length.

Parameters

<code>plconContext</code>	the drawing context; is not altered by the function.
<code>fValue</code>	the value representing a percentage, in [-1,1]. if negative, the gradation is inverted, and the absolute value is used.
<code>plcon</code>	the icon.

5.3.4.3 void `cairo_dock_set_icon_surface_with_reflect` (`cairo_t * plconContext`, `cairo_surface_t * pSurface`, `Icon * plcon`, `CairoContainer * pContainer`)

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

Parameters

<code>plconContext</code>	the drawing context; is not altered by the function.
<code>pSurface</code>	the surface to apply.
<code>plcon</code>	the icon.
<code>pContainer</code>	the container of the icon.

5.3.4.4 `gboolean` `cairo_dock_set_image_on_icon` (`cairo_t * plconContext`, `const gchar * clconName`, `Icon * plcon`, `CairoContainer * pContainer`)

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

Parameters

<code>plconContext</code>	the drawing context; is not altered by the function.
<code>clconName</code>	name or path to an icon image.
<code>plcon</code>	the icon.
<code>pContainer</code>	the container of the icon.

Returns

TRUE if everything went smoothly.

5.3.4.5 void `cairo_dock_set_image_on_icon_with_default` (`cairo_t * plconContext`, `const gchar * clmage`, `Icon * plcon`, `CairoContainer * 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

<code>plconContext</code>	the drawing context; is not altered by the function.
<code>clmage</code>	name of an image to apply on the icon.
<code>plcon</code>	the icon.
<code>pContainer</code>	the container of the icon.
<code>cDefaultImagePath</code>	path to a default image.

5.3.4.6 `gchar*` `cairo_dock_get_human_readable_size (long long int iSizeInBytes)`

Convert a size in bytes into a readable format.

Parameters

<i>iSizeInBytes</i>	size in bytes.
---------------------	----------------

Returns

a newly allocated string.

5.3.4.7 `void` `cairo_dock_play_sound (const gchar * cSoundPath)`

Play a sound, through `Alsa` or `PulseAudio`.

Parameters

<i>cSoundPath</i>	path to an audio file.
-------------------	------------------------

5.4 `cairo-dock-applications-manager.h` File Reference

Macros

- `#define` `_cairo_dock_appli_is_on_our_way`(`icon`, `pDock`)
`void` `cairo_dock_reset_applications_manager (void)`;

Functions

- `void` `cairo_dock_start_applications_manager (CairoDock *pDock)`
- `Icon *` `cairo_dock_search_window_covering_dock (CairoDock *pDock, gboolean bMaximizedWindow, gboolean bFullScreenWindow)`
- `Icon *` `cairo_dock_search_window_overlapping_dock (CairoDock *pDock)`
- `GList *` `cairo_dock_get_current_applis_list (void)`
- `Window` `cairo_dock_get_current_active_window (void)`
- `Icon *` `cairo_dock_get_current_active_icon (void)`
- `Icon *` `cairo_dock_get_icon_with_Xid (Window Xid)`
- `void` `cairo_dock_foreach_applis (CairoDockForeachIconFunc pFunction, gboolean bOutsideDockOnly, gpointer pUserData)`
- `void` `cairo_dock_foreach_applis_on_viewport (CairoDockForeachIconFunc pFunction, int iNumDesktop, int iNumViewportX, int iNumViewportY, gpointer pUserData)`

5.4.1 Detailed Description

This class manages the list of known windows, ie the Taskbar. It also provides convenient functions to act on all the applis icons at once.

5.4.2 Function Documentation

5.4.2.1 `void` `cairo_dock_start_applications_manager (CairoDock * pDock)`

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

Parameters

<i>pDock</i>	the main dock
--------------	---------------

5.4.2.2 Icon* cairo_dock_search_window_covering_dock (CairoDock * *pDock*, gboolean *bMaximizedWindow*, gboolean *bFullScreenWindow*)

Get the icon of an application whose window covers entirely a dock, or NULL if none. If both parameters are FALSE, check for all windows.

Parameters

<i>pDock</i>	the dock to test.
<i>bMaximizedWindow</i>	check for maximized windows only.
<i>bFullScreenWindow</i>	check for full screen windows only.

Returns

the icon representing the window, or NULL if none has been found.

5.4.2.3 Icon* cairo_dock_search_window_overlapping_dock (CairoDock * *pDock*)

Get the icon of an application whose window overlaps a dock, or NULL if none.

Parameters

<i>pDock</i>	the dock to test.
--------------	-------------------

Returns

the icon of the window, or NULL if none has been found.

5.4.2.4 GList* cairo_dock_get_current_applis_list (void)

Get the list of appli's icons currently known by Cairo-Dock, 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's icons. You must free the list when you're done with it, but not the icons.

5.4.2.5 Window cairo_dock_get_current_active_window (void)

Get the currently active window ID.

Returns

the X id.

5.4.2.6 Icon* cairo_dock_get_current_active_icon (void)

Get the icon of the currently active window.

Returns

the icon (maybe not inside a dock).

5.4.2.7 Icon* cairo_dock_get_icon_with_Xid (Window Xid)

Get the icon of a given window. The search is fast.

Parameters

<i>Xid</i>	the id of the X window.
------------	-------------------------

Returns

the icon (maybe not inside a dock).

5.4.2.8 void cairo_dock_foreach_applis (CairoDockForeachIconFunc *pFunction*, gboolean *bOutsideDockOnly*, gpointer *pUserData*)

Run a function on all appli's icons.

Parameters

<i>pFunction</i>	a /ref CairoDockForeachIconFunc function to be called
<i>bOutsideDockOnly</i>	TRUE if you only want to go through icons that are not inside a dock, FALSE to go through all icons.
<i>pUserData</i>	a data passed to the function.

5.4.2.9 void cairo_dock_foreach_applis_on_viewport (CairoDockForeachIconFunc *pFunction*, int *iNumDesktop*, int *iNumViewportX*, int *iNumViewportY*, gpointer *pUserData*)

Run a function on all appli's icons present on a given workspace.

Parameters

<i>pFunction</i>	a /ref CairoDockForeachIconFunc function to be called
<i>iNumDesktop</i>	number of the desktop
<i>iNumViewportX</i>	number of the horizontal viewport
<i>iNumViewportY</i>	number of the vertical viewport
<i>pUserData</i>	a data passed to the function.

5.5 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 `cairo_dock_set_class_order_in_dock` (`Icon *plcon`, `CairoDock *pDock`)
`void cairo_dock_reorder_classes (void);`
- void `cairo_dock_set_data_from_class` (`const gchar *cClass`, `Icon *plcon`)

5.5.1 Detailed Description

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

5.5.2 Macro Definition Documentation

5.5.2.1 #define `cairo_dock_register_class`(`cDesktopFile`)

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

Parameters

<code>cDesktopFile</code>	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.

5.5.3 Function Documentation

5.5.3.1 void `cairo_dock_set_data_from_class` (`const gchar * cClass`, `Icon * plcon`)

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

Parameters

<code>cClass</code>	the class name
<code>plcon</code>	the icon

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

5.6.1 Detailed Description

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

5.7 cairo-dock-config.h File Reference

Macros

- #define `cairo_dock_get_pango_weight_from_1_9`(*iWeight*)

Functions

- void `cairo_dock_load_config` (const gchar **cConfFilePath*)
- gboolean `cairo_dock_is_loading` (void)
- void `cairo_dock_get_version_from_string` (const gchar **cVersionString*, int **iMajorVersion*, int **iMinorVersion*, int **iMicroVersion*)
- void `cairo_dock_decrypt_string` (const gchar **cEncryptedString*, gchar ***cDecryptedString*)
- void `cairo_dock_encrypt_string` (const gchar **cDecryptedString*, gchar ***cEncryptedString*)
- void `cairo_dock_load_current_theme` (void)

5.7.1 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 contains 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.

5.7.2 Macro Definition Documentation

5.7.2.1 #define `cairo_dock_get_pango_weight_from_1_9`(*iWeight*)

Convert an integer in [0,9] into a Pango text weight.

Parameters

<i>iWeight</i>	weight between 0 and 9.
----------------	-------------------------

5.7.3 Function Documentation

5.7.3.1 void `cairo_dock_load_config` (const gchar * *cConfFilePath*)

Load the Cairo-Dock's config and everything.

Parameters

<i>cConfFilePath</i>	path to the main conf file.
----------------------	-----------------------------

5.7.3.2 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).

5.7.3.3 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

<i>cVersionString</i>	the string of the form "x.y.z".
<i>iMajorVersion</i>	pointer to the major version.
<i>iMinorVersion</i>	pointer to the minor version.
<i>iMicroVersion</i>	pointer to the micro version.

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

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

Parameters

<i>cEncrypted-String</i>	the encrypted string.
<i>cDecrypted-String</i>	the decrypted string.

5.7.3.5 void cairo_dock_encrypt_string (const gchar * *cDecryptedString*, gchar ** *cEncryptedString*)

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

Parameters

<i>cDecrypted-String</i>	the decrypted string.
<i>cEncrypted-String</i>	the encrypted string.

5.7.3.6 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.

5.8 cairo-dock-container.h File Reference

Data Structures

- struct [_CairoContainer](#)

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

Macros

- #define [CAIRO_CONTAINER\(p\)](#)
Get the Container part of a pointer.
- #define [cairo_dock_init_container\(pContainer\)](#)
- #define [cairo_dock_init_container_no_opengl\(pContainer\)](#)
- #define [gldi_container_enable_drop\(pContainer, pCallBack, data\)](#)
- #define [cairo_dock_get_max_scale\(pContainer\)](#)
- #define [cairo_dock_popup_menu_on_container\(menu, pContainer\)](#)

Enumerations

- enum [CairoContainerNotifications](#) {
 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 [CairoDockTypeHorizontal](#)
Main orientation of a container.
- enum [CairoDockTypeContainer](#)
Types of available containers.

Functions

- void [cairo_dock_finish_container](#) ([CairoContainer](#) *pContainer)
- void [cairo_dock_redraw_container](#) ([CairoContainer](#) *pContainer)
- void [cairo_dock_redraw_container_area](#) ([CairoContainer](#) *pContainer, [GdkRectangle](#) *pArea)
- void [cairo_dock_redraw_icon](#) ([Icon](#) *icon, [CairoContainer](#) *pContainer)
- [CairoContainer](#) * [cairo_dock_search_container_from_icon](#) ([Icon](#) *icon)
- gboolean [cairo_dock_string_is_adress](#) (const gchar *cString)
- void [cairo_dock_notify_drop_data](#) (gchar *cReceivedData, [Icon](#) *pPointedIcon, double fOrder, [CairoContainer](#) *pContainer)
- void [cairo_dock_popup_menu_on_icon](#) ([GtkWidget](#) *menu, [Icon](#) *pIcon, [CairoContainer](#) *pContainer)
- [GtkWidget](#) * [cairo_dock_add_in_menu_with_stock_and_data](#) (const gchar *cLabel, const gchar *gtkStock, [GCallback](#) pFunction, [GtkWidget](#) *pMenu, [gpointer](#) pData)
- [GtkWidget](#) * [cairo_dock_create_sub_menu](#) (const gchar *cLabel, [GtkWidget](#) *pMenu, const gchar *cImage)
- [GtkWidget](#) * [cairo_dock_build_menu](#) ([Icon](#) *icon, [CairoContainer](#) *pContainer)

5.8.1 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.

If you write a new type of container, you must call [cairo_dock_init_container](#) when you create it and [cairo_dock_finish_container](#) when you destroy it.

5.8.2 Macro Definition Documentation

5.8.2.1 #define cairo_dock_init_container(*pContainer*)

Initialize a Container : create a GTK window with transparency and OpenGL support. To be called when you create a new container.

Parameters

<i>pContainer</i>	a Container.
-------------------	--------------

Returns

the newly allocated GTK window.

5.8.2.2 #define cairo_dock_init_container_no_opengl(*pContainer*)

Same as above, but with no OpenGL support.

Parameters

<i>pContainer</i>	a Container.
-------------------	--------------

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

Enable a Container to accept drag-and-drops.

Parameters

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

5.8.2.4 #define cairo_dock_get_max_scale(*pContainer*)

Get the maximum zoom of the icons inside a given container.

Parameters

<i>pContainer</i>	the container.
-------------------	----------------

Returns

the maximum scale factor.

5.8.2.5 #define cairo_dock_popup_menu_on_container(*menu*, *pContainer*)

Pop-up a menu on a container. In the case of a dock, it prevents this one from shrinking down.

Parameters

<i>menu</i>	the menu.
<i>pContainer</i>	the container that was clicked.

5.8.3 Enumeration Type Documentation

5.8.3.1 enum CairoContainerNotifications

signals

Enumerator:

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

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

5.8.4 Function Documentation

5.8.4.1 void cairo_dock_finish_container (CairoContainer * pContainer)

Finish a Container. To be called before you free it.

Parameters

<i>pContainer</i>	a Container.
-------------------	--------------

5.8.4.2 void cairo_dock_redraw_container (CairoContainer * pContainer)

Clear and trigger the redraw of a Container.

Parameters

<i>pContainer</i>	the Container to redraw.
-------------------	--------------------------

5.8.4.3 void `cairo_dock_redraw_container_area` (`CairoContainer * pContainer`, `GdkRectangle * pArea`)

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

Parameters

<i>pContainer</i>	the Container to redraw.
<i>pArea</i>	the zone to redraw.

5.8.4.4 void `cairo_dock_redraw_icon` (`Icon * icon`, `CairoContainer * pContainer`)

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

Parameters

<i>icon</i>	l'icone a retracer.
<i>pContainer</i>	le container de l'icone.

5.8.4.5 `CairoContainer*` `cairo_dock_search_container_from_icon` (`Icon * icon`)

Search for the Container of a given Icon (dock or desklet in the case of an applet).

Parameters

<i>icon</i>	the icon.
-------------	-----------

Returns

the container contening this icon, or NULL if the icon is nowhere.

5.8.4.6 `gboolean` `cairo_dock_string_is_adress` (`const gchar * cString`)

Say if a string is an adress ([file://xxx](#), [http://xxx](#), [ftp://xxx](#), etc).

Parameters

<i>cString</i>	a string.
----------------	-----------

Returns

TRUE if it's an address.

5.8.4.7 void `cairo_dock_notify_drop_data` (`gchar * cReceivedData`, `Icon * pPointedIcon`, `double fOrder`, `CairoContainer * pContainer`)

Notify everybody that a drop has just occurred.

Parameters

<i>cReceivedData</i>	the dropped data.
<i>pPointedIcon</i>	the icon which was pointed when the drop occurred.
<i>fOrder</i>	the order of the icon if the drop occurred on it, or <code>LAST_ORDER</code> if the drop occurred between 2 icons.
<i>pContainer</i>	the container of the icon

5.8.4.8 void `cairo_dock_popup_menu_on_icon (GtkWidget * menu, Icon * pIcon, CairoContainer * pContainer)`

Pop-up a menu on an icon. The menu is placed so that it touches the icon, without overlapping it. If the icon is NULL, it will be placed it at the mouse's position. In the case of a dock, it prevents this one from shrinking down.

Parameters

<i>menu</i>	the menu.
<i>pIcon</i>	the icon, or NULL.
<i>pContainer</i>	the container that was clicked.

5.8.4.9 GtkWidget* `cairo_dock_add_in_menu_with_stock_and_data (const gchar * cLabel, const gchar * gtkStock, GCallback pFunction, GtkWidget * pMenu, gpointer pData)`

Add an entry to a given menu.

Parameters

<i>cLabel</i>	label of the entry
<i>gtkStock</i>	a GTK stock or a path to an image
<i>pFunction</i>	callback
<i>pMenu</i>	the menu to insert the entry in
<i>pData</i>	data to feed the callback with

Returns

the new menu-entry that has been added.

5.8.4.10 GtkWidget* `cairo_dock_create_sub_menu (const gchar * cLabel, GtkWidget * pMenu, const gchar * cImage)`

Add sub-menu to a given menu.

Parameters

<i>cLabel</i>	label of the sub-menu
<i>pMenu</i>	the menu to insert the entry in
<i>cImage</i>	a GTK stock or a path to an image

Returns

the new sub-menu that has been added.

5.8.4.11 GtkWidget* `cairo_dock_build_menu (Icon * icon, CairoContainer * pContainer)`

Build the main menu of a Container.

Parameters

<i>icon</i>	the icon that was left-clicked, or NULL if none.
<i>pContainer</i>	the container that was left-clicked.

Returns

the menu.

5.9 cairo-dock-core.h File Reference

5.9.1 Detailed Description

This class instanciates the different core managers.

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

Functions

- [CairoDockGLFont](#) * [cairo_dock_get_default_data_renderer_font](#) (void)

5.10.1 Detailed Description

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

5.10.2 Function Documentation

5.10.2.1 [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

5.11 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(plcon)`
- `#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](#) *plcon, [CairoContainer](#) *pContainer, [CairoDataRendererAttribute](#) *pAttribute)
- void [cairo_dock_render_new_data_on_icon](#) ([Icon](#) *plcon, [CairoContainer](#) *pContainer, [cairo_t](#) *pCairoContext, double *pNewValues)
- void [cairo_dock_remove_data_renderer_on_icon](#) ([Icon](#) *plcon)
- void [cairo_dock_reload_data_renderer_on_icon](#) ([Icon](#) *plcon, [CairoContainer](#) *pContainer, [CairoDataRendererAttribute](#) *pAttribute)
- void [cairo_dock_resize_data_renderer_history](#) ([Icon](#) *plcon, int iNewMemorySize)
- void [cairo_dock_refresh_data_renderer](#) ([Icon](#) *plcon, [CairoContainer](#) *pContainer, [cairo_t](#) *pCairoContext)

5.11.1 Detailed Description

This class defines the Data Renderer structure and API. A [CairoDataRenderer](#) is a generic way to render onto an icon a set of N values defined by : $\{y_k = f(t)\}$, $k=0..N$. For instance you could represent the (cpu, mem, swap) evolution over the time. You bind a Data Renderer with [/ref cairo_dock_add_new_data_renderer_on_icon](#). You can specify some configuration parameters for the Data Renderer with a set of attributes, that derive from a [CairoDataRendererAttribute](#). You feed the Data Renderer with [/ref cairo_dock_render_new_data_on_icon](#), providing it the correct number of values. If you want to change any parameter of a Data Renderer, use [/ref cairo_dock_reload_data_renderer_on_icon](#), which keeps the history. To remove the Data Renderer from an icon, use [/ref cairo_dock_remove_data_renderer_on_icon](#).

5.11.2 Macro Definition Documentation

5.11.2.1 `#define cairo_dock_get_icon_data_renderer(plcon)`

Structure Access

5.11.2.2 `#define CAIRO_DATA_RENDERER(r)`

Get the elementary part of a Data Renderer

Parameters

<i>r</i>	a high level data renderer
----------	----------------------------

Returns

a CairoDataRenderer*

5.11.2.3 #define cairo_data_renderer_get_data(*pRenderer*)

Get the data of a Data Renderer

Parameters

<i>pRenderer</i>	a data renderer
------------------	-----------------

Returns

a CairoDataToRenderer*

5.11.2.4 #define CAIRO_DATA_RENDERER_ATTRIBUTE(*pAttr*)

Get the elementary part of a Data Renderer Attribute

Parameters

<i>pAttr</i>	a high level data renderer attribute
--------------	--------------------------------------

Returns

a CairoDataRendererAttribute*

5.11.2.5 #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

<i>pRenderer</i>	a data renderer
------------------	-----------------

Returns

number of values a DataRenderer displays

5.11.2.6 #define cairo_data_renderer_get_min_value(*pRenderer*, *i*)

Data AccessGet the lower range of the i-th value.

Parameters

<i>pRenderer</i>	a data renderer
<i>i</i>	the number of the value

Returns

a double

5.11.2.7 `#define cairo_data_renderer_get_max_value(pRenderer, i)`

Get the upper range of the i-th value.

Parameters

<i>pRenderer</i>	a data renderer
<i>i</i>	the number of the value

Returns

a double

5.11.2.8 `#define cairo_data_renderer_get_value(pRenderer, i, t)`

Get the i-th value at the time t.

Parameters

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

Returns

a double

5.11.2.9 `#define cairo_data_renderer_get_current_value(pRenderer, i)`

Get the current i-th value.

Parameters

<i>pRenderer</i>	a data renderer
<i>i</i>	the number of the value

Returns

a double

5.11.2.10 `#define cairo_data_renderer_get_previous_value(pRenderer, i)`

Get the previous i-th value.

Parameters

<i>pRenderer</i>	a data renderer
<i>i</i>	the number of the value

Returns

a double

5.11.2.11 `#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

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

Returns

a double in [0,1]

5.11.2.12 `#define cairo_data_renderer_get_normalized_current_value(pRenderer, i)`

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

Parameters

<i>pRenderer</i>	a data renderer
<i>i</i>	the number of the value

Returns

a double in [0,1]

5.11.2.13 `#define cairo_data_renderer_get_normalized_previous_value(pRenderer, i)`

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

Parameters

<i>pRenderer</i>	a data renderer
<i>i</i>	the number of the value

Returns

a double in [0,1]

5.11.2.14 `#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

<i>pRenderer</i>	a data renderer
<i>i</i>	the number of the value

Returns

a double in [0,1]

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

Data FormatWrite a value in a readable text format.

Parameters

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

5.11.2.16 #define cairo_data_renderer_format_value(*pRenderer*, *i*)

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

Parameters

<i>pRenderer</i>	a data renderer
<i>i</i>	the number of the value

5.11.3 Function Documentation

5.11.3.1 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

5.11.3.2 void cairo_dock_add_new_data_renderer_on_icon (Icon * *pIcon*, CairoContainer * *pContainer*, CairoDataRendererAttribute * *pAttribute*)

Add a Data Renderer on an icon (usually the icon of an applet). A Data Renderer is a view that will be used to display a set of values on the icon.

Parameters

<i>pIcon</i>	the icon
<i>pContainer</i>	the icon's container
<i>pAttribute</i>	attributes defining the Renderer

5.11.3.3 void cairo_dock_render_new_data_on_icon (Icon * *pIcon*, CairoContainer * *pContainer*, cairo_t * *pCairoContext*, double * *pNewValues*)

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

Parameters

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

5.11.3.4 void `cairo_dock_remove_data_renderer_on_icon (Icon * pIcon)`

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

Parameters

<i>pIcon</i>	the icon
--------------	----------

5.11.3.5 void `cairo_dock_reload_data_renderer_on_icon (Icon * pIcon, CairoContainer * pContainer, CairoDataRendererAttribute * pAttribute)`

Reload the Data Renderer of an icon. If no attributes are provided, it simply reload it with its current attributes. History is kept.

Parameters

<i>pIcon</i>	the icon
<i>pContainer</i>	the icon's container
<i>pAttribute</i>	new attributes defining the Renderer, or NULL to keep the current ones

5.11.3.6 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

<i>pIcon</i>	the icon
<i>iNewMemorySize</i>	the new size of history

5.11.3.7 void `cairo_dock_refresh_data_renderer (Icon * pIcon, CairoContainer * pContainer, cairo_t * pCairoContext)`

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

Parameters

<i>pIcon</i>	the icon
<i>pContainer</i>	the icon's container
<i>pCairoContext</i>	a drawing context on the icon

5.12 cairo-dock-dbus.h File Reference

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)
 - gchar * `cairo_dock_dbus_get_uchar` (DBusGProxy *pDBusProxy, const gchar *cAccessor)
 - void `cairo_dock_dbus_call` (DBusGProxy *pDBusProxy, const gchar *cCommand)
 - void `cairo_dock_dbus_get_property_in_value` (DBusGProxy *pDBusProxy, const gchar *cInterface, const gchar *cProperty, GValue *pProperty)
- deprecated...*

5.12.1 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.

5.12.2 Function Documentation

5.12.2.1 DBusGConnection* cairo_dock_get_session_connection (void)

Get the connection to the 'session' Bus.

Returns

the connection to the bus.

5.12.2.2 gboolean cairo_dock_register_service_name (const gchar * cServiceName)

Register a new service on the session bus.

Parameters

<code>cServiceName</code>	name of the service.
---------------------------	----------------------

Returns

TRUE in case of success, false otherwise.

5.12.2.3 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.

5.12.2.4 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

<i>name</i>	a name on the bus.
<i>path</i>	the path.
<i>interface</i>	name of the interface.

Returns

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

5.12.2.5 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

<i>name</i>	a name on the bus.
<i>path</i>	the path.
<i>interface</i>	name of the interface.

Returns

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

5.12.2.6 gboolean cairo_dock_dbus_detect_application (const gchar * *cName*)

Detect if an application is currently running on Session bus.

Parameters

<i>cName</i>	name of the application.
--------------	--------------------------

Returns

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

5.12.2.7 gboolean cairo_dock_dbus_detect_system_application (const gchar * *cName*)

Detect if an application is currently running on System bus.

Parameters

<i>cName</i>	name of the application.
--------------	--------------------------

Returns

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

5.12.2.8 gboolean cairo_dock_dbus_get_boolean (DBusGProxy * *pDBusProxy*, const gchar * *cAccessor*)

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

Parameters

<i>pDBusProxy</i>	proxy to the connection.
<i>cAccessor</i>	name of the accessor.

Returns

the value of the parameter.

5.12.2.9 `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

<i>pDBusProxy</i>	proxy to the connection.
<i>cAccessor</i>	name of the accessor.

Returns

the value of the parameter.

5.12.2.10 `int cairo_dock_dbus_get_integer (DBusGProxy * pDBusProxy, const gchar * cAccessor)`

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

Parameters

<i>pDBusProxy</i>	proxy to the connection.
<i>cAccessor</i>	name of the accessor.

Returns

the value of the parameter.

5.12.2.11 `gchar* cairo_dock_dbus_get_string (DBusGProxy * pDBusProxy, const gchar * cAccessor)`

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

Parameters

<i>pDBusProxy</i>	proxy to the connection.
<i>cAccessor</i>	name of the accessor.

Returns

the value of the parameter, to be freed with `g_free`.

5.12.2.12 `gchar** cairo_dock_dbus_get_string_list (DBusGProxy * pDBusProxy, const gchar * cAccessor)`

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

Parameters

<i>pDBusProxy</i>	proxy to the connection.
<i>cAccessor</i>	name of the accessor.

Returns

the value of the parameter, to be freed with `g_strfreev`.

5.12.2.13 `guchar* cairo_dock_dbus_get_uchar (DBusGProxy * pDBusProxy, const gchar * cAccessor)`

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

Parameters

<i>pDBusProxy</i>	proxy to the connection.
<i>cAccessor</i>	name of the accessor.

Returns

the value of the parameter.

5.12.2.14 `void cairo_dock_dbus_call (DBusGProxy * pDBusProxy, const gchar * cCommand)`

Call a command on the bus.

Parameters

<i>pDBusProxy</i>	proxy to the connection.
<i>cCommand</i>	name of the commande.

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

5.13.1 Detailed Description

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

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

Data Structures

- [struct `_CairoDeskletDecoration`](#)
Decoration of a Desklet.
- [struct `_CairoDeskletAttribute`](#)
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 `CAIRO_DOCK_IS_DESKLET`(pContainer)
- #define `CAIRO_DESKLET`(pContainer)
- #define `cairo_dock_add_interactive_widget_to_desklet`(pInteractiveWidget, pDesklet)

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 * cairo_dock_new_desklet` (void)
- void `cairo_dock_free_desklet` (`CairoDesklet *pDesklet`)
- void `cairo_dock_configure_desklet` (`CairoDesklet *pDesklet`, `CairoDeskletAttribute *pAttribute`)
- void `cairo_dock_add_interactive_widget_to_desklet_full` (`GtkWidget *pInteractiveWidget`, `CairoDesklet *pDesklet`, int iRightMargin)
- void `cairo_dock_set_desklet_margin` (`CairoDesklet *pDesklet`, int iRightMargin)
- `GtkWidget * cairo_dock_steal_interactive_widget_from_desklet` (`CairoDesklet *pDesklet`)
- void `cairo_dock_hide_desklet` (`CairoDesklet *pDesklet`)
- void `cairo_dock_show_desklet` (`CairoDesklet *pDesklet`)
- void `cairo_dock_zoom_out_desklet` (`CairoDesklet *pDesklet`)
- void `cairo_dock_set_desklet_accessibility` (`CairoDesklet *pDesklet`, `CairoDeskletVisibility iVisibility`, gboolean bSaveState)
- void `cairo_dock_set_desklet_sticky` (`CairoDesklet *pDesklet`, gboolean bSticky)
- void `cairo_dock_lock_desklet_position` (`CairoDesklet *pDesklet`, gboolean bPositionLocked)

5.14.1 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, supports 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 functions in [cairo-dock-desklet-manager.h](#).

5.14.2 Macro Definition Documentation

5.14.2.1 #define CAIRO_DOCK_IS_DESKLET(*pContainer*)

Say if a Container is a Desklet.

Parameters

<i>pContainer</i>	the container.
-------------------	----------------

Returns

TRUE if the container is a desklet.

5.14.2.2 #define CAIRO_DESKLET(*pContainer*)

Cast a Container into a Desklet.

Parameters

<i>pContainer</i>	the container.
-------------------	----------------

Returns

the desklet.

5.14.2.3 #define cairo_dock_add_interactive_widget_to_desklet(*pInteractiveWidget*, *pDesklet*)

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

Parameters

<i>pInteractiveWidget</i>	the widget to add.
<i>pDesklet</i>	the desklet.

5.14.3 Enumeration Type Documentation

5.14.3.1 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

5.14.4 Function Documentation

5.14.4.1 CairoDesklet* cairo_dock_new_desklet (void)

Create a simple desklet container. This function should NOT be used directly.

Returns

the newly allocated desklet.

5.14.4.2 void cairo_dock_free_desklet (CairoDesklet * pDesklet)

Destroy a desklet, and free all the allocated resources. The interactive widget is removed before, and can be inserted anywhere after that. This function should NOT be used directly.

Parameters

<i>pDesklet</i>	the desklet to destroy.
-----------------	-------------------------

5.14.4.3 void cairo_dock_configure_desklet (CairoDesklet * pDesklet, CairoDeskletAttribute * pAttribute)

Configure a un desklet. It places it, resizes it, sets up its accessibility, locks its position, and sets up its decorations.

Parameters

<i>pDesklet</i>	the desklet.
<i>pAttribute</i>	the attributes to configure the desklet.

5.14.4.4 void cairo_dock_add_interactive_widget_to_desklet_full (GtkWidget * pInteractiveWidget, CairoDesklet * pDesklet, 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

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

5.14.4.5 void cairo_dock_set_desklet_margin (CairoDesklet * pDesklet, int iRightMargin)

Sezt 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

<i>pDesklet</i>	the desklet.
<i>iRightMargin</i>	right margin, in pixels.

5.14.4.6 GtkWidget* cairo_dock_steal_interactive_widget_from_desklet (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

<i>pDesklet</i>	the desklet with an interactive widget.
-----------------	---

Returns

the widget.

5.14.4.7 void cairo_dock_hide_desklet (CairoDesklet * *pDesklet*)

Hide a desklet.

Parameters

<i>pDesklet</i>	the desklet.
-----------------	--------------

5.14.4.8 void cairo_dock_show_desklet (CairoDesklet * *pDesklet*)

Show a desklet, and give it the focus.

Parameters

<i>pDesklet</i>	the desklet.
-----------------	--------------

5.14.4.9 void cairo_dock_zoom_out_desklet (CairoDesklet * *pDesklet*)

Launch a "zoom out" animation on a desklet.

Parameters

<i>pDesklet</i>	the desklet.
-----------------	--------------

5.14.4.10 void cairo_dock_set_desklet_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

<i>pDesklet</i>	the desklet.
<i>iVisibility</i>	the new accessibility.
<i>bSaveState</i>	whether to save the new state in the conf file.

5.14.4.11 void `cairo_dock_set_desklet_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/view-port is saved.

Parameters

<code>pDesklet</code>	the desklet.
<code>bSticky</code>	whether the desklet should be sticky or not.

5.14.4.12 void `cairo_dock_lock_desklet_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

<code>pDesklet</code>	the desklet.
<code>bPositionLocked</code>	whether the position should be locked or not.

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

Typedefs

- typedef `gboolean(* CairoDockForeachDeskletFunc)`(`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`,
`NOTIFICATION_NEW_DESKLET` }
signals

Functions

- `CairoDesklet * cairo_dock_create_desklet` (`Icon *pIcon`, `CairoDeskletAttribute *pAttributes`)
- void `cairo_dock_destroy_desklet` (`CairoDesklet *pDesklet`)
- `CairoDesklet * cairo_dock_foreach_desklet` (`CairoDockForeachDeskletFunc` `pCallback`, `gpointer user_data`)
- void `cairo_dock_foreach_icons_in_desklets` (`CairoDockForeachIconFunc` `pFunction`, `gpointer pUserData`)
- void `cairo_dock_reload_desklets_decorations` (`gboolean bDefaultThemeOnly`)
- void `cairo_dock_set_all_desklets_visible` (`gboolean bOnWidgetLayerToo`)
- void `cairo_dock_set_desklets_visibility_to_default` (void)
- `CairoDesklet * cairo_dock_get_desklet_by_Xid` (`Window Xid`)
- `Icon * cairo_dock_find_clicked_icon_in_desklet` (`CairoDesklet *pDesklet`)

5.15.1 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 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, supports 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.

5.15.2 Enumeration Type Documentation

5.15.2.1 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.

NOTIFICATION_NEW_DESKLET notification called when a new desklet is created.

5.15.3 Function Documentation

5.15.3.1 CairoDesklet* cairo_dock_create_desklet (Icon * *pIcon*, CairoDeskletAttribute * *pAttributes*)

Create a desklet linked to an Icon, and load its configuration.

Parameters

<i>pIcon</i>	the main icon, or NULL.
<i>pAttributes</i>	the configuration attributes, or NULL.

Returns

the newly allocated desklet.

5.15.3.2 void cairo_dock_destroy_desklet (CairoDesklet * *pDesklet*)

Destroy a desklet.

Parameters

<i>pDesklet</i>	a desklet.
-----------------	------------

5.15.3.3 **CairoDesklet*** `cairo_dock_foreach_desklet (CairoDockForeachDeskletFunc 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

<i>pCallback</i>	function to be called on each desklet. If it returns TRUE, the loop ends and the function returns the current desklet.
<i>user_data</i>	data to be passed to the callback.

Returns

the found desklet, or NULL.

5.15.3.4 `void cairo_dock_foreach_icons_in_desklets (CairoDockForeachIconFunc pFunction, gpointer pUserData)`

Execute an action on all icons being inside a desklet.

Parameters

<i>pFunction</i>	the action.
<i>pUserData</i>	data passed to the callback.

5.15.3.5 `void cairo_dock_reload_desklets_decorations (gboolean bDefaultThemeOnly)`

Reload the decorations of all the desklets.

Parameters

<i>bDefaultTheme-Only</i>	whether to reload only the desklet that have the default decoration theme.
---------------------------	--

5.15.3.6 `void cairo_dock_set_all_desklets_visible (gboolean bOnWidgetLayerToo)`

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

Parameters

<i>bOnWidget-LayerToo</i>	TRUE if you want to act on the desklet that are on the WidgetLayer as well.
---------------------------	---

5.15.3.7 `void cairo_dock_set_desklets_visibility_to_default (void)`

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

5.15.3.8 CairoDesklet* cairo_dock_get_desklet_by_Xid (Window Xid)

Search the desklet whose X ID matches the given one.

Parameters

<i>Xid</i>	an X ID.
------------	----------

Returns

the desklet that matches, or NULL if none match.

5.15.3.9 Icon* cairo_dock_find_clicked_icon_in_desklet (CairoDesklet * pDesklet)

Find the currently pointed icon in a desklet, taking into account the 3D rotations.

Parameters

<i>pDesklet</i>	the desklet.
-----------------	--------------

Returns

the pointed icon, or NULL if none.

5.16 cairo-dock-desktop-file-factory.h File Reference

Functions

- void [cairo_dock_remove_html_spaces](#) (gchar *cString)
- gchar * [cairo_dock_add_desktop_file_from_uri](#) (const gchar *cURI, const gchar *cDockName, double fOrder, GError **erreur)
- gchar * [cairo_dock_add_desktop_file_from_type](#) (CairoDockDesktopFileType iLauncherType, const gchar *cDockName, double fOrder, GError **erreur)

5.16.1 Detailed Description

This class handles the creation and update of desktop files, which are group/key pair files used by Cairo-Dock to store information about icons : launchers, separators, sub-docks.

5.16.2 Function Documentation

5.16.2.1 void cairo_dock_remove_html_spaces (gchar * cString)

Replace the %20 by normal spaces into the string. The string is directly modified.

Parameters

<i>cString</i>	the string (it can't be a constant string)
----------------	--

5.16.2.2 `gchar* cairo_dock_add_desktop_file_from_uri (const gchar * cURI, const gchar * cDockName, double fOrder, GError ** erreur)`

Create, add and fill a desktop file for a given URI. The URI can be either a common desktop file, a script, or a file/folder/mounting point.

Parameters

<i>cURI</i>	URI of a file defining the launcher.
<i>cDockName</i>	name of the dock the separator will be added.
<i>fOrder</i>	order of the icon inside the dock.
<i>erreur</i>	an error filled if something went wrong.

Returns

the name of the new desktop file, in a newly allocated string, or NULL if failed.

5.16.2.3 `gchar* cairo_dock_add_desktop_file_from_type (CairoDockDesktopFileType iLauncherType, const gchar * cDockName, double fOrder, GError ** erreur)`

Create and add an empty default desktop file for a given type.

Parameters

<i>iLauncherType</i>	type of the icon it will represent : launcher, file, container icon, separator.
<i>cDockName</i>	name of the dock the separator will be added.
<i>fOrder</i>	order of the icon inside the dock.
<i>erreur</i>	an error filled if something went wrong.

Returns

the name of the new desktop file, in a newly allocated string, or NULL if failed.

5.17 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 decorator. It draws the frame of the Dialog.
- struct [_CairoDialogAttribute](#)
Configuration attributes of a Dialog.
- struct [_CairoDialog](#)
Definition of a Dialog.

Macros

- #define [CAIRO_DOCK_IS_DIALOG](#)(pContainer)
- #define [CAIRO_DIALOG](#)(pContainer)

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'.

Functions

- [CairoDialog](#) * [cairo_dock_new_dialog](#) ([CairoDialogAttribute](#) *pAttribute, [Icon](#) *pIcon, [CairoContainer](#) *pContainer)
- void [cairo_dock_free_dialog](#) ([CairoDialog](#) *pDialog)
- GtkWidget * [cairo_dock_steal_interactive_widget_from_dialog](#) ([CairoDialog](#) *pDialog)

5.17.1 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 only defines the constructor and destructor of a Dialog; to actually pop up a Dialog, use the Dialog Manager's functions in [cairo-dock-dialog-manager.h](#).

5.17.2 Macro Definition Documentation

5.17.2.1 #define CAIRO_DOCK_IS_DIALOG(pContainer)

Say if a Container is a Dialog.

Parameters

<i>pContainer</i>	the container.
-------------------	----------------

Returns

TRUE if the container is a dialog.

5.17.2.2 #define CAIRO_DIALOG(pContainer)

Cast a Container into a Dialog.

Parameters

<i>pContainer</i>	the container.
-------------------	----------------

Returns

the dialog.

5.17.3 Function Documentation

5.17.3.1 CairoDialog* cairo_dock_new_dialog (CairoDialogAttribute * pAttribute, Icon * plcon, CairoContainer * pContainer)

Creates a Dialog from a set of attributes. The Dialog is not placed, and has no interaction with the user.

Parameters

<i>pAttribute</i>	attributes of the dialog.
<i>plcon</i>	the icon that will hold the dialog.
<i>pContainer</i>	the container of the icon.

Returns

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

5.17.3.2 void cairo_dock_free_dialog (CairoDialog * pDialog)

Free a Dialog and all its allocated resources. Should never be used, use [cairo_dock_dialog_unreference](#) instead.

Parameters

<i>pDialog</i>	the dialog.
----------------	-------------

5.17.3.3 GtkWidget* cairo_dock_steal_interactive_widget_from_dialog (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

<i>pDialog</i>	the desklet with an interactive widget.
----------------	---

Returns

the widget.

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

Macros

- #define [cairo_dock_remove_dialog_if_any](#)(icon)

Enumerations

- enum [CairoDialogNotifications](#)
signals

Functions

- gboolean [cairo_dock_dialog_reference](#) (CairoDialog *pDialog)
- gboolean [cairo_dock_dialog_unreference](#) (CairoDialog *pDialog)
- gboolean [cairo_dock_remove_dialog_if_any_full](#) (Icon *icon, gboolean bAll)
- CairoDialog * [cairo_dock_build_dialog](#) (CairoDialogAttribute *pAttribute, Icon *plcon, CairoContainer *pContainer)
- CairoDialog * [cairo_dock_show_dialog_full](#) (const gchar *cText, Icon *plcon, CairoContainer *pContainer, double fTimeLength, const gchar *clconPath, GtkWidget *pInteractiveWidget, CairoDockActionOnAnswerFunc pActionFunc, gpointer data, GFreeFunc pFreeDataFunc)
- CairoDialog * [cairo_dock_show_temporary_dialog_with_icon_printf](#) (const gchar *cText, Icon *plcon, CairoContainer *pContainer, double fTimeLength, const gchar *clconPath,...) G_GNUC_PRINTF(1
- CairoDialog CairoDialog * [cairo_dock_show_temporary_dialog_with_icon](#) (const gchar *cText, Icon *plcon, CairoContainer *pContainer, double fTimeLength, const gchar *clconPath)
- CairoDialog * [cairo_dock_show_temporary_dialog](#) (const gchar *cText, Icon *plcon, CairoContainer *pContainer, double fTimeLength)
- CairoDialog * [cairo_dock_show_temporary_dialog_with_default_icon](#) (const gchar *cText, Icon *plcon, CairoContainer *pContainer, double fTimeLength)
- CairoDialog * [cairo_dock_show_dialog_with_question](#) (const gchar *cText, Icon *plcon, CairoContainer *pContainer, const gchar *clconPath, CairoDockActionOnAnswerFunc pActionFunc, gpointer data, GFreeFunc pFreeDataFunc)
- CairoDialog * [cairo_dock_show_dialog_with_entry](#) (const gchar *cText, Icon *plcon, CairoContainer *pContainer, const gchar *clconPath, const gchar *cTextForEntry, CairoDockActionOnAnswerFunc pActionFunc, gpointer data, GFreeFunc pFreeDataFunc)
- CairoDialog * [cairo_dock_show_dialog_with_value](#) (const gchar *cText, Icon *plcon, CairoContainer *pContainer, const gchar *clconPath, double fValue, double fMaxValue, CairoDockActionOnAnswerFunc pActionFunc, gpointer data, GFreeFunc pFreeDataFunc)
- int [cairo_dock_show_dialog_and_wait](#) (const gchar *cText, Icon *plcon, CairoContainer *pContainer, double fTimeLength, const gchar *clconPath, GtkWidget *pInteractiveWidget)
- gchar * [cairo_dock_show_demand_and_wait](#) (const gchar *cMessage, Icon *plcon, CairoContainer *pContainer, const gchar *clInitialAnswer)
- double [cairo_dock_show_value_and_wait](#) (const gchar *cMessage, Icon *plcon, CairoContainer *pContainer, double flInitialValue, double fMaxValue)
- int [cairo_dock_ask_question_and_wait](#) (const gchar *cQuestion, Icon *plcon, CairoContainer *pContainer)
- gboolean [cairo_dock_icon_has_dialog](#) (Icon *plcon)
- Icon * [cairo_dock_get_dialogless_icon_full](#) (CairoDock *pDock)
- CairoDialog * [cairo_dock_show_general_message](#) (const gchar *cMessage, double fTimeLength)
- int [cairo_dock_ask_general_question_and_wait](#) (const gchar *cQuestion)
- void [cairo_dock_hide_dialog](#) (CairoDialog *pDialog)
- void [cairo_dock_unhide_dialog](#) (CairoDialog *pDialog)
- void [cairo_dock_toggle_dialog_visibility](#) (CairoDialog *pDialog)

5.18.1 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.

Dialogs are constructed with a set of attributes grouped inside a [_CairoDialogAttribute](#). See [cairo-dock-dialog-factory.h](#) for the list of available attributes.

The most generic way to build a Dialog is to fill a `_CairoDialogAttribute` and pass it to `cairo_dock_build_dialog`.

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 `cairo_dock_show_temporary_dialog_with_icon`
- to ask the user a choice, a value or a text, you can use `cairo_dock_show_dialog_with_question`, `cairo_dock_show_dialog_with_value` or `cairo_dock_show_dialog_with_entry`.
- if you need to block while waiting for the user, use the `xxx_and_wait` version of these functions.
- if you want to pop up only 1 dialog at once on a given icon, use `cairo_dock_remove_dialog_if_any` before you pop up your dialog.

5.18.2 Macro Definition Documentation

5.18.2.1 `#define cairo_dock_remove_dialog_if_any(icon)`

Unreference all the dialogs pointing on an icon.

Parameters

<i>icon</i>	the icon you want to delete all dialogs from.
-------------	---

Returns

TRUE if at least one dialog has been unreferenced.

5.18.3 Function Documentation

5.18.3.1 `gboolean cairo_dock_dialog_reference (CairoDialog * pDialog)`

Increase by 1 the reference of a dialog. Use `cairo_dock_dialog_unreference` when you're done, so that the dialog can be destroyed.

Parameters

<i>pDialog</i>	the dialog.
----------------	-------------

Returns

TRUE if the reference was not nul, otherwise you must not use it.

5.18.3.2 `gboolean cairo_dock_dialog_unreference (CairoDialog * pDialog)`

Decrease by 1 the reference of a dialog. If the reference becomes nul, the dialog is destroyed.

Parameters

<i>pDialog</i>	the dialog.
----------------	-------------

Returns

TRUE if the reference became nul, in which case the dialog must not be used anymore.

5.18.3.3 gboolean cairo_dock_remove_dialog_if_any_full (Icon * icon, gboolean bAll)

Unreference the dialogs pointing on an icon.

Parameters

<i>icon</i>	the icon you want to delete all dialogs from.
<i>bAll</i>	whether all dialogs should be removed or only the one that don't have interaction with the user.

Returns

TRUE if at least one dialog has been unreferenced.

5.18.3.4 CairoDialog* cairo_dock_build_dialog (CairoDialogAttribute * pAttribute, Icon * plcon, CairoContainer * pContainer)

Generic function to pop up a dialog.

Parameters

<i>pAttribute</i>	attributes of the dialog.
<i>plcon</i>	the icon that will hold the dialog.
<i>pContainer</i>	the container of the icon.

Returns

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

5.18.3.5 CairoDialog* cairo_dock_show_dialog_full (const gchar * cText, Icon * plcon, CairoContainer * pContainer, double fTimeLength, const gchar * clconPath, 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 optional.

Parameters

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

Returns

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

5.18.3.6 CairoDialog* cairo_dock_show_temporary_dialog_with_icon_printf (const gchar * cText, Icon * plcon, CairoContainer * pContainer, double fTimeLength, const gchar * clconPath, ...)

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

Parameters

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

Returns

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

5.18.3.7 CairoDialog CairoDialog* cairo_dock_show_temporary_dialog_with_icon (const gchar * *cText*, Icon * *pIcon*, CairoContainer * *pContainer*, double *fTimeLength*, const gchar * *clconPath*)

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

Parameters

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

Returns

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

5.18.3.8 CairoDialog* cairo_dock_show_temporary_dialog (const gchar * *cText*, Icon * *pIcon*, CairoContainer * *pContainer*, double *fTimeLength*)

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

Parameters

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

Returns

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

5.18.3.9 CairoDialog* cairo_dock_show_temporary_dialog_with_default_icon (const gchar * *cText*, Icon * *pIcon*, CairoContainer * *pContainer*, double *fTimeLength*)

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

Parameters

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

Returns

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

5.18.3.10 CairoDialog* cairo_dock_show_dialog_with_question (const gchar * *cText*, Icon * *pIcon*, CairoContainer * *pContainer*, const gchar * *clconPath*, 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

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

Returns

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

5.18.3.11 CairoDialog* cairo_dock_show_dialog_with_entry (const gchar * *cText*, Icon * *pIcon*, CairoContainer * *pContainer*, const gchar * *clconPath*, 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

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

Returns

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

5.18.3.12 CairoDialog* cairo_dock_show_dialog_with_value (const gchar * *cText*, Icon * *pIcon*, CairoContainer * *pContainer*, const gchar * *clconPath*, 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

<i>cText</i>	the message to display.
<i>pIcon</i>	the icon that will hold the dialog.
<i>pContainer</i>	the container of the icon.
<i>clconPath</i>	path to an icon to display in the margin.
<i>fValue</i>	initial value of the scale.
<i>fMaxValue</i>	maximum value of the scale.
<i>pActionFunc</i>	the callback.
<i>data</i>	data passed as a parameter of the callback.
<i>pFreeDataFunc</i>	function used to free the data.

Returns

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

5.18.3.13 int cairo_dock_show_dialog_and_wait (const gchar * *cText*, Icon * *pIcon*, CairoContainer * *pContainer*, double *fTimeLength*, const gchar * *clconPath*, GtkWidget * *pInteractiveWidget*)

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

Parameters

<i>cText</i>	the message to display.
<i>pIcon</i>	the icon that will hold the dialog.
<i>pContainer</i>	the container of the icon.
<i>fTimeLength</i>	time length of the dialog, or 0 for an unlimited dialog.
<i>clconPath</i>	path to an icon to display in the margin.
<i>pInteractive-Widget</i>	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.

5.18.3.14 gchar* cairo_dock_show_demand_and_wait (const gchar * *cMessage*, Icon * *pIcon*, CairoContainer * *pContainer*, const gchar * *clnitialAnswer*)

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

Parameters

<i>cMessage</i>	the message to display.
<i>pIcon</i>	the icon that will hold the dialog.
<i>pContainer</i>	the container of the icon.
<i>clnitialAnswer</i>	the initial value of the entry (can be NULL).

Returns

the text entered by the user, or NULL if he cancelled or if the dialog has been destroyed before.

5.18.3.15 `double cairo_dock_show_value_and_wait (const gchar * cMessage, Icon * pIcon, CairoContainer * pContainer, double fInitialValue, double fMaxValue)`

Pop up a dialog with an horizontal scale between 0 and *fMaxValue*, and 2 buttons ok/cancel, and block until the user makes its choice.

Parameters

<i>cMessage</i>	the message to display.
<i>pIcon</i>	the icon that will hold the dialog.
<i>pContainer</i>	the container of the icon.
<i>fInitialValue</i>	the initial value of the scale.
<i>fMaxValue</i>	the maximum value of the scale.

Returns

the value choosed by the user, or -1 if he cancelled or if the dialog has been destroyed before.

5.18.3.16 `int cairo_dock_ask_question_and_wait (const gchar * cQuestion, Icon * pIcon, CairoContainer * pContainer)`

Pop up a dialog with a question and 2 buttons yes/no, and block until the user makes its choice.

Parameters

<i>cQuestion</i>	the question to ask.
<i>pIcon</i>	the icon that will hold the dialog.
<i>pContainer</i>	the container of the icon.

Returns

GTK_RESPONSE_YES ou GTK_RESPONSE_NO according to the user's choice, or GTK_RESPONSE_NONE if the dialog has been destroyed before.

5.18.3.17 `gboolean cairo_dock_icon_has_dialog (Icon * pIcon)`

Test if an icon has at least one dialog.

Parameters

<i>pIcon</i>	the icon.
--------------	-----------

Returns

TRUE if the icon has one or more dialog(s).

5.18.3.18 `Icon* cairo_dock_get_dialogless_icon_full (CairoDock * pDock)`

Search the "the best icon possible" for a Dock to hold a general dialog.

Parameters

<i>pDock</i>	a dock (NULL to search inside the main dock).
--------------	---

Returns

an Icon, or NULL if the dock is empty.

5.18.3.19 CairoDialog* cairo_dock_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

<i>cMessage</i>	the message.
<i>fTimeLength</i>	life time of the dialog, in ms.

Returns

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

5.18.3.20 int cairo_dock_ask_general_question_and_wait (const gchar * *cQuestion*)

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

Parameters

<i>cQuestion</i>	the message.
------------------	--------------

Returns

GTK_RESPONSE_YES ou GTK_RESPONSE_NO according to the user's choice, or GTK_RESPONSE_NONE if the dialog has been destroyed before.

5.18.3.21 void cairo_dock_hide_dialog (CairoDialog * *pDialog*)

Hide a dialog.

Parameters

<i>pDialog</i>	the dialog.
----------------	-------------

5.18.3.22 void cairo_dock_unhide_dialog (CairoDialog * *pDialog*)

Show a dialog and give it focus.

Parameters

<i>pDialog</i>	the dialog.
----------------	-------------

5.18.3.23 void `cairo_dock_toggle_dialog_visibility` (`CairoDialog * pDialog`)

Toggle the visibility of a dialog.

Parameters

<code>pDialog</code>	the dialog.
----------------------	-------------

5.19 cairo-dock-dock-facility.h File Reference

Functions

- void `cairo_dock_update_dock_size` (`CairoDock *pDock`)
 void cairo_dock_reload_reflects_in_dock (`CairoDock *pDock`);
- `Icon *` `cairo_dock_calculate_dock_icons` (`CairoDock *pDock`)
- void `cairo_dock_show_subdock` (`Icon *pPointedIcon`, `CairoDock *pParentDock`)
- 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`)
 void cairo_dock_manage_mouse_position (`CairoDock *pDock`);
- `GList *` `cairo_dock_get_first_drawn_element_linear` (`GList *icons`)

5.19.1 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.

5.19.2 Function Documentation

5.19.2.1 void `cairo_dock_update_dock_size` (`CairoDock * pDock`)

`void cairo_dock_reload_reflects_in_dock` (`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

<code>pDock</code>	the dock.
--------------------	-----------

5.19.2.2 `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

<code>pDock</code>	the dock.
--------------------	-----------

Returns

the pointed icon, or NULL if none is pointed.

5.19.2.3 void cairo_dock_show_subdock (Icon * *pPointedIcon*, CairoDock * *pParentDock*)

Pop up a sub-dock.

Parameters

<i>pPointedIcon</i>	icon pointing on the sub-dock.
<i>pParentDock</i>	dock containing the icon.

5.19.2.4 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

<i>pIconList</i>	a list of icons.
<i>fFlatDockWidth</i>	width of all the icons placed next to each other.
<i>iXOffset</i>	an offset on the position of the first icon.

5.19.2.5 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

<i>pDock</i>	a linear dock.
--------------	----------------

Returns

the pointed icon, or NULL if none is pointed.

5.19.2.6 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

<i>pDock</i>	a linear dock.
--------------	----------------

Returns

the dock's width.

5.19.2.7 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

<i>pDock</i>	a linear dock.
--------------	----------------

5.19.2.8 void `cairo_dock_check_can_drop_linear (CairoDock * pDock)`

void `cairo_dock_manage_mouse_position (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

<i>pDock</i>	a linear dock.
--------------	----------------

5.19.2.9 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

<i>icons</i>	a list of icons of a linear dock.
--------------	-----------------------------------

Returns

the element of the list that contains the first icon to draw.

5.20 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 [CAIRO_DOCK_IS_DOCK](#)(pContainer)
- #define [CAIRO_DOCK](#)(pDock)
- #define [cairo_dock_insert_icon_in_dock](#)(icon, pDock, bAnimated)
- #define [cairo_dock_remove_one_icon_from_dock](#)(pDock, icon)
- #define [cairo_dock_remove_icon_from_dock](#)(pDock, icon)
- #define [cairo_dock_add_new_launcher_by_uri](#)(cExternDesktopFileURI, pReceivingDock, fOrder)
- #define [cairo_dock_add_new_launcher_by_type](#)(iType, pReceivingDock, fOrder)

Functions

- void [cairo_dock_insert_icon_in_dock_full](#) (Icon *icon, CairoDock *pDock, gboolean bAnimated, gboolean bInsertSeparator, GCompareFunc pCompareFunc)
- gboolean [cairo_dock_detach_icon_from_dock_full](#) (Icon *icon, CairoDock *pDock, gboolean bCheckUnusedSeparator)

- void `cairo_dock_remove_automatic_separators` (CairoDock *pDock)
- void `cairo_dock_insert_automatic_separators_in_dock` (CairoDock *pDock)
- void `cairo_dock_remove_icons_from_dock` (CairoDock *pDock, CairoDock *pReceivingDock, const gchar *cReceivingDockName)

5.20.1 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.

5.20.2 Macro Definition Documentation

5.20.2.1 #define CAIRO_DOCK_IS_DOCK(*pContainer*)

Say if a Container is a Dock.

Parameters

<i>pContainer</i>	the container.
-------------------	----------------

Returns

TRUE if the container is a Dock.

5.20.2.2 #define CAIRO_DOCK(*pDock*)

Cast a Container into a Dock.

Parameters

<i>pDock</i>	the container to consider as a dock.
--------------	--------------------------------------

Returns

the dock.

5.20.2.3 #define cairo_dock_insert_icon_in_dock(*icon*, *pDock*, *bAnimated*)

Insert an icon into a dock. Insert an automatic separator if needed. Do nothing if the icon already exists inside the dock.

Parameters

<i>icon</i>	the icon to be inserted. It should have been filled beforehand.
<i>pDock</i>	the dock to insert inside.
<i>bAnimated</i>	TRUE to arm the icon's animation for insertion.

5.20.2.4 #define cairo_dock_remove_one_icon_from_dock(*pDock*, *icon*)

Completely remove an icon from the dock, that is to say detach the icon, and remove all links with Cairo-Dock : its .desktop is deleted, its module is deactivated, and its Xid is removed from the Taskbar (its class is handled too). Unnecessary separators are not tested. The icon is not yet destroyed, but loses its sub-dock in case of a container launcher.

Parameters

<i>pDock</i>	the dock containing the icon, or NULL if the icon is already detached.
<i>icon</i>	the icon to be removed.

5.20.2.5 #define cairo_dock_remove_icon_from_dock(*pDock*, *icon*)

Completely remove an icon from the dock, that is to say detach the icon, and remove all links with Cairo-Dock : its .desktop is deleted, its module is deactivated, and its Xid is removed from the Taskbar (its class is handled too). Unnecessary separators are removed as well. The icon is not yet destroyed, but loses its sub-dock in case of a container launcher.

Parameters

<i>pDock</i>	the dock containing the icon, or NULL if the icon is already detached.
<i>icon</i>	the icon to be removed.

5.20.2.6 #define cairo_dock_add_new_launcher_by_uri(*cExternDesktopFileURI*, *pReceivingDock*, *fOrder*)

Add a launcher from a common desktop file : create and add the corresponding .desktop file with the others, load the corresponding icon, and insert it inside a dock with an animation.

Parameters

<i>cExternDesktopFileURI</i>	path to a desktop file.
<i>pReceivingDock</i>	the dock that will hold the new launcher.
<i>fOrder</i>	the order of the icon inside the dock.

Returns

the newly created Icon corresponding to the file, or NULL if an error occurred.

5.20.2.7 #define cairo_dock_add_new_launcher_by_type(*iType*, *pReceivingDock*, *fOrder*)

Add an empty default launcher of a given type : create and add the corresponding .desktop file with the others, load the corresponding icon, and insert it inside a dock with an animation. The launcher is then suitable for being edited by the user to add real properties.

Parameters

<i>iType</i>	type of the launcher.
<i>pReceivingDock</i>	the dock that will hold the new launcher.
<i>fOrder</i>	the order of the icon inside the dock.
<i>iGroup</i>	the group it will belong to

Returns

the newly created Icon corresponding to the type, or NULL if an error occurred.

5.20.3 Function Documentation

5.20.3.1 void `cairo_dock_insert_icon_in_dock_full` (**Icon** * *icon*, **CairoDock** * *pDock*, gboolean *bAnimated*, gboolean *bInsertSeparator*, GCompareFunc *pCompareFunc*)

Insert an icon into a dock. Do nothing if the icon already exists inside the dock.

Parameters

<i>icon</i>	the icon to be inserted. It should have been filled beforehand.
<i>pDock</i>	the dock to insert inside.
<i>bAnimated</i>	TRUE to arm the icon's animation for insertion.
<i>bInsertSeparator</i>	TRUE to insert an automatic separator if needed.
<i>pCompareFunc</i>	a sorting function to place the new icon amongst the others, or NULL to sort by group/order.

5.20.3.2 gboolean `cairo_dock_detach_icon_from_dock_full` (**Icon** * *icon*, **CairoDock** * *pDock*, gboolean *bCheckUnusedSeparator*)

Detach an icon from its dock. The icon is not destroyed, and can be directly re-inserted in another container; it keeps its sub-dock, but loses its dialogs. Do nothing if the icon doesn't exist inside the dock.

Parameters

<i>icon</i>	the icon to detach.
<i>pDock</i>	the dock containing the icon.
<i>bCheckUnusedSeparator</i>	TRUE to check and remove unnecessary separators.

Returns

TRUE if the icon has been detached.

5.20.3.3 void `cairo_dock_remove_automatic_separators` (**CairoDock** * *pDock*)

Remove and destroy all automatic separators inside a dock.

Parameters

<i>pDock</i>	the dock.
--------------	-----------

5.20.3.4 void `cairo_dock_insert_automatic_separators_in_dock` (**CairoDock** * *pDock*)

Add automatic separators between the different groups of icons inside a dock.

Parameters

<i>pDock</i>	the dock.
--------------	-----------

5.20.3.5 void `cairo_dock_remove_icons_from_dock` (CairoDock * *pDock*, CairoDock * *pReceivingDock*, const gchar * *cReceivingDockName*)

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

<i>pDock</i>	a dock.
<i>pReceivingDock</i>	the dock that will receive the icons, or NULL to destroy and remove the icons.
<i>cReceivingDockName</i>	name of the receiving dock.

5.21 cairo-dock-dock-manager.h File Reference

Enumerations

- enum [GldIconSize](#)

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

- CairoDock * [cairo_dock_create_dock](#) (const gchar *cDockName)
- CairoDock * [cairo_dock_create_subdock](#) (const gchar *cDockName, const gchar *cRendererName, CairoDock *pParentDock, GList *plconList)
- void [cairo_dock_destroy_dock](#) (CairoDock *pDock, const gchar *cDockName)
- const gchar * [cairo_dock_search_dock_name](#) (CairoDock *pDock)
- gchar * [cairo_dock_get_readable_name_for_fock](#) (CairoDock *pDock)
- CairoDock * [cairo_dock_search_dock_from_name](#) (const gchar *cDockName)
- Icon * [cairo_dock_search_icon_pointing_on_dock](#) (CairoDock *pDock, CairoDock **pParentDock)
- void [cairo_dock_rename_dock](#) (const gchar *cDockName, CairoDock *pDock, const gchar *cNewName)
- void [cairo_dock_foreach_docks](#) (GFunc pFunction, gpointer pUserData)
- void [cairo_dock_foreach_root_docks](#) (GFunc pFunction, gpointer pUserData)
- void [cairo_dock_foreach_icons_in_docks](#) (CairoDockForeachIconFunc pFunction, gpointer pUserData)
- void [cairo_dock_hide_parent_dock](#) (CairoDock *pDock)
- gboolean [cairo_dock_hide_child_docks](#) (CairoDock *pDock)
- void [cairo_dock_reload_buffers_in_all_docks](#) (gboolean bUpdateIconSize)
- void [cairo_dock_set_all_views_to_default](#) (int iDockType)
- void [cairo_dock_update_all_docks_size](#) (void);*
- void [cairo_dock_reload_one_root_dock](#) (const gchar *cDockName, CairoDock *pDock)
- void [cairo_dock_remove_root_dock_config](#) (const gchar *cDockName)
- void [cairo_dock_add_root_dock_config_for_name](#) (const gchar *cDockName)
- gchar * [cairo_dock_add_root_dock_config](#) (void)
- void [cairo_dock_set_dock_visibility](#) (CairoDock *pDock, CairoDockVisibility iVisibility)

5.21.1 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.

5.21.2 Enumeration Type Documentation

5.21.2.1 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, CairoDock}

5.21.3 Function Documentation

5.21.3.1 CairoDock* cairo_dock_create_dock (const gchar * cDockName)

Create a new root dock.

Parameters

<i>cDockName</i>	name (= ID) of the dock. If the name is already used, the corresponding dock is returned.
------------------	---

Returns

the dock, to destroy with [cairo_dock_destroy_dock](#)

5.21.3.2 CairoDock* cairo_dock_create_subdock (const gchar * cDockName, const gchar * cRendererName, CairoDock * pParentDock, GList * plconList)

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 freed after that. The buffers of each icon are loaded, so they just need to have an image filename and a name.

Parameters

<i>cDockName</i>	desired name for the new dock.
<i>cRendererName</i>	name of a renderer. If NULL, the default renderer will be applied.
<i>pParentDock</i>	the parent dock.
<i>plconList</i>	a list of icons that will be loaded and inserted into the new dock.

Returns

the newly allocated dock.

5.21.3.3 void `cairo_dock_destroy_dock` (`CairoDock * pDock`, `const gchar * cDockName`)

Destroy a dock and its icons.

Parameters

<i>pDock</i>	the dock.
<i>cDockName</i>	name for the dock.

5.21.3.4 const gchar* `cairo_dock_search_dock_name` (`CairoDock * pDock`)

Search the name of a Dock. It does a linear search in the table of Docks.

Parameters

<i>pDock</i>	the dock.
--------------	-----------

Returns

the name of the dock, or NULL if not found.

5.21.3.5 gchar* `cairo_dock_get_readable_name_for_fock` (`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 [cairo_dock_search_dock_name](#) for them.

Parameters

<i>pDock</i>	the dock.
--------------	-----------

Returns

the readable name of the dock, or NULL if not found. Free it when you're done.

5.21.3.6 CairoDock* `cairo_dock_search_dock_from_name` (`const gchar * cDockName`)

Search a Dock from a given name. It does a fast search in the table of Docks.

Parameters

<i>cDockName</i>	the name of the dock.
------------------	-----------------------

Returns

the dock that has been registered under this name, or NULL if none exists.

5.21.3.7 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

<i>pDock</i>	the dock.
<i>pParentDock</i>	if not NULL, this will be filled with the dock containing the icon.

Returns

the icon pointing on the dock.

5.21.3.8 void `cairo_dock_rename_dock` (const gchar * *cDockName*, CairoDock * *pDock*, const gchar * *cNewName*)

Rename a dock. Update the container's name of all of its icons.

Parameters

<i>cDockName</i>	name of the dock.
<i>pDock</i>	the dock (optional).
<i>cNewName</i>	the new name.

5.21.3.9 void `cairo_dock_foreach_docks` (GHFunc *pFunction*, gpointer *pUserData*)

Execute an action on all docks.

Parameters

<i>pFunction</i>	the action.
<i>pUserData</i>	data passed to the callback.

5.21.3.10 void `cairo_dock_foreach_root_docks` (GFunc *pFunction*, gpointer *pUserData*)

Execute an action on all main docks.

Parameters

<i>pFunction</i>	the action.
<i>pUserData</i>	data passed to the callback.

5.21.3.11 void `cairo_dock_foreach_icons_in_docks` (CairoDockForeachIconFunc *pFunction*, gpointer *pUserData*)

Execute an action on all icons being inside a dock.

Parameters

<i>pFunction</i>	the action.
<i>pUserData</i>	data passed to the callback.

5.21.3.12 void `cairo_dock_hide_parent_dock` (CairoDock * *pDock*)

Recursively hides all the parent docks of a sub-dock.

Parameters

<i>pDock</i>	the (sub)dock.
--------------	----------------

5.21.3.13 gboolean `cairo_dock_hide_child_docks` (CairoDock * *pDock*)

Recursively hides all the sub-docks of a given dock.

Parameters

<i>pDock</i>	the dock.
--------------	-----------

Returns

TRUE if a sub-dock has been hidden.

5.21.3.14 void `cairo_dock_reload_buffers_in_all_docks (gboolean bUpdateIconSize)`

(Re)load all buffers of all icons in all docks.

5.21.3.15 void `cairo_dock_set_all_views_to_default (int iDockType)`

void `cairo_dock_update_all_docks_size (void);`

void `cairo_dock_reset_all_views (void);`

5.21.3.16 void `cairo_dock_reload_one_root_dock (const gchar * cDockName, CairoDock * pDock)`

Reload the config of a root dock and update it accordingly.

Parameters

<i>cDockName</i>	name of the dock.
<i>pDock</i>	the dock.

5.21.3.17 void `cairo_dock_remove_root_dock_config (const gchar * cDockName)`

Delete the config of a root dock. Doesn't delete the dock (use [cairo_dock_destroy_dock](#) for that), but if it was empty, it won't be created the next time you restart Cairo-Dock.

Parameters

<i>cDockName</i>	name of the dock.
------------------	-------------------

5.21.3.18 void `cairo_dock_add_root_dock_config_for_name (const gchar * cDockName)`

Add a config file for a root dock. Does not create the dock (use [cairo_dock_create_dock](#) for that). If the config file already exists, it is overwritten (use [cairo_dock_search_dock_from_name](#) to check if the dock already exists).

Parameters

<i>cDockName</i>	name of the dock.
------------------	-------------------

5.21.3.19 gchar* `cairo_dock_add_root_dock_config (void)`

Add a config file for a new root dock. Does not create the dock (use [cairo_dock_create_dock](#) for that).

Returns

the unique name for the new dock, to be passed to [cairo_dock_create_dock](#).

5.21.3.20 void `cairo_dock_set_dock_visibility (CairoDock * pDock, CairoDockVisibility iVisibility)`

Set the visibility of a root dock. Perform all the necessary actions.

Parameters

<i>pDock</i>	a root dock.
<i>iVisibility</i>	its new visibility.

5.22 cairo-dock-draw-opengl.h File Reference

Macros

- #define `cairo_dock_create_texture_from_image`(*clmagePath*)
- #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 gchar* **pTextureRaw*, *int* *iWidth*, *int* *iHeight*)
- GLuint `cairo_dock_create_texture_from_image_full` (*const gchar* **clmagePath*, double **fImageWidth*, double **fImageHeight*)
- void `cairo_dock_update_icon_texture` (*Icon* **pIcon*)
- void `cairo_dock_update_label_texture` (*Icon* **pIcon*)
- void `cairo_dock_update_quick_info_texture` (*Icon* **pIcon*)

5.22.1 Detailed Description

This class provides some useful functions to draw with OpenGL.

5.22.2 Macro Definition Documentation

5.22.2.1 #define `cairo_dock_create_texture_from_image`(*clmagePath*)

Load an image on the dock into an OpenGL texture. The texture will have the same size as the image.

Parameters

<i>clmagePath</i>	path to an image.
-------------------	-------------------

Returns

the newly allocated texture, to be destroyed with `_cairo_dock_delete_texture`.

5.22.2.2 `#define _cairo_dock_delete_texture(iTexture)`

Delete an OpenGL texture from the Graphic Card.

Parameters

<i>iTexture</i>	variable containing the ID of a texture.
-----------------	--

5.22.2.3 `#define _cairo_dock_enable_texture(...)`

Enable texture drawing.

5.22.2.4 `#define _cairo_dock_disable_texture(...)`

Disable texture drawing.

5.22.2.5 `#define _cairo_dock_set_alpha(fAlpha)`

Set the alpha channel to a current value, other channels are set to 1.

Parameters

<i>fAlpha</i>	alpha
---------------	-------

5.22.2.6 `#define _cairo_dock_set_blend_source(...)`

Set the color blending to overwrite.

5.22.2.7 `#define _cairo_dock_set_blend_alpha(...)`

Set the color blending to mix, for premultiplied texture.

5.22.2.8 `#define _cairo_dock_set_blend_over(...)`

Set the color blending to mix.

5.22.2.9 `#define _cairo_dock_set_blend_pbuffer(...)`

Set the color blending to mix on a pbuffer.

5.22.2.10 `#define _cairo_dock_apply_texture_at_size(iTexture, w, h)`

Draw a texture centered on the current point, at a given size.

Parameters

<i>iTexture</i>	the texture
<i>w</i>	width
<i>h</i>	height

5.22.2.11 #define _cairo_dock_apply_texture(*iTexture*)

Apply a texture centered on the current point and at the given scale.

Parameters

<i>iTexture</i>	the texture
-----------------	-------------

5.22.2.12 #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

<i>iTexture</i>	the texture
<i>w</i>	width
<i>h</i>	height
<i>fAlpha</i>	the transparency, between 0 and 1.

5.22.3 Function Documentation

5.22.3.1 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

<i>icon</i>	the icon to draw.
<i>pDock</i>	the dock containing the icon.
<i>fDockMagnitude</i>	current magnitude of the dock.
<i>bUseText</i>	TRUE to draw the labels.

5.22.3.2 GLuint cairo_dock_create_texture_from_surface (cairo_surface_t * *plmageSurface*)

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

<i>plmageSurface</i>	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`.

5.22.3.3 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

<i>pTextureRaw</i>	a buffer of pixels.
<i>iWidth</i>	width of the image.
<i>iHeight</i>	height of the image.

Returns

the newly allocated texture, to be destroyed with `_cairo_dock_delete_texture`.

5.22.3.4 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

<i>cImagePath</i>	path to an image.
<i>fImageWidth</i>	pointer that will be filled with the width of the image.
<i>fImageHeight</i>	pointer that will be filled with the height of the image.

Returns

the newly allocated texture, to be destroyed with `_cairo_dock_delete_texture`.

5.22.3.5 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

<i>pIcon</i>	the icon.
--------------	-----------

5.22.3.6 void cairo_dock_update_label_texture (Icon * *pIcon*)

Update the icon's label texture with its current label surface.

Parameters

<i>pIcon</i>	the icon.
--------------	-----------

5.22.3.7 void cairo_dock_update_quick_info_texture (Icon * *pIcon*)

Update the icon's quick-info texture with its current quick-info surface.

Parameters

<i>pIcon</i>	the icon.
--------------	-----------

5.23 cairo-dock-draw.h File Reference

Macros

- #define `cairo_dock_erase_cairo_context(pCairoContext)`

Functions

- `cairo_t * cairo_dock_create_drawing_context_generic (CairoContainer *pContainer)`
CONTEXT !!! ///.
- `cairo_t * cairo_dock_create_drawing_context_on_container (CairoContainer *pContainer)`
- `cairo_t * cairo_dock_create_drawing_context_on_area (CairoContainer *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 fDockMagnitude, gboolean bUseText)`
- `void cairo_dock_draw_string (cairo_t *pCairoContext, CairoDock *pDock, double fStringLineWidth, gboolean blsLoop, gboolean bForceConstantSeparator)`

5.23.1 Detailed Description

This class provides some useful functions to draw with libcairo.

5.23.2 Macro Definition Documentation

5.23.2.1 #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

<i>pCairoContext</i>	a drawing context.
----------------------	--------------------

5.23.3 Function Documentation

5.23.3.1 `cairo_t* cairo_dock_create_drawing_context_generic (CairoContainer * pContainer)`

CONTEXT !!! ///.

Create a generic drawing context, to be used as a source context (for instance, for creating a surface).

Parameters

<i>pContainer</i>	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.

5.23.3.2 `cairo_t* cairo_dock_create_drawing_context_on_container (CairoContainer * pContainer)`

Create a drawing context to draw on a container. It handles fake transparency.

Parameters

<i>pContainer</i>	the container on which you want to draw.
-------------------	--

Returns

the newly allocated context, to be destroyed with 'cairo_destroy'.

5.23.3.3 `cairo_t* cairo_dock_create_drawing_context_on_area (CairoContainer * pContainer, GdkRectangle * pArea, double * fBgColor)`

Create a drawing context to draw on a part of a container. It handles fake transparency.

Parameters

<i>pContainer</i>	the container on which you want to draw
<i>pArea</i>	part of the container to draw.
<i>fBgColor</i>	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'.

5.23.3.4 `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

<i>pCairoContext</i>	a drawing context; the current matrix is not altered, but the current path is.
<i>fRadius</i>	radius if the corners.
<i>fLineWidth</i>	width of the line.
<i>fFrameWidth</i>	width of the rectangle, without the corners.
<i>fFrameHeight</i>	height of the rectangle, including the corners.

5.23.3.5 `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

<i>icon</i>	the icon to draw.
<i>pDock</i>	the dock containing the icon.
<i>pCairoContext</i>	a context on the dock, not altered by the function.

5.23.3.6 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

<code>icon</code>	the icon to draw.
<code>pDock</code>	the dock containing the icon.
<code>pCairoContext</code>	a context on the dock, it is altered by the function.
<code>fDockMagnitude</code>	current magnitude of the dock.
<code>bUseText</code>	TRUE to draw the labels.

5.23.3.7 void `cairo_dock_draw_string` (`cairo_t * pCairoContext`, `CairoDock * pDock`, double `fStringLineWidth`, gboolean `blsLoop`, gboolean `bForceConstantSeparator`)

Draw a string linking the center of all the icons of a dock.

Parameters

<code>pCairoContext</code>	a context on the dock, not altered by the function.
<code>pDock</code>	the dock.
<code>fStringLineWidth</code>	width of the line.
<code>blsLoop</code>	TRUE to loop (link the last icon to the first one).
<code>bForceConstantSeparator</code>	TRUE to consider separators having a constant size.

5.24 cairo-dock-emblem.h File Reference

Data Structures

- struct [_CairoEmblem](#)

Definition of an Emblem. You shouldn't access any of its fields directly.

Macros

- #define [cairo_dock_set_emblem_position](#)(pEmblem, pos)

Enumerations

- enum [CairoEmblemPosition](#)

Available position of the emblem on the icon.

Functions

- `CairoEmblem * cairo_dock_make_emblem` (const gchar *cImageFile, `Icon *plcon`)
- `CairoEmblem * cairo_dock_make_emblem_from_surface` (`cairo_surface_t *pSurface`, int iSurfaceWidth, int iSurfaceHeight, `Icon *plcon`)
- `CairoEmblem * cairo_dock_make_emblem_from_texture` (GLuint iTexture, `Icon *plcon`)
- void [cairo_dock_free_emblem](#) (`CairoEmblem *pEmblem`)
- void [cairo_dock_draw_emblem_on_icon](#) (`CairoEmblem *pEmblem`, `Icon *plcon`, `CairoContainer *pContainer`)

5.24.1 Detailed Description

This class defines Emblems, that are small images superimposed on the icon at a given place.

The emblem is drawn directly on the icon, so it modifies its surface/texture itself. Thus, to remove an emblem, you have to set the original image on the icon.

Emblems can be placed in the corners of the icon, or in the middle of it.

Usage : create an Emblem with [cairo_dock_make_emblem](#) set its position with [cairo_dock_set_emblem_position](#) you can then render the emblem on the icon with [cairo_dock_draw_emblem_on_icon](#). free the emblem when you're done with [cairo_dock_free_emblem](#)

An emblem can be used as many times as you want on any icon. The only limitation is that an emblem uses either Cairo or OpenGL, this is decided at the creation time; so the icons you draw the emblem on must be drawn with the same rendering.

5.24.2 Macro Definition Documentation

5.24.2.1 #define `cairo_dock_set_emblem_position(pEmblem, pos)`

Set the position of an emblem.

Parameters

<i>pEmblem</i>	the emblem
<i>pos</i>	the position (a CairoEmblemPosition)

5.24.3 Function Documentation

5.24.3.1 `CairoEmblem* cairo_dock_make_emblem (const gchar * cImageFile, Icon * pIcon)`

Create an emblem from an image, that suits the given icon and container. If the image is given by its sole name, it is searched inside the current theme root folder.

Parameters

<i>cImageFile</i>	an image.
<i>pIcon</i>	an icon.

Returns

the newly allocated emblem.

5.24.3.2 `CairoEmblem* cairo_dock_make_emblem_from_surface (cairo_surface_t * pSurface, int iSurfaceWidth, int iSurfaceHeight, Icon * pIcon)`

Create an emblem from an existing surface. The surface is appropriated by the emblem, so if you free it with [cairo_dock_free_emblem](#), it will also free the surface. Use `g_free` to destroy the emblem if you don't want the surface to be destroyed with.

Parameters

<i>pSurface</i>	a surface.
<i>iSurfaceWidth</i>	width of the surface, 0 means it has the same width as the icon.
<i>iSurfaceHeight</i>	height of the surface, 0 means it has the same height as the icon.
<i>pIcon</i>	an icon.

Returns

the newly allocated emblem.

5.24.3.3 CairoEmblem* cairo_dock_make_emblem_from_texture (GLuint *iTexture*, Icon * *pIcon*)

Create an emblem from an existing texture. The texture is appropriated by the emblem, so if you free it with [cairo_dock_free_emblem](#), it will also free the texture. Use `g_free` to destroy the emblem if you don't want the texture to be destroyed with.

Parameters

<i>iTexture</i>	a texture.
<i>pIcon</i>	an icon.

Returns

the newly allocated emblem.

5.24.3.4 void cairo_dock_free_emblem (CairoEmblem * *pEmblem*)

Destroy an emblem and all its allocated resources.

Parameters

<i>pEmblem</i>	the emblem
----------------	------------

5.24.3.5 void cairo_dock_draw_emblem_on_icon (CairoEmblem * *pEmblem*, Icon * *pIcon*, CairoContainer * *pContainer*)

Permanently draw an emblem on an icon.

Parameters

<i>pEmblem</i>	the emblem
<i>pIcon</i>	an icon
<i>pContainer</i>	its container, or NULL if the icon is not yet inside a container

5.25 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 iNewIconType, [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](#) **cIconName, [gboolean](#) *bIsDirectory, int *iVolumeID, double *fOrder, [CairoDockFMSortType](#) iSortType)
- [gboolean](#) [cairo_dock_fm_get_file_properties](#) (const [gchar](#) *cURI, [guint64](#) *iSize, [time_t](#) *iLastModificationTime, [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](#) *cMountedURI, [CairoDockFMMonitorCallback](#) pCallback, [gpointer](#) data)
- [gboolean](#) [cairo_dock_fm_remove_monitor_full](#) (const [gchar](#) *cURI, [gboolean](#) bDirectory, const [gchar](#) *cMountedURI)
- [gboolean](#) [cairo_dock_fm_mount_full](#) (const [gchar](#) *cURI, int iVolumeID, [CairoDockFMMountCallback](#) pCallback, [gpointer](#) user_data)
- [gboolean](#) [cairo_dock_fm_unmount_full](#) (const [gchar](#) *cURI, int iVolumeID, [CairoDockFMMountCallback](#) pCallback, [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, [CairoContainer](#) *pContainer, [CairoDockFMSortType](#) iFileSortType)
- int [cairo_dock_get_file_size](#) (const [gchar](#) *cFilePath)

5.25.1 Detailed Description

This class manages the integration into the desktop environment, which includes :

- the VFS (Virtual File System)
- the various desktop-related tools.

5.25.2 Function Documentation

5.25.2.1 void [cairo_dock_fm_register_vfs_backend](#) ([CairoDockDesktopEnvBackend](#) * [pVFSBackend](#))

Register a environment backend, overwriting any previous backend.

5.25.2.2 `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.

5.25.2.3 `gsize cairo_dock_fm_measure_directory (const gchar * cBaseURI, gint iCountType, gboolean bRecursive, gint * pCancel)`

Measure a directory (number of files or total size).

5.25.2.4 `gboolean cairo_dock_fm_get_file_info (const gchar * cBaseURI, gchar ** cName, gchar ** cURI, gchar ** clconName, gboolean * blsDirectory, int * iVolumeID, double * fOrder, CairoDockFMSortType iSortType)`

Get the main info to represent a file.

5.25.2.5 `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.

5.25.2.6 `gboolean cairo_dock_fm_launch_uri (const gchar * cURI)`

Open a file with the default application.

5.25.2.7 `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.

5.25.2.8 `gboolean cairo_dock_fm_remove_monitor_full (const gchar * cURI, gboolean bDirectory, const gchar * cMountedURI)`

Remove a monitor on an URI.

5.25.2.9 `gboolean cairo_dock_fm_mount_full (const gchar * cURI, int iVolumeID, CairoDockFMMountCallback pCallback, gpointer user_data)`

Mount a point.

5.25.2.10 `gboolean cairo_dock_fm_unmount_full (const gchar * cURI, int iVolumeID, CairoDockFMMountCallback pCallback, gpointer user_data)`

Unmount a point.

5.25.2.11 `gboolean cairo_dock_fm_is_mounted (const gchar * cURI, gboolean * blsMounted)`

Say if a point is currently mounted.

5.25.2.12 `gboolean cairo_dock_fm_can_eject (const gchar * cURI)`

Say if a point can be ejected (like a CD player).

5.25.2.13 `gboolean cairo_dock_fm_eject_drive (const gchar * cURI)`

Eject a drive, like a CD player.

5.25.2.14 `gboolean cairo_dock_fm_delete_file (const gchar * cURI, gboolean bNoTrash)`

Delete a file.

5.25.2.15 `gboolean cairo_dock_fm_rename_file (const gchar * cOldURI, const gchar * cNewName)`

Rename a file.

5.25.2.16 `gboolean cairo_dock_fm_move_file (const gchar * cURI, const gchar * cDirectoryURI)`

Move a file.

5.25.2.17 `gboolean cairo_dock_fm_create_file (const gchar * cURI, gboolean bDirectory)`

Create a new file.

5.25.2.18 `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}.

5.25.2.19 `gboolean cairo_dock_fm_empty_trash (void)`

Empty the Trash.

5.25.2.20 `gchar* cairo_dock_fm_get_trash_path (const gchar * cNearURI, gchar ** cFileInfoPath)`

Get the path to the Trash.

5.25.2.21 `gchar* cairo_dock_fm_get_desktop_path (void)`

Get the path to the Desktop.

5.25.2.22 `gboolean cairo_dock_fm_logout (void)`

Raise the logout panel.

5.25.2.23 `gboolean cairo_dock_fm_shutdown (void)`

Raise the shutdown panel.

5.25.2.24 `gboolean cairo_dock_fm_reboot (void)`

Raise the reboot panel.

5.25.2.25 `gboolean cairo_dock_fm_lock_screen (void)`

Lock the screen.

5.25.2.26 `gboolean cairo_dock_fm_setup_time (void)`

Raise the panel to configure the time.

5.25.2.27 `gboolean cairo_dock_fm_show_system_monitor (void)`

Raise the default system monitor.

5.25.2.28 `Icon* cairo_dock_fm_create_icon_from_URI (const gchar * cURI, CairoContainer * pContainer, CairoDockFMSortType iFileSortType)`

Create an Icon representing a given URI.

5.25.2.29 `int cairo_dock_get_file_size (const gchar * cFilePath)`

Get the size of a local file.

Parameters

<code><i>cFilePath</i></code>	path of a file on the hard disk.
-------------------------------	----------------------------------

Returns

the size of the file, or 0 if it doesn't exist.

5.26 cairo-dock-gauge.h File Reference

Typedefs

- typedef struct `_CairoGaugeAttribute` [CairoGaugeAttribute](#)

Attributes of a Gauge.

5.26.1 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](#).

5.27 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.

5.27.1 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](#).

5.27.2 Enumeration Type Documentation

5.27.2.1 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.

5.28 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_THEME_LIST_ENTRY,`
 - `CAIRO_DOCK_WIDGET_THEME_SELECTOR,`
 - `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_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_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_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.

5.28.1 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 CairoDockGUIWidgetType 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 appearance 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.

5.28.2 Enumeration Type Documentation

5.28.2.1 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_THEME_LIST_ENTRY same but with a combo-entry to let the user enter any text.

CAIRO_DOCK_WIDGET_THEME_SELECTOR list of dock themes, sortable by name, rating, and sobriety.

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_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_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_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.

5.29 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)
- #define [cairo_dock_build_generic_gui](#)(cConfFilePath, cGettextDomain, cTitle, iWidth, iHeight, pAction, pUserData, pFreeUserData)

Typedefs

- typedef gboolean(* [CairoDockApplyConfigFunc](#))(gpointer data)
Definition of the callback called when the user apply the config panel.

Functions

- [CairoDockGroupKeyWidget](#) * [cairo_dock_get_group_key_widget_from_name](#) ([CairoDockModuleInstance](#) *pModuleInstance, const gchar *cGroupName, const gchar *cKeyName)
- GtkWidget * [cairo_dock_get_widget_from_name](#) ([CairoDockModuleInstance](#) *pModuleInstance, const gchar *cGroupName, const gchar *cKeyName)
- 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_PRINTF(2)
- void GtkWidget * [cairo_dock_build_generic_gui_window](#) (const gchar *cTitle, int iWidth, int iHeight, [CairoDockApplyConfigFunc](#) pAction, gpointer pUserData, GFreeFunc pFreeUserData)
- void [cairo_dock_reload_generic_gui](#) (GtkWidget *pWindow)

5.29.1 Detailed Description

This class provides useful functions to build config panels from keyfiles.

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 character 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.

The class defines the interface that a backend to the main GUI of Cairo-Dock should implement. It also provides a useful function to easily build a window from a conf file : [cairo_dock_build_generic_gui](#)

5.29.2 Macro Definition Documentation

5.29.2.1 #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

<i>pModule-Instance</i>	an instance of a module.
-------------------------	--------------------------

5.29.2.2 `#define cairo_dock_build_generic_gui(cConfFilePath, cGettextDomain, cTitle, iWidth, iHeight, pAction, pUserData, pFreeUserData)`

Load a conf file into a generic window, that contains 2 buttons apply and quit, and a statusbar. If no callback is provided, the window is blocking, until the user press one of the button; in this case, TRUE is returned if ok was pressed.

Parameters

<i>cConfFilePath</i>	conf file to load into the window.
<i>cGettextDomain</i>	translation domain.
<i>cTitle</i>	title to set to the window.
<i>iWidth</i>	width of the window.
<i>iHeight</i>	height of the window.
<i>pAction</i>	callback to be called when the apply button is pressed, or NULL.
<i>pUserData</i>	data passed to the previous callback, or NULL.
<i>pFreeUserData</i>	callback called when the window is destroyed, to free the previous data, or NULL.

Returns

the new window.

5.29.3 Function Documentation

5.29.3.1 `CairoDockGroupKeyWidget* cairo_dock_get_group_key_widget_from_name (CairoDockModuleInstance * pModuleInstance, const gchar * cGroupName, const gchar * cKeyName)`

Retrieve the group-key widget in the current config panel, corresponding to the (group,key) pair in its conf file.

Parameters

<i>pModule-Instance</i>	the applet making the demand.
<i>cGroupName</i>	name of the group in the conf file.
<i>cKeyName</i>	name of the key in the conf file.

Returns

the group-key widget that match the group and key, or NULL if none was found.

5.29.3.2 `GtkWidget* cairo_dock_get_widget_from_name (CairoDockModuleInstance * pModuleInstance, const gchar * cGroupName, const gchar * cKeyName)`

A mere wrapper around the previous function, that returns directly the GTK widget corresponding to the (group,key). Note that empty widgets will return NULL, so you can't you can't distinguish between an empty widget and an inexisant widget.

Parameters

<i>pModule-Instance</i>	the applet making the demand.
<i>cGroupName</i>	name of the group in the conf file.
<i>cKeyName</i>	name of the key in the conf file.

Returns

the widget that match the group and key, or NULL if the widget is empty or if none was found.

5.29.3.3 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

<i>pWindow</i>	window where the message should be displayed, or NULL to target the config panel.
<i>cMessage</i>	the message.

5.29.3.4 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

<i>pWindow</i>	window where the message should be displayed, or NULL to target the config panel.
<i>cFormat</i>	the message, in a printf-like format
...	arguments of the format.

5.29.3.5 void GtkWidget* cairo_dock_build_generic_gui_window (const gchar * *cTitle*, int *iWidth*, int *iHeight*, CairoDockApplyConfigFunc *pAction*, gpointer *pUserData*, GFreeFunc *pFreeUserData*)

Build a generic GUI window, that contains 2 buttons apply and quit, and a statusbar.

Parameters

<i>cTitle</i>	title to set to the window.
<i>iWidth</i>	width of the window.
<i>iHeight</i>	height of the window.
<i>pAction</i>	callback to be called when the apply button is pressed, or NULL.
<i>pUserData</i>	data passed to the previous callback, or NULL.
<i>pFreeUserData</i>	callback called when the window is destroyed, to free the previous data, or NULL.

Returns

the new window.

5.29.3.6 void cairo_dock_reload_generic_gui (GtkWidget * *pWindow*)

Reload a generic window built upon a conf file, by parsing again the conf file.

Parameters

<i>pWindow</i>	the window.
----------------	-------------

5.30 cairo-dock-hiding-effect.h File Reference

5.30.1 Detailed Description

This class implements the rendering interface for hiding docks.

5.31 cairo-dock-icon-container.h File Reference

5.31.1 Detailed Description

This class implements the rendering interface for icons pointing on a sub-dock.

5.32 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 cairo_dock_remove_quick_info(plcon)`

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 *plconList)`
- `GList * cairo_dock_sort_icons_by_name (GList *plconList)`
- `Icon * cairo_dock_get_first_icon (GList *plconList)`
- `Icon * cairo_dock_get_last_icon (GList *plconList)`
- `Icon * cairo_dock_get_first_icon_of_group (GList *plconList, CairoDockIconGroup iGroup)`
- `Icon * cairo_dock_get_last_icon_of_group (GList *plconList, CairoDockIconGroup iGroup)`
- `Icon * cairo_dock_get_first_icon_of_order (GList *plconList, CairoDockIconGroup iGroup)`
- `Icon * cairo_dock_get_last_icon_of_order (GList *plconList, CairoDockIconGroup iGroup)`
- `Icon * cairo_dock_get_pointed_icon (GList *plconList)`
- `Icon * cairo_dock_get_next_icon (GList *plconList, Icon *plcon)`
- `Icon * cairo_dock_get_previous_icon (GList *plconList, Icon *plcon)`
- `Icon * cairo_dock_get_icon_with_command (GList *plconList, const gchar *cCommand)`
- `Icon * cairo_dock_get_icon_with_base_uri (GList *plconList, const gchar *cBaseURI)`
- `Icon * cairo_dock_get_icon_with_name (GList *plconList, const gchar *cName)`
- `Icon * cairo_dock_get_icon_with_subdock (GList *plconList, CairoDock *pSubDock)`
- `Icon * cairo_dock_get_icon_with_module (GList *plconList, CairoDockModule *pModule)`
- `void cairo_dock_get_icon_extent (Icon *plcon, int *iWidth, int *iHeight)`
- `void cairo_dock_get_current_icon_size (Icon *plcon, CairoContainer *pContainer, double *fSizeX, double *fSizeY)`
- `void cairo_dock_compute_icon_area (Icon *icon, CairoContainer *pContainer, GdkRectangle *pArea)`

- void `cairo_dock_update_icon_s_container_name` (`Icon *icon`, `const gchar *cNewParentDockName`)
- void `cairo_dock_set_icon_name` (`const gchar *clconName`, `Icon *plcon`, `CairoContainer *pContainer`)
- void `cairo_dock_set_icon_name_printf` (`Icon *plcon`, `CairoContainer *pContainer`, `const gchar *clconName-Format,...`) `G_GNUC_PRINTF(3)`
- void `cairo_dock_set_quick_info` (`Icon *plcon`, `CairoContainer *pContainer`, `const gchar *cQuickInfo`)
- void `cairo_dock_set_quick_info_printf` (`Icon *plcon`, `CairoContainer *pContainer`, `const gchar *cQuickInfo-Format,...`) `G_GNUC_PRINTF(3)`

5.32.1 Detailed Description

This class provides utility functions on Icons.

5.32.2 Macro Definition Documentation

5.32.2.1 `#define cairo_dock_icon_is_being_inserted(icon)`

Say whether an icon is currently being inserted.

5.32.2.2 `#define cairo_dock_icon_is_being_removed(icon)`

Say whether an icon is currently being removed.

5.32.2.3 `#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.

5.32.2.4 `#define cairo_dock_get_next_element(ic, list)`

Get the next element in a list, looping if necessary..

Parameters

<i>ic</i>	the current element.
<i>list</i>	a list.

Returns

the next element, or the first element of the list if 'ic' is the last one.

5.32.2.5 `#define cairo_dock_get_previous_element(ic, list)`

Get the previous element in a list, looping if necessary..

Parameters

<i>ic</i>	the current element.
<i>list</i>	a list.

Returns

the previous element, or the last element of the list if 'ic' is the first one.

5.32.2.6 #define cairo_dock_set_icon_static(*icon*, *_bStatic*)

Make an icon static or not. Static icons are not animated when mouse hovers them.

Parameters

<i>icon</i>	an icon.
<i>_bStatic</i>	static or not.

5.32.2.7 #define cairo_dock_set_icon_always_visible(*icon*, *_bAlwaysVisible*)

Make an icon always visible, even when the dock is hidden.

Parameters

<i>icon</i>	an icon.
<i>_bAlwaysVisible</i>	whether the icon is always visible or not.

5.32.2.8 #define cairo_dock_remove_quick_info(*pIcon*)

Clear the quick-info of an icon.

Parameters

<i>pIcon</i>	the icon.
--------------	-----------

5.32.3 Function Documentation

5.32.3.1 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

<i>icon</i>	the icon.
-------------	-----------

Returns

the type of the icon.

5.32.3.2 int cairo_dock_compare_icons_order (*Icon * icon1*, *Icon * icon2*)

Compare 2 icons with the order relation on (group order, icon order).

Parameters

<i>icon1</i>	an icon.
<i>icon2</i>	another icon.

Returns

-1 if $icon1 < icon2$, 1 if $icon1 > icon2$, 0 if $icon1 = icon2$.

5.32.3.3 int cairo_dock_compare_icons_name (Icon * icon1, Icon * icon2)

Compare 2 icons with the order relation on the name (case insensitive alphabetical order).

Parameters

<i>icon1</i>	an icon.
<i>icon2</i>	another icon.

Returns

-1 if icon1 < icon2, 1 if icon1 > icon2, 0 if icon1 = icon2.

5.32.3.4 int cairo_dock_compare_icons_extension (Icon * icon1, Icon * icon2)

*Compare 2 icons with the order relation on the extension of their URIs (case insensitive alphabetical order).

Parameters

<i>icon1</i>	an icon.
<i>icon2</i>	another icon.

Returns

-1 if icon1 < icon2, 1 if icon1 > icon2, 0 if icon1 = icon2.

5.32.3.5 GList* cairo_dock_sort_icons_by_order (GList * plconList)

Sort a list with the order relation on (group order, icon order).

Parameters

<i>plconList</i>	a list of icons.
------------------	------------------

Returns

the sorted list. Elements are the same as the initial list, only their order has changed.

5.32.3.6 GList* cairo_dock_sort_icons_by_name (GList * plconList)

Sort a list with the alphabetical order on the icons' name.

Parameters

<i>plconList</i>	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.

5.32.3.7 Icon* cairo_dock_get_first_icon (GList * *plconList*)

Get the first icon of a list of icons.

Parameters

<i>plconList</i>	a list of icons.
------------------	------------------

Returns

the first icon, or NULL if the list is empty.

5.32.3.8 Icon* cairo_dock_get_last_icon (GList * *plconList*)

Get the last icon of a list of icons.

Parameters

<i>plconList</i>	a list of icons.
------------------	------------------

Returns

the last icon, or NULL if the list is empty.

5.32.3.9 Icon* cairo_dock_get_first_icon_of_group (GList * *plconList*, CairoDockIconGroup *iGroup*)

Get the first icon of a given group.

Parameters

<i>plconList</i>	a list of icons.
<i>iGroup</i>	the group of icon.

Returns

the first found icon with this group, or NULL if none matches.

5.32.3.10 Icon* cairo_dock_get_last_icon_of_group (GList * *plconList*, CairoDockIconGroup *iGroup*)

Get the last icon of a given group.

Parameters

<i>plconList</i>	a list of icons.
<i>iGroup</i>	the group of icon.

Returns

the last found icon with this group, or NULL if none matches.

5.32.3.11 Icon* cairo_dock_get_first_icon_of_order (GList * *plconList*, CairoDockIconGroup *iGroup*)

Get the first icon whose group has the same order as a given one.

Parameters

<i>plconList</i>	a list of icons.
<i>iGroup</i>	a group of icon.

Returns

the first found icon, or NULL if none matches.

5.32.3.12 `Icon* cairo_dock_get_last_icon_of_order (GList * plconList, CairoDockIconGroup iGroup)`

Get the last icon whose group has the same order as a given one.

Parameters

<i>plconList</i>	a list of icons.
<i>iGroup</i>	a group of icon.

Returns

the last found icon, or NULL if none matches.

5.32.3.13 `Icon* cairo_dock_get_pointed_icon (GList * plconList)`

Get the currently pointed icon in a list of icons.

Parameters

<i>plconList</i>	a list of icons.
------------------	------------------

Returns

the icon whose field 'bPointed' is TRUE, or NULL if none is pointed.

5.32.3.14 `Icon* cairo_dock_get_next_icon (GList * plconList, Icon * plcon)`

Get the icon next to a given one. The cost is O(n).

Parameters

<i>plconList</i>	a list of icons.
<i>plcon</i>	an icon in the list.

Returns

the icon whose left neighbor is plcon, or NULL if the list is empty or if plcon is the last icon.

5.32.3.15 `Icon* cairo_dock_get_previous_icon (GList * plconList, Icon * plcon)`

Get the icon previous to a given one. The cost is O(n).

Parameters

<i>plconList</i>	a list of icons.
<i>plcon</i>	an icon in the list.

Returns

the icon whose right neighbor is *plcon*, or NULL if the list is empty or if *plcon* is the first icon.

5.32.3.16 **Icon*** `cairo_dock_get_icon_with_command (GList * plconList, const gchar * cCommand)`

Search an icon with a given command in a list of icons.

Parameters

<i>plconList</i>	a list of icons.
<i>cCommand</i>	the command.

Returns

the first icon whose field 'cCommand' is identical to the given command, or NULL if no icon matches.

5.32.3.17 **Icon*** `cairo_dock_get_icon_with_base_uri (GList * plconList, const gchar * cBaseURI)`

Search an icon with a given URI in a list of icons.

Parameters

<i>plconList</i>	a list of icons.
<i>cBaseURI</i>	the URI.

Returns

the first icon whose field 'cURI' is identical to the given URI, or NULL if no icon matches.

5.32.3.18 **Icon*** `cairo_dock_get_icon_with_name (GList * plconList, const gchar * cName)`

Search an icon with a given name in a list of icons.

Parameters

<i>plconList</i>	a list of icons.
<i>cName</i>	the name.

Returns

the first icon whose field 'cName' is identical to the given name, or NULL if no icon matches.

5.32.3.19 **Icon*** `cairo_dock_get_icon_with_subdock (GList * plconList, CairoDock * pSubDock)`

Search the icon pointing on a given sub-dock in a list of icons.

Parameters

<i>pIconList</i>	a list of icons.
<i>pSubDock</i>	a sub-dock.

Returns

the first icon whose field 'pSubDock' is equal to the given sub-dock, or NULL if no icon matches.

5.32.3.20 `Icon* cairo_dock_get_icon_with_module (GList * pIconList, CairoDockModule * pModule)`

Search the icon of a given module in a list of icons.

Parameters

<i>pIconList</i>	a list of icons.
<i>pModule</i>	the module.

Returns

the first icon which has an instance of the given module, or NULL if no icon matches.

5.32.3.21 `void cairo_dock_get_icon_extent (Icon * pIcon, int * iWidth, int * iHeight)`

Get the dimension allocated to the surface/texture of an icon.

Parameters

<i>pIcon</i>	the icon.
<i>iWidth</i>	pointer to the width.
<i>iHeight</i>	pointer to the height.

5.32.3.22 `void cairo_dock_get_current_icon_size (Icon * pIcon, CairoContainer * 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

<i>pIcon</i>	the icon
<i>pContainer</i>	its container
<i>fSizeX</i>	pointer to the X size (horizontal)
<i>fSizeY</i>	pointer to the Y size (vertical)

5.32.3.23 `void cairo_dock_compute_icon_area (Icon * icon, CairoContainer * 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

<i>icon</i>	the icon
<i>pContainer</i>	its container
<i>pArea</i>	a rectangle filled with the zone used by the icon on its container.

5.32.3.24 `void cairo_dock_update_icon_s_container_name (Icon * icon, const gchar * cNewParentDockName)`

Update the container's name of an icon with the name of a dock. In the case of a launcher or an applet, the conf file is updated too.

Parameters

<i>icon</i>	an icon.
<i>cNewParent-DockName</i>	the name of its new dock.

5.32.3.25 `void cairo_dock_set_icon_name (const gchar * clconName, Icon * plcon, CairoContainer * pContainer)`

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

<i>clconName</i>	the new label of the icon. You can even pass <code>plcon->cName</code> .
<i>plcon</i>	the icon.
<i>pContainer</i>	the container of the icon.

5.32.3.26 `void cairo_dock_set_icon_name_printf (Icon * plcon, CairoContainer * pContainer, const gchar * clconNameFormat, ...)`

Same as above, but takes a printf-like format string.

Parameters

<i>plcon</i>	the icon.
<i>pContainer</i>	the container of the icon.
<i>clconName-Format</i>	the new label of the icon, in a 'printf' way.
<i>...</i>	data to be inserted into the string.

5.32.3.27 `void void cairo_dock_set_quick_info (Icon * plcon, CairoContainer * pContainer, 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

<i>plcon</i>	the icon.
<i>pContainer</i>	the container of the icon.
<i>cQuickInfo</i>	the text of the quick-info.

5.32.3.28 `void cairo_dock_set_quick_info_printf (Icon * plcon, CairoContainer * pContainer, const gchar * cQuickInfoFormat, ...)`

Same as above, but takes a printf-like format string.

Parameters

<i>plcon</i>	the icon.
<i>pContainer</i>	the container of the icon.

<i>cQuickInfo-Format</i>	the text of the quick-info, in a 'printf' way.
...	data to be inserted into the string.

5.33 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_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 [CairoDockIconTrueType](#)
Definition of the type of icons.
- enum [CairoDockIconGroup](#)
Available groups of icons.
- enum [CairoDockAnimationState](#)
Animation state of an icon, sorted by priority.

Functions

- [Icon * cairo_dock_new_icon](#) (void)
- void [cairo_dock_load_icon_image](#) (Icon *icon, CairoContainer *pContainer)
- void [cairo_dock_load_icon_text](#) (Icon *icon, CairoDockLabelDescription *pTextDescription)
- void [cairo_dock_load_icon_quickinfo](#) (Icon *icon, CairoDockLabelDescription *pTextDescription)
- void [cairo_dock_load_icon_buffers](#) (Icon *pIcon, CairoContainer *pContainer)

5.33.1 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.

5.33.2 Macro Definition Documentation

5.33.2.1 #define CAIRO_DOCK_IS_APPLI(*icon*)

TRUE if the icon holds a window.

Parameters

<i>icon</i>	an icon.
-------------	----------

5.33.2.2 #define CAIRO_DOCK_IS_APPLET(*icon*)

TRUE if the icon holds an instance of a module.

Parameters

<i>icon</i>	an icon.
-------------	----------

5.33.2.3 #define CAIRO_DOCK_IS_MULTI_APPLI(*icon*)

TRUE if the icon is an icon pointing on the sub-dock of a class.

Parameters

<i>icon</i>	an icon.
-------------	----------

5.33.2.4 #define CAIRO_DOCK_IS_AUTOMATIC_SEPARATOR(*icon*)

TRUE if the icon is an automatic separator.

Parameters

<i>icon</i>	an icon.
-------------	----------

5.33.2.5 #define CAIRO_DOCK_IS_USER_SEPARATOR(*icon*)

TRUE if the icon is a separator added by the user.

Parameters

<i>icon</i>	an icon.
-------------	----------

5.33.2.6 #define CAIRO_DOCK_IS_NORMAL_APPLI(*icon*)

*TRUE if the icon is an icon d'appli only.

Parameters

<i>icon</i>	an icon.
-------------	----------

5.33.2.7 #define CAIRO_DOCK_IS_DETACHABLE_APPLET(*icon*)

*TRUE if the icon is an icon d'applet detachable en desklet.

Parameters

<i>icon</i>	an icon.
-------------	----------

5.33.3 Function Documentation

5.33.3.1 Icon* cairo_dock_new_icon (void)

Create an empty icon.

Returns

the newly allocated icon object.

5.33.3.2 void cairo_dock_load_icon_image (Icon * *icon*, CairoContainer * *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

<i>icon</i>	the icon.
<i>pContainer</i>	its container.

5.33.3.3 void cairo_dock_load_icon_text (Icon * *icon*, CairoDockLabelDescription * *pTextDescription*)

Fill the label buffer (surface & texture) of a given icon, according to a text description.

Parameters

<i>icon</i>	the icon.
<i>pTextDescription</i>	desctiption of the text rendering.

5.33.3.4 void cairo_dock_load_icon_quickinfo (Icon * *icon*, CairoDockLabelDescription * *pTextDescription*)

Fill the quick-info buffer (surface & texture) of a given icon, according to a text description.

Parameters

<i>icon</i>	the icon.
<i>pTextDescription</i>	desctiption of the text rendering.

5.33.3.5 void cairo_dock_load_icon_buffers (Icon * *pIcon*, CairoContainer * *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

<i>pIcon</i>	the icon.
<i>pContainer</i>	its container.

5.34 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 [cairo_dock_free_icon](#) (Icon *icon)
- void [cairo_dock_foreach_icons](#) (CairoDockForeachIconFunc pFunction, gpointer pUserData)
- gint [cairo_dock_search_icon_size](#) (GtkIconSize iIconSize)
- gchar * [cairo_dock_search_icon_s_path](#) (const gchar *cFileName, gint iDesiredIconSize)

5.34.1 Detailed Description

This class manages the icons parameters and their associated resources.

Specialized Icons are handled by the corresponding manager.

5.34.2 Enumeration Type Documentation

5.34.2.1 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.

5.34.3 Function Documentation

5.34.3.1 void cairo_dock_free_icon (Icon * icon)

Terminate an Icon and free all its allocated resources, except its sub-dock.

Parameters

<i>icon</i>	the icon to destroy.
-------------	----------------------

5.34.3.2 void cairo_dock_foreach_icons (CairoDockForeachIconFunc pFunction, gpointer pUserData)

Execute an action on all icons.

Parameters

<i>pFunction</i>	the action.
<i>pUserData</i>	data passed to the callback.

5.34.3.3 gint cairo_dock_search_icon_size (GtkIconSize iIconSize)

Search the icon size of a GtkIconSize.

Parameters

<i>iIconSize</i>	a GtkIconSize
------------------	---------------

Returns

the maximum between the width and the height of the icon size in pixel (or 128 if there is a problem)

5.34.3.4 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 '~' character in paths.

Parameters

<i>cFileName</i>	name of the icon file.
<i>iDesiredIconSize</i>	desired icon size if we use icons from user icons theme.

Returns

the complete path of the icon, or NULL if not found.

5.35 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_strings_differ(s1, s2)`
- #define `cairo_dock_colors_rvb_differ(c1, c2)`
- #define `cairo_dock_colors_differ(c1, c2)`
- #define `cairo_dock_load_image_buffer(pImage, cImageFile, iWidth, iHeight, iLoadModifier)`

Functions

- `gchar * cairo_dock_search_image_s_path` (const `gchar *cImageFile`)
- void `cairo_dock_load_image_buffer_full` (`CairoDockImageBuffer *pImage`, const `gchar *cImageFile`, int `iWidth`, int `iHeight`, `CairoDockLoadImageModifier iLoadModifier`, double `fAlpha`)
- void `cairo_dock_load_image_buffer_from_surface` (`CairoDockImageBuffer *pImage`, `cairo_surface_t *pSurface`, 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`)

5.35.1 Detailed Description

This class defines a simple image loader interface. It also handles the main image buffers of the dock. 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. If you just want to load an image into a mere `cairo_surface`, use the functions of the surface-factory.

5.35.2 Macro Definition Documentation

5.35.2.1 #define `cairo_dock_strings_differ(s1, s2)`

Say if 2 strings differ, taking into account NULL strings.

5.35.2.2 #define `cairo_dock.colors_rvb_differ(c1, c2)`

Say if 2 RGBA colors differ.

5.35.2.3 #define `cairo_dock.colors_differ(c1, c2)`

Say if 2 RGB colors differ.

5.35.2.4 #define `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

<i>pImage</i>	an ImageBuffer.
<i>cImageFile</i>	name of a file
<i>iWidth</i>	width it should be loaded. The resulting width can be different depending on the modifier.
<i>iHeight</i>	height it should be loaded. The resulting width can be different depending on the modifier.
<i>iLoadModifier</i>	modifier

5.35.3 Function Documentation

5.35.3.1 `gchar* cairo_dock_search_image_s_path (const gchar * clmageFile)`

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

<i>clmageFile</i>	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.

5.35.3.2 `void cairo_dock_load_image_buffer_full (CairoDockImageBuffer * plmage, const gchar * clmageFile, 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

<i>plmage</i>	an ImageBuffer.
<i>clmageFile</i>	name of a file
<i>iWidth</i>	width it should be loaded.
<i>iHeight</i>	height it should be loaded.
<i>iLoadModifier</i>	modifier
<i>fAlpha</i>	transparency (1:fully opaque)

5.35.3.3 `void cairo_dock_load_image_buffer_from_surface (CairoDockImageBuffer * plmage, cairo_surface_t * pSurface, int iWidth, int iHeight)`

Load a surface into an ImageBuffer.

Parameters

<i>plmage</i>	an ImageBuffer.
<i>pSurface</i>	a cairo surface
<i>iWidth</i>	width of the surface
<i>iHeight</i>	height of the surface

5.35.3.4 `CairoDockImageBuffer* cairo_dock_create_image_buffer (const gchar * clmageFile, 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

<i>clmageFile</i>	name of a file
<i>iWidth</i>	width it should be loaded.
<i>iHeight</i>	height it should be loaded.
<i>iLoadModifier</i>	modifier

Returns

a newly allocated ImageBuffer.

5.35.3.5 void cairo_dock_unload_image_buffer (CairoDockImageBuffer * *plmage*)

Reset an ImageBuffer's resources. It can be used to load another image then.

Parameters

<i>plmage</i>	an ImageBuffer.
---------------	-----------------

5.35.3.6 void cairo_dock_free_image_buffer (CairoDockImageBuffer * *plmage*)

Reset and free an ImageBuffer.

Parameters

<i>plmage</i>	an ImageBuffer.
---------------	-----------------

5.36 cairo-dock-indicator-manager.h File Reference

5.36.1 Detailed Description

This class manages the indicators.

5.37 cairo-dock-keybinder.h File Reference

Macros

- #define [cd_keybinder_could_grab](#)(binding)

Typedefs

- typedef void(* [CDBindkeyHandler](#))(const gchar *keysting, gpointer user_data)

Definition of a callback, called when a shortcut is pressed by the user.

Functions

- CairoKeyBinding * [cd_keybinder_bind](#) (const gchar *keysting, const gchar *cDemander, const gchar *c-Description, const gchar *clconFilePath, const gchar *cConfFilePath, const gchar *cGroupName, const gchar *cKeyName, [CDBindkeyHandler](#) handler, gpointer user_data)
- void [cd_keybinder_unbind](#) (CairoKeyBinding *binding)
- gboolean [cd_keybinder_rebind](#) (CairoKeyBinding *binding, const gchar *cNewKeyString, const gchar *c-NewDescription)
- gboolean [cairo_dock_trigger_shortkey](#) (const gchar *cKeyString)

5.37.1 Detailed Description

This class contains functions to easily bind a keyboard shortcut to an action. These shortcuts are defined globally in your session, that is to say they will be effective whatever window has the focus. Shortkeys are of the form <alt>F1 or <ctrl><shift>s.

You bind an action to a shortcut with `cd_keybinder_bind`, and unbind it with `cd_keybinder_unbind`. To update a binding (on shortcut or description change, or just to re-grab it), use `cd_keybinder_rebind`.

5.37.2 Macro Definition Documentation

5.37.2.1 `#define cd_keybinder_could_grab(binding)`

Says if the shortcut of a key binding could be grabbed.

Parameters

<i>binding</i>	a key binding.
----------------	----------------

Returns

TRUE iff the shortcut has been successfully grabbed by the key binding.

5.37.3 Function Documentation

5.37.3.1 `CairoKeyBinding* cd_keybinder_bind (const gchar * keystring, const gchar * cDemander, const gchar * cDescription, const gchar * clconFilePath, const gchar * cConfFilePath, const gchar * cGroupName, const gchar * cKeyName, CDBindkeyHandler handler, gpointer user_data)`

Bind a shortcut to an action. Unbind it when you don't want it anymore, or when 'user_data' is freed.

Parameters

<i>keystring</i>	a shortcut.
<i>cDemander</i>	the actor making the demand
<i>cDescription</i>	a short description of the action
<i>clconFilePath</i>	an icon that represents the demander
<i>cConfFilePath</i>	conf file where the shortcut stored
<i>cGroupName</i>	group name where it's stored in the conf file
<i>cKeyName</i>	key name where it's stored in the conf file
<i>handler</i>	function called when the shortcut is pressed by the user
<i>user_data</i>	data passed to the callback

Returns

the key binding

5.37.3.2 `void cd_keybinder_unbind (CairoKeyBinding * binding)`

Unbind a shortcut. The binding is destroyed.

Parameters

<i>binding</i>	a key binding.
----------------	----------------

5.37.3.3 `gboolean cd_keybinder_rebind (CairoKeyBinding * binding, const gchar * cNewKeyString, const gchar * cNewDescription)`

Rebind a shortcutkey to a new one. If the shortcutkey is the same, don't re-bind it.

Parameters

<i>binding</i>	a key binding.
<i>.cNewKeyString</i>	the new shortcutkey
<i>.cNew-Description</i>	the new description, or NULL to keep the current one.

Returns

TRUE on success

5.37.3.4 `gboolean cairo_dock_trigger_shortkey (const gchar * cKeyString)`

Trigger the given shortcutkey. It will be as if the user effectively pressed the shortcutkey on its keyboard. It uses the 'XTest' X extension.

Parameters

<i>cKeyString</i>	a shortcutkey.
-------------------	----------------

Returns

TRUE if success.

5.38 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_conf_file (const gchar *cConfFilePath, GType iFirstDataType,...)`

5.38.1 Detailed Description

This class provides useful functions to manipulate the conf files of Cairo-Dock, which are classic group/key pair files.

5.38.2 Function Documentation

5.38.2.1 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.

5.38.2.2 void cairo_dock_write_keys_to_file (GKeyFile * *pKeyFile*, const gchar * *cConfFilePath*)

Write a conf file on the disk.

5.38.2.3 void cairo_dock_merge_conf_files (const gchar * *cConfFilePath*, gchar * *cReplacementConfFilePath*, gchar *identifier*)

Merge the values of a conf-file into another one. Keys are filtered by an identifier on the original conf-file.

Parameters

<i>cConfFilePath</i>	an up-to-date conf-file with old values, that will be updated.
<i>cReplacementConfFilePath</i>	an old conf-file containing values we want to use
<i>identifier</i>	a character to filter the keys, or 0.

5.38.2.4 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

<i>cConfFilePath</i>	path to the conf-file to update.
<i>pKeyFile</i>	a key-file with correct values, but old comments and possibly missing or old keys. It is not modified by the function.
<i>cDefaultConfFilePath</i>	a template conf-file.
<i>bUpdateKeys</i>	whether to remove old keys (hidden and persistent) or not.

5.38.2.5 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.

5.38.2.6 gboolean cairo_dock_conf_file_needs_update (GKeyFile * *pKeyFile*, const gchar * *cVersion*)

Say if a conf file's version mismatches a given version.

5.38.2.7 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.

5.38.2.8 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.

5.38.2.9 void `cairo_dock_remove_group_key_from_conf_file` (`GKeyFile * pKeyFile`, `const gchar * cGroupName`, `const gchar * ckeyName`)

Remove a key from a conf file.

5.38.2.10 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

<code>cConfFilePath</code>	path to the conf file.
<code>iFirstDataType</code>	type of the first value.

5.39 cairo-dock-kwin-integration.h File Reference

5.39.1 Detailed Description

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

5.40 cairo-dock-launcher-factory.h File Reference

Functions

- void `cairo_dock_set_launcher_class` (`Icon *icon`, `const gchar *cStartupWMClass`)
- `CairoDockIconTrueType` `cairo_dock_load_icon_info_from_desktop_file` (`const gchar *cDesktopFileName`, `Icon *icon`, `gchar **cSubDockRendererName`)
- `G_GNUC_MALLOC` `Icon *` `cairo_dock_new_launcher_icon` (`const gchar *cDesktopFileName`, `gchar **cSubDockRendererName`)

5.40.1 Detailed Description

- This class handles the creation launcher icons, from the desktop files contained inside the 'launchers' folder. The files holding the information are common desktop files, with additionnal keys added by the dock on the launcher creation.

5.40.2 Function Documentation

5.40.2.1 void `cairo_dock_set_launcher_class` (`Icon * icon`, `const gchar * cStartupWMClass`)

Set the class of a launcher. You can safely free the paramater 'cStartupWMClass' after calling this function. This function is to be called on a launcher well defined (all other parameters should be already filled).

Parameters

<i>icon</i>	a launcher.
<i>cStartupWM-Class</i>	the class of the launcher defined in its .desktop file, or NULL. You can't expect the resulting class to be the one you provide, because this function makes a lot of guesses.

5.40.2.2 CairoDockIconTrueType `cairo_dock_load_icon_info_from_desktop_file (const gchar * cDesktopFileName, Icon * icon, gchar ** cSubDockRendererName)`

Read a desktop file and fetch all its data into an Icon.

Parameters

<i>cDesktopFileName</i>	name or path of a desktop file. If it's a simple name, it will be taken in the "launchers" folder of the current theme.
<i>icon</i>	the Icon to fill.
<i>cSubDockRendererName</i>	filled with the renderer name of the sub-dock, if the icon will hold one.

Returns

the type of the icon, guessed from the values of the desktop file.

5.40.2.3 G.GNUC_MALLOC Icon* `cairo_dock_new_launcher_icon (const gchar * cDesktopFileName, gchar ** cSubDockRendererName)`

Create an Icon from a given desktop file. The resulting icon can directly be used inside a container. Class inhibiting is handled.

Parameters

<i>cDesktopFileName</i>	name of the desktop file, present in the "launchers" folder of the current theme.
<i>cSubDockRendererName</i>	filled with the renderer name of the sub-dock, if the icon will hold one.

Returns

the newly created icon.

5.41 cairo-dock-launcher-manager.h File Reference

Functions

- Icon * `cairo_dock_create_icon_from_desktop_file` (const gchar **cDesktopFileName*)
- Icon * `cairo_dock_create_dummy_launcher` (gchar **cName*, gchar **cFileName*, gchar **cCommand*, gchar **cQuickInfo*, double *fOrder*)
- void `cairo_dock_load_launchers_from_dir` (const gchar **cDirectory*)
- void `cairo_dock_reload_launcher` (Icon **icon*)

5.41.1 Detailed Description

- This class handles the creation, load and reload of launcher icons, from the desktop files contained inside the 'launchers' folder. The files holding the information are common desktop files, with additionnal keys added by the dock on the launcher creation.

5.41.2 Function Documentation

5.41.2.1 `Icon* cairo_dock_create_icon_from_desktop_file (const gchar * cDesktopFileName)`

Create an Icon from a given desktop file, and fill its buffers. The resulting icon can directly be used inside a container. Class inhibiting is handled.

Parameters

<i>cDesktopFileName</i>	name of the desktop file, present in the "launchers" folder of the current theme.
-------------------------	---

Returns

the newly created icon.

5.41.2.2 `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

<i>cName</i>	label of the icon
<i>cFileName</i>	name of an image
<i>cCommand</i>	a command, or NULL
<i>cQuickInfo</i>	a quick-info, or NULL
<i>fOrder</i>	order of the icon in its container.

Returns

the newly created icon.

5.41.2.3 `void cairo_dock_load_launchers_from_dir (const gchar * cDirectory)`

Load a set of .desktop files that define icons, and build the corresponding tree of docks. All the icons are created and placed inside their dock, which is created if necessary.

Parameters

<i>cDirectory</i>	a folder containing some .desktop files.
-------------------	--

5.41.2.4 `void cairo_dock_reload_launcher (Icon * icon)`

Reload completely a launcher. It handles all the side-effects like modifying the class, the sub-dock's view, the container, etc.

Parameters

<i>icon</i>	the launcher Icon to reload.
-------------	------------------------------

5.42 cairo-dock-manager.h File Reference

5.42.1 Detailed Description

This class defines the Managers. Managers are the core of Cairo-Dock. A Manager is a set of parameters and an interface, and manages all the resources associated to its functions.

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.

5.43 cairo-dock-module-factory.h File Reference

Data Structures

- struct [_CairoDockVisitCard](#)
Definition of the visit card of a module. Contains everything that is statically defined for a module.
- struct [_CairoDockModuleInterface](#)
Definition of the interface of a module.
- struct [_CairoDockModuleInstance](#)
Definition of an instance of a module. A module can be instanciated several times.
- struct [_CairoDockModule](#)
Definition of an external module.

Typedefs

- typedef [gboolean](#)(* [CairoDockModulePreInit](#))(CairoDockVisitCard *pVisitCard, CairoDockModuleInterface *pInterface)
Pre-init function of a module. Fills the visit card and the interface of a module.

Enumerations

- enum [CairoDockModuleCategory](#)
Categories a module can be in.

Functions

- void [cairo_dock_deinstanciate_module](#) (CairoDockModuleInstance *pInstance)
- void [cairo_dock_reload_module_instance](#) (CairoDockModuleInstance *pInstance, gboolean bReloadAppletConf)
- void [cairo_dock_activate_module](#) (CairoDockModule *module, GError **erreur)
- void [cairo_dock_deactivate_module](#) (CairoDockModule *module)
- void [cairo_dock_reload_module](#) (CairoDockModule *module, gboolean bReloadAppletConf)

5.43.1 Detailed Description

This class defines 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 datas.

Modules can be instanciaded several times; each time they are, an instance is created. Each instance holds all a set of the 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 that has an icon is also called an applet.

5.43.2 Function Documentation

5.43.2.1 void cairo_dock_deinstanciate_module (CairoDockModuleInstance * *pInstance*)

Stop and free a module instance. If it was an applet, the icon is not destroyed (but is no more a valid applet). If it was in a desklet, the desklet is destroyed.

Parameters

<i>pInstance</i>	the instance to stop.
------------------	-----------------------

5.43.2.2 void cairo_dock_reload_module_instance (CairoDockModuleInstance * *pInstance*, gboolean *bReloadAppletConf*)

Reload an instance of a module.

Parameters

<i>pInstance</i>	the instance to reload
<i>bReloadApplet-Conf</i>	TRUE to reload the config of the instance before reloading it.

5.43.2.3 void cairo_dock_activate_module (CairoDockModule * *module*, GError ** *erreur*)

Create and initialize all the instances of a module.

Parameters

<i>module</i>	the module to activate.
<i>erreur</i>	error set if something bad happens.

5.43.2.4 void cairo_dock_deactivate_module (CairoDockModule * *module*)

Stop and destroy all the instances of a module.

Parameters

<i>module</i>	the module to deactivate
---------------	--------------------------

5.43.2.5 void cairo_dock_reload_module (CairoDockModule * *module*, gboolean *bReloadAppletConf*)

Reload all the instances of a module.

Parameters

<i>module</i>	the module to reload
---------------	----------------------

<i>bReloadApplet-Conf</i>	TRUE to reload the config of the instances before reloading them.
---------------------------	---

5.44 cairo-dock-module-manager.h File Reference

Functions

- [CairoDockModule * cairo_dock_find_module_from_name](#) (const gchar *cModuleName)
- [CairoDockModule * cairo_dock_load_module](#) (const gchar *cSoFilePath)
- void [cairo_dock_load_modules_in_directory](#) (const gchar *cModuleDirPath, GError **erreur)

5.44.1 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 datas.

Modules can be instanciated several times; each time they are, an instance is created. Each instance holds all a set of the 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 that has an icon is also called an applet.

5.44.2 Function Documentation

5.44.2.1 CairoDockModule* cairo_dock_find_module_from_name (const gchar * cModuleName)

Get the module which has a given name.

Parameters

<i>cModuleName</i>	the unique name of the module.
--------------------	--------------------------------

Returns

the module, or NULL if not found.

5.44.2.2 CairoDockModule* cairo_dock_load_module (const gchar * cSoFilePath)

Load a module into the table of modules. The module is opened and its visit card and interface are retrieved.

Parameters

<i>cSoFilePath</i>	path to the .so file.
--------------------	-----------------------

Returns

the newly allocated module.

5.44.2.3 void `cairo_dock_load_modules_in_directory` (const gchar * *cModuleDirPath*, GError ** *erreur*)

Load all the modules of a given folder. If the path is NULL, plug-ins are taken in the gldi install dir.

Parameters

<i>cModuleDirPath</i>	path to the a folder containing .so files.
<i>erreur</i>	error set if something bad happens.

5.45 cairo-dock-notifications.h File Reference

Macros

- #define `CAIRO_DOCK_RUN_FIRST`
Use this in `cairo_dock_register_notification_on_object` to be called before the dock.
- #define `CAIRO_DOCK_RUN_AFTER`
Use this in `cairo_dock_register_notification_on_object` to be called after the dock.
- #define `CAIRO_DOCK_INTERCEPT_NOTIFICATION`
Return this in your callback to prevent the other callbacks from being called after you.
- #define `CAIRO_DOCK_LET_PASS_NOTIFICATION`
Return this in your callback to let pass the notification to the other callbacks after you.
- #define `cairo_dock_notify_on_object`(pObject, iNotifType,...)

Typedefs

- typedef gboolean(* `CairoDockNotificationFunc`)(gpointer pUserData,...)
Generic prototype of a notification callback.

Functions

- void `cairo_dock_register_notification_on_object` (gpointer pObject, CairoDockNotificationType iNotifType, CairoDockNotificationFunc pFunction, gboolean bRunFirst, gpointer pUserData)
- void `cairo_dock_remove_notification_func_on_object` (gpointer pObject, CairoDockNotificationType iNotifType, CairoDockNotificationFunc pFunction, gpointer pUserData)

5.45.1 Detailed Description

This class defines the notification system. Each time an event occurs (like an icon being clicked), Cairo-Dock broadcasts the corresponding notification. Anybody that has registered to it will be called then. Common objects that hold notifications are Managers, Icons, and Containers (and their derivatives Docks, Desklets, Dialogs, Flyings).

5.45.2 Macro Definition Documentation

5.45.2.1 #define `cairo_dock_notify_on_object`(*pObject*, *iNotifType*, ...)

Broadcast a notification on a given object, and on all its managers.

Parameters

<i>pObject</i>	the object (Icon, Container, Manager, ...).
<i>iNotifType</i>	type of the notification.
...	parameters to be passed to the callbacks that have registered to this notification.

5.45.3 Function Documentation

5.45.3.1 void `cairo_dock_register_notification_on_object` (*gpointer* *pObject*, `CairoDockNotificationType` *iNotifType*, `CairoDockNotificationFunc` *pFunction*, `gboolean` *bRunFirst*, *gpointer* *pUserData*)

Register an action to be called when a given notification is broadcasted from a given object.

Parameters

<i>pObject</i>	the object (Icon, Container, Manager).
<i>iNotifType</i>	type of the notification.
<i>pFunction</i>	callback.
<i>bRunFirst</i>	CAIRO_DOCK_RUN_FIRST to be called before Cairo-Dock, CAIRO_DOCK_RUN_AFTER to be called after.
<i>pUserData</i>	data to be passed as the first parameter of the callback.

5.45.3.2 void `cairo_dock_remove_notification_func_on_object` (*gpointer* *pObject*, `CairoDockNotificationType` *iNotifType*, `CairoDockNotificationFunc` *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

<i>pObject</i>	the object (Icon, Container, Manager) for which the action has been registered.
<i>iNotifType</i>	type of the notification.
<i>pFunction</i>	callback.
<i>pUserData</i>	data that was registered with the callback.

5.46 cairo-dock-object.h File Reference

Enumerations

- enum `CairoObjectNotifications` { `NOTIFICATION_DESTROY` }
signals

5.46.1 Detailed Description

This class defines the Objects, a sort of GObject, but simpler and more optimized.

5.46.2 Enumeration Type Documentation

5.46.2.1 enum `CairoObjectNotifications`

signals

Enumerator:

NOTIFICATION_DESTROY notification called when the object is going to be destroyed. data : NULL

5.47 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_bitmap_font` (const gchar *cFontDescription, int first, int count)
- 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 gchar *cText, CairoDockGLFont *pFont)
- void `cairo_dock_draw_gl_text_at_position` (const gchar *cText, CairoDockGLFont *pFont, int x, int y)
- void `cairo_dock_draw_gl_text_in_area` (const gchar *cText, CairoDockGLFont *pFont, int iWidth, int iHeight, gboolean bCentered)
- void `cairo_dock_draw_gl_text_at_position_in_area` (const gchar *cText, CairoDockGLFont *pFont, int x, int y, int iWidth, int iHeight, gboolean bCentered)

5.47.1 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_bitmap_font` to load a subset of any font into bitmaps (bitmaps are not influenced by the transformation matrix) `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.

5.47.2 Function Documentation

- 5.47.2.1 `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

<i>cText</i>	the text
<i>cFontDescription</i>	a description of the font, for instance "Monospace Bold 12"
<i>pSourceContext</i>	a cairo context, not altered by the function.
<i>iWidth</i>	a pointer that will be filled with the width of the texture.
<i>iHeight</i>	a pointer that will be filled with the height of the texture.

Returns

a newly allocated texture.

- 5.47.2.2 `CairoDockGLFont* cairo_dock_load_bitmap_font (const gchar * cFontDescription, int first, int count)`

Load a font into bitmaps. You can load any characters of font with this function. The drawback is that each character is a bitmap, that is to say you can't zoom them.

Parameters

<i>cFontDescription</i>	a description of the font, for instance "Monospace Bold 12"
<i>first</i>	first character to load.
<i>count</i>	number of characters to load.

Returns

a newly allocated opengl font.

5.47.2.3 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

<i>cFontDescription</i>	a description of the font, for instance "Monospace Bold 12"
<i>first</i>	first character to load.
<i>count</i>	number of characters to load.

Returns

a newly allocated opengl font.

5.47.2.4 CairoDockGLFont* cairo_dock_load_textured_font_from_image (const gchar * *clmagePath*)

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

<i>clmagePath</i>	path to the image.
-------------------	--------------------

Returns

a newly allocated opengl font.

5.47.2.5 void cairo_dock_free_gl_font (CairoDockGLFont * *pFont*)

Free an opengl font.

Parameters

<i>pFont</i>	the font.
--------------	-----------

5.47.2.6 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

<i>cText</i>	the text
<i>pFont</i>	the font.

<i>iWidth</i>	a pointer that will be filled with the width of the text.
<i>iHeight</i>	a pointer that will be filled with the height of the text.

5.47.2.7 void cairo_dock_draw_gl_text (const gchar * *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

<i>cText</i>	the text
<i>pFont</i>	the font.

5.47.2.8 void cairo_dock_draw_gl_text_at_position (const gchar * *cText*, CairoDockGLFont * *pFont*, int *x*, int *y*)

Like /ref cairo_dock_draw_gl_text but at a given position.

Parameters

<i>cText</i>	the text
<i>pFont</i>	the font.
<i>x</i>	x position of the left bottom corner of the text.
<i>y</i>	y position of the left bottom corner of the text.

5.47.2.9 void cairo_dock_draw_gl_text_in_area (const gchar * *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

<i>cText</i>	the text
<i>pFont</i>	the font.
<i>iWidth</i>	iWidth of the area.
<i>iHeight</i>	iHeight of the area
<i>bCentered</i>	whether the text is centered on the current position or not.

5.47.2.10 void cairo_dock_draw_gl_text_at_position_in_area (const gchar * *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

<i>cText</i>	the text
<i>pFont</i>	the font.
<i>x</i>	x position of the left bottom corner of the text.
<i>y</i>	y position of the left bottom corner of the text.
<i>iWidth</i>	iWidth of the area.
<i>iHeight</i>	iHeight of the area
<i>bCentered</i>	whether the text is centered on the given position or not.

5.48 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 *fRadius*, double *fLineWidth*, double **fLineColor*)

5.48.1 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 either 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](#).

5.48.2 Function Documentation

5.48.2.1 [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 able to fill it with a texture, you can specify here the dimension of the path's husk.

Parameters

<i>iNbVertices</i>	maximum number of vertices the path will have
<i>x0</i>	x coordinate of the origin point
<i>y0</i>	y coordinate of the origin point
<i>iWidth</i>	width of the husk of the path.
<i>iHeight</i>	height of the husk of the path

Returns

a newly allocated path, with 1 point.

5.48.2.2 void cairo_dock_free_gl_path (CairoDockGLPath * *pPath*)

Destroy a path and free its allocated resources.

Parameters

<i>pPath</i>	the path.
--------------	-----------

5.48.2.3 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

<i>pPath</i>	the path.
<i>x0</i>	x coordinate of the origin point
<i>y0</i>	y coordinate of the origin point

5.48.2.4 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

<i>pPath</i>	the path.
<i>iWidth</i>	width of the hulk
<i>iHeight</i>	height of the hulk

5.48.2.5 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

<i>pPath</i>	the path.
<i>x</i>	x coordinate of the point
<i>y</i>	y coordinate of the point

5.48.2.6 void cairo_dock_gl_path_rel_line_to (CairoDockGLPath * *pPath*, GLfloat *dx*, GLfloat *dy*)

Add a line defined relatively to the current point.

Parameters

<i>pPath</i>	the path.
<i>dx</i>	horizontal offset
<i>dy</i>	vertical offset

5.48.2.7 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

<i>pPath</i>	the path.
<i>iNbPoints</i>	number of points used to discretize the curve
<i>x1</i>	first control point x
<i>y1</i>	first control point y
<i>x2</i>	second control point x
<i>y2</i>	second control point y
<i>x3</i>	terminal point of the curve x
<i>y3</i>	terminal point of the curve y

5.48.2.8 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

<i>pPath</i>	the path.
<i>iNbPoints</i>	number of points used to discretize the curve
<i>dx1</i>	first control point offset x
<i>dy1</i>	first control point offset y
<i>dx2</i>	second control point offset x
<i>dy2</i>	second control point offset y
<i>dx3</i>	terminal point of the curve offset x
<i>dy3</i>	terminal point of the curve offset y

5.48.2.9 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

<i>pPath</i>	the path.
<i>iNbPoints</i>	number of points used to discretize the curve
<i>x1</i>	control point x
<i>y1</i>	control point y
<i>x2</i>	terminal point of the curve x
<i>y2</i>	terminal point of the curve y

5.48.2.10 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

<i>pPath</i>	the path.
<i>iNbPoints</i>	number of points used to discretize the curve
<i>dx1</i>	control point offset x
<i>dy1</i>	control point offset y
<i>dx2</i>	terminal point of the curve offset x
<i>dy2</i>	terminal point of the curve offset y

5.48.2.11 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

<i>pPath</i>	the path.
<i>iNbPoints</i>	number of points used to discretize the arc
<i>xc</i>	x coordinate of the center
<i>yc</i>	y coordinate of the center
<i>r</i>	radius
<i>teta0</i>	initial angle
<i>cone</i>	cone of the arc (a negative value means clockwise).

5.48.2.12 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

<i>pPath</i>	the path.
<i>bClosePath</i>	whether to close the path (that is to say, join the last point with the first one) or not.

5.48.2.13 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

<i>pPath</i>	the path.
<i>iTexture</i>	the texture, or 0 to fill the path with the current color. To fill the path with a gradation, use <code>GL_COLOR_ARRAY</code> and feed it with a table of colors that matches the vertices.

5.48.2.14 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

<i>fFrameWidth</i>	width of the rectangle, without the corners.
<i>fFrameHeight</i>	height of the rectangle, including the corners.
<i>fRadius</i>	radius of the corners (can be 0).

<i>fLineWidth</i>	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.
<i>fLineColor</i>	color of the line if <i>fLineWidth</i> is non nul, or color of the background otherwise.

5.49 cairo-dock-opengl.h File Reference

Data Structures

- struct [_CairoDockGLConfig](#)

This structure summarizes the available OpenGL configuration on the system.

Functions

- gboolean [cairo_dock_initialize_opengl_backend](#) (gboolean bForceOpenGL)
- void [cairo_dock_create_icon_fbo](#) (void)
- void [cairo_dock_destroy_icon_fbo](#) (void)
- gboolean [cairo_dock_begin_draw_icon](#) (Icon *pIcon, CairoContainer *pContainer, gint iRenderingMode)
- void [cairo_dock_end_draw_icon](#) (Icon *pIcon, CairoContainer *pContainer)
- void [cairo_dock_set_perspective_view](#) (CairoContainer *pContainer)
- void [cairo_dock_set_ortho_view](#) (CairoContainer *pContainer)
- void [gldi_glx_apply_desktop_background](#) (CairoContainer *pContainer)
- void [gldi_glx_init_container](#) (CairoContainer *pContainer)

5.49.1 Detailed Description

This class manages the OpenGL backend and context.

5.49.2 Function Documentation

5.49.2.1 gboolean [cairo_dock_initialize_opengl_backend](#) (gboolean *bForceOpenGL*)

Initialize the OpenGL backend, by trying to get a suitable GLX configuration.

Parameters

<i>bForceOpenGL</i>	whether to force the use of OpenGL, or let the function decide.
---------------------	---

Returns

TRUE if OpenGL is usable.

5.49.2.2 void [cairo_dock_create_icon_fbo](#) (void)

Create an FBO to render the icons inside a dock.

5.49.2.3 void [cairo_dock_destroy_icon_fbo](#) (void)

Destroy the icons FBO.

5.49.2.4 gboolean cairo_dock_begin_draw_icon (Icon * *pIcon*, CairoContainer * *pContainer*, gint *iRenderingMode*)

Initiate an OpenGL drawing session on an icon's texture.

Parameters

<i>pIcon</i>	the icon on which to draw.
<i>pContainer</i>	its container, or NULL if the icon is not yet inside a container.
<i>iRenderingMode</i>	rendering mode. 0:normal, 1:don't clear the current texture, so that the drawing will be super-imposed 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 occurred.

5.49.2.5 void cairo_dock_end_draw_icon (Icon * *pIcon*, CairoContainer * *pContainer*)

Finish an OpenGL drawing session on an icon.

Parameters

<i>pIcon</i>	the icon on which to draw.
<i>pContainer</i>	its container, or NULL if the icon is not yet inside a container.

Returns

TRUE if you can proceed to the drawing, FALSE if an error occurred.

5.49.2.6 void cairo_dock_set_perspective_view (CairoContainer * *pContainer*)

Set a perspective view to the current GL context to fit a given ontainer. Perspective view accentuates the depth effect of the scene, but can distort it on the edges, and is difficult to manipulate because the size of objects depends on their position.

Parameters

<i>pContainer</i>	the container
-------------------	---------------

5.49.2.7 void cairo_dock_set_ortho_view (CairoContainer * *pContainer*)

Set an orthogonal view to the current GL context to fit a given ontainer. Orthogonal view is convenient to draw classic 2D, because the objects are not zoomed according to their position. The drawback is a poor depth effect.

Parameters

<i>pContainer</i>	the container
-------------------	---------------

5.49.2.8 void gldi_glx_apply_desktop_background (CairoContainer * *pContainer*)

Apply the desktop background onto a container, to emulate fake transparency.

Parameters

<i>pContainer</i>	the container
-------------------	---------------

5.49.2.9 void `gldi_glx_init_container` (`CairoContainer * pContainer`)

Set a shared default-initialized GL context on a window.

Parameters

<i>pContainer</i>	the container, not yet realized.
-------------------	----------------------------------

5.50 cairo-dock-overlay.h File Reference

Data Structures

- struct [_CairoOverlay](#)
Definition of an Icon Overlay.

Enumerations

- enum [CairoOverlayPosition](#)
Available position of an overlay on an icon.

Functions

- void [cairo_dock_add_overlay_to_icon](#) (`Icon *pIcon`, `CairoOverlay *pOverlay`, `CairoOverlayPosition iPosition`)
- gboolean [cairo_dock_add_overlay_from_image](#) (`Icon *pIcon`, `const gchar *cImageFile`, `CairoOverlayPosition iPosition`)
- void [cairo_dock_add_overlay_from_surface](#) (`Icon *pIcon`, `cairo_surface_t *pSurface`, `int iWidth`, `int iHeight`, `CairoOverlayPosition iPosition`)
- void [cairo_dock_add_overlay_from_texture](#) (`Icon *pIcon`, `GLuint iTexture`, `CairoOverlayPosition iPosition`)
- void [cairo_dock_remove_overlay_at_position](#) (`Icon *pIcon`, `CairoOverlayPosition iPosition`)
- `CairoDockImageBuffer *` [cairo_dock_get_overlay_buffer_at_position](#) (`Icon *pIcon`, `CairoOverlayPosition iPosition`)
- gboolean [cairo_dock_print_overlay_on_icon_from_image](#) (`Icon *pIcon`, `CairoContainer *pContainer`, `const gchar *cImageFile`, `CairoOverlayPosition iPosition`)
- void [cairo_dock_print_overlay_on_icon_from_surface](#) (`Icon *pIcon`, `CairoContainer *pContainer`, `cairo_surface_t *pSurface`, `int iWidth`, `int iHeight`, `CairoOverlayPosition iPosition`)

5.50.1 Detailed Description

This class defines Overlays, that are small images superimposed on the icon at a given position. You can either print the overlay directly on the icon's surface/texture, or add it (in this case it is drawn separately, and can be removed without modifying the icon's surface/texture, and will remain even if you erase the icon's surface/texture). Only one overlay can be added 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](#). To remove an overlay, use [cairo_dock_remove_overlay_at_position](#). If you need to modify an overlay directly, you can get its image buffer with [cairo_dock_get_overlay_buffer_at_position](#). 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 optimized.

5.50.2 Function Documentation

5.50.2.1 void `cairo_dock_add_overlay_to_icon` (`Icon * pIcon`, `CairoOverlay * pOverlay`, `CairoOverlayPosition iPosition`)

Add an overlay on an icon.

Parameters

<i>pIcon</i>	the icon
<i>pOverlay</i>	the overlay
<i>iPosition</i>	position where to display the overlay

5.50.2.2 gboolean `cairo_dock_add_overlay_from_image` (`Icon * pIcon`, `const gchar * cImageFile`, `CairoOverlayPosition iPosition`)

Add an overlay on an icon from an image.

Parameters

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

Returns

TRUE if the overlay has been successfully added.

5.50.2.3 void `cairo_dock_add_overlay_from_surface` (`Icon * pIcon`, `cairo_surface_t * pSurface`, `int iWidth`, `int iHeight`, `CairoOverlayPosition iPosition`)

Add an overlay on an icon from a surface.

Parameters

<i>pIcon</i>	the icon
<i>pSurface</i>	a cairo surface
<i>iWidth</i>	width of the surface
<i>iHeight</i>	height of the surface
<i>iPosition</i>	position where to display the overlay

5.50.2.4 void `cairo_dock_add_overlay_from_texture` (`Icon * pIcon`, `GLuint iTexture`, `CairoOverlayPosition iPosition`)

Add an overlay on an icon from a texture.

Parameters

<i>pIcon</i>	the icon
<i>iTexture</i>	a texture
<i>iPosition</i>	position where to display the overlay

5.50.2.5 void `cairo_dock_remove_overlay_at_position (Icon * pIcon, CairoOverlayPosition iPosition)`

Remove an overlay on an icon, given its position (there is only one overlay at a given position).

Parameters

<i>pIcon</i>	the icon
<i>iPosition</i>	position of the overlay

5.50.2.6 CairoDockImageBuffer* `cairo_dock_get_overlay_buffer_at_position (Icon * pIcon, CairoOverlayPosition iPosition)`

Get the image buffer of an overlay, given its position (there is only one overlay at a given position).

Parameters

<i>pIcon</i>	the icon
<i>iPosition</i>	position of the overlay

Returns

the image-buffer of the overlay, or NULL if there is no overlay at this position.

5.50.2.7 gboolean `cairo_dock_print_overlay_on_icon_from_image (Icon * pIcon, CairoContainer * pContainer, const gchar * clmageFile, 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

<i>pIcon</i>	the icon
<i>pContainer</i>	container of the icon
<i>clmageFile</i>	an image (if it's not a path, it is searched amongst the current theme's images)
<i>iPosition</i>	position where to display the overlay

Returns

TRUE if the overlay has been successfully printed.

5.50.2.8 void `cairo_dock_print_overlay_on_icon_from_surface (Icon * pIcon, CairoContainer * pContainer, 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

<i>pIcon</i>	the icon
<i>pContainer</i>	container of the icon
<i>pSurface</i>	a cairo surface
<i>iWidth</i>	width of the surface
<i>iHeight</i>	height of the surface
<i>iPosition</i>	position where to display the overlay

Returns

TRUE if the overlay has been successfully printed.

5.51 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 packages.

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)
- [CairoDockTask](#) * [cairo_dock_download_file_async](#) (const gchar *cURL, const gchar *cLocalPath, GFunc pCallback, gpointer data)
- gchar * [cairo_dock_get_url_data_with_post](#) (const gchar *cURL, gboolean bGetOutputHeaders, GError **erreur, const gchar *cFirstProperty,...)
- [CairoDockTask](#) * [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 *cUserPackagesDir, const gchar *cDistantPackagesDir, GHashTable *pTable)
- [CairoDockTask](#) * [cairo_dock_list_packages_async](#) (const gchar *cSharePackagesDir, const gchar *cUserPackagesDir, 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)

5.51.1 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](#).

5.51.2 Macro Definition Documentation

5.51.2.1 #define `cairo_dock_get_url_data(cURL, erreur)`

Retrieve the data of a distant URL.

Parameters

<i>cURL</i>	distant adress to get data from.
<i>erreur</i>	an error.

Returns

the data (NULL if failed). It's an array of chars, possibly containing nul chars. Free it after using.

5.51.3 Enumeration Type Documentation

5.51.3.1 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).

5.51.4 Function Documentation

5.51.4.1 gboolean `cairo_dock_download_file (const gchar * cURL, const gchar * cLocalPath)`

Download a distant file into a given location.

Parameters

<i>cURL</i>	adress of the file.
<i>cLocalPath</i>	a local path where to store the file.

Returns

TRUE on success, else FALSE..

5.51.4.2 gchar* cairo_dock_download_file_in_tmp (const gchar * cURL)

Download a distant file as a temporary file.

Parameters

<i>cURL</i>	adress of the file.
-------------	---------------------

Returns

the local path of the file on success, else NULL. Free the string after using it.

5.51.4.3 gchar* cairo_dock_download_archive (const gchar * cURL, const gchar * cExtractTo)

Download an archive and extract it into a given folder.

Parameters

<i>cURL</i>	adress of the file.
<i>cExtractTo</i>	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.

5.51.4.4 CairoDockTask* 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

<i>cURL</i>	adress of the file.
<i>cLocalPath</i>	a local path where to store the file, or NULL for a temporary file.
<i>pCallback</i>	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.
<i>data</i>	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.

5.51.4.5 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

<i>cURL</i>	the URL request
<i>bGetOutput-Headers</i>	whether to retrieve the page's header.
<i>erreur</i>	an error.
<i>cFirstProperty</i>	first property of the POST data.
...	tuples of property and data to insert in POST data; the POST data will be formed with <code>a=urlencode(b)&c=urlencode(d)&...</code> End it with NULL.

Returns

the data (NULL if failed). It's an array of chars, possibly containing nul chars. Free it after using.

5.51.4.6 CairoDockTask* 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

<i>cURL</i>	distant adress to get data from.
<i>pCallback</i>	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.
<i>data</i>	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.

5.51.4.7 void cairo_dock_free_package (CairoDockPackage * pPackage)

Destroy a package and free all its allocated memory.

Parameters

<i>pPackage</i>	the package.
-----------------	--------------

5.51.4.8 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

<i>cShare-PackagesDir</i>	path of a local folder containg packages or NULL.
<i>cUserPackages-Dir</i>	path of a user folder containg packages or NULL.
<i>cDistant-PackagesDir</i>	path of a distant folder containg packages or NULL.
<i>pTable</i>	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.

5.51.4.9 CairoDockTask* 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 different 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

<i>cSharePackagesDir</i>	path of a local folder containing packages or NULL.
<i>cUserPackagesDir</i>	path of a user folder containing packages or NULL.
<i>cDistantPackagesDir</i>	path of a distant folder containing packages or NULL.
<i>pCallback</i>	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.
<i>data</i>	data to be passed to the callback.
<i>pTable</i>	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.

5.51.4.10 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 different 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

<i>cPackageName</i>	name of the package.
<i>cSharePackagesDir</i>	path of a local folder containing packages or NULL.
<i>cUserPackagesDir</i>	path of a user folder containing packages or NULL.
<i>cDistantPackagesDir</i>	path of a distant folder containing packages or NULL.
<i>iGivenType</i>	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.

5.52 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, [CairoDockRewindParticleFunc](#) pRewindParticle)

5.52.1 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.

5.52.2 Macro Definition Documentation

5.52.2.1 #define [cairo_dock_render_particles](#)(*pParticleSystem*)

Render all the particles of a particle system.

Parameters

<i>pParticleSystem</i>	the particle system.
------------------------	----------------------

5.52.3 Function Documentation

5.52.3.1 void [cairo_dock_render_particles_full](#) ([CairoParticleSystem](#) * *pParticleSystem*, int *iDepth*)

Render all the particles of a particle system with a given depth.

Parameters

<i>pParticleSystem</i>	the particle system.
<i>iDepth</i>	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.

5.52.3.2 CairoParticleSystem* cairo_dock_create_particle_system (int *iNbParticles*, GLuint *iTexture*, double *fWidth*, double *fHeight*)

Create a particle system.

Parameters

<i>iNbParticles</i>	number of particles of the system.
<i>iTexture</i>	texture to map on each particle.
<i>fWidth</i>	width of the system.
<i>fHeight</i>	height of the system.

Returns

a newly allocated particle system.

5.52.3.3 void cairo_dock_free_particle_system (CairoParticleSystem * *pParticleSystem*)

Destroy a particle system, freeing all the resources it was using.

Parameters

<i>pParticleSystem</i>	the particle system.
------------------------	----------------------

5.52.3.4 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

<i>pParticleSystem</i>	the particle system.
<i>pRewindParticle</i>	function called on a particle when its life is over.

Returns

TRUE if some particles are still alive.

5.53 cairo-dock-surface-factory.h File Reference

Data Structures

- struct [_CairoDockLabelDescription](#)
Description of the rendering of a text.

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

- void `cairo_dock_calculate_constrained_size` (double *fImageWidth, double *fImageHeight, int iWidthConstraint, int iHeightConstraint, `CairoDockLoadImageModifier` iLoadingModifier, double *fZoomWidth, double *fZoomHeight)
- `cairo_surface_t` * `cairo_dock_create_surface_from_xicon_buffer` (gulong *pXIconBuffer, int iBufferNbElements, int iWidth, int iHeight)
- `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)
- `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 *fImageWidth, double *fImageHeight, double *fZoomX, double *fZoomY)
- `cairo_surface_t` * `cairo_dock_create_surface_from_image_simple` (const gchar *cImageFile, double fImageWidth, 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 fImageHeight, double fRotationAngle)
- `cairo_surface_t` * `cairo_dock_create_surface_from_text_full` (const gchar *cText, `CairoDockLabelDescription` *pLabelDescription, 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)

5.53.1 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 [_CairoDockLabelDescription](#), 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](#)).

5.53.2 Macro Definition Documentation

5.53.2.1 #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

<i>cImagePath</i>	path or name of an image.
<i>fImageSize</i>	the desired surface size.

Returns

the newly allocated surface.

5.53.2.2 #define cairo_dock_create_surface_from_text(*cText*, *pLabelDescription*, *iTextWidthPtr*, *iTextHeightPtr*)

Create a surface representing a text, according to a given text description.

Parameters

<i>cText</i>	the text.
<i>pLabel-Description</i>	description of the text rendering.
<i>iTextWidthPtr</i>	will be filled the width of the resulting surface.
<i>iTextHeightPtr</i>	will be filled the height of the resulting surface.

Returns

the newly allocated surface.

5.53.3 Enumeration Type Documentation

5.53.3.1 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.

5.53.4 Function Documentation

5.53.4.1 void `cairo_dock_calculate_constrained_size` (double * *fImageWidth*, double * *fImageHeight*, int *iWidthConstraint*, int *iHeightConstraint*, CairoDockLoadImageModifier *iLoadingModifier*, double * *fZoomWidth*, double * *fZoomHeight*)

Calculate the size of an image according to a constraint on width and height, and a loading modifier.

Parameters

<i>fImageWidth</i>	pointer to the width of the image. Initially contains the width of the original image, and is updated with the resulting width.
<i>fImageHeight</i>	pointer to the height of the image. Initially contains the height of the original image, and is updated with the resulting height.
<i>iWidthConstraint</i>	constraint on width (0 <=> no constraint).
<i>iHeightConstraint</i>	constraint on height (0 <=> no constraint).
<i>iLoadingModifier</i>	a mask of different loading modifiers.
<i>fZoomWidth</i>	will be filled with the zoom that has been applied on width.
<i>fZoomHeight</i>	will be filled with the zoom that has been applied on height.

5.53.4.2 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

<i>pXIconBuffer</i>	raw data of the icon.
<i>iBufferNbElements</i>	number of elements in the buffer.
<i>iWidth</i>	will be filled with the resulting width of the surface.
<i>iHeight</i>	will be filled with the resulting height of the surface.

Returns

the newly allocated surface.

5.53.4.3 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

<i>pixbuf</i>	the pixbuf.
<i>fMaxScale</i>	maximum zoom of the icon.
<i>iWidthConstraint</i>	constraint on the width, or 0 to not constraint it.
<i>iHeight-Constraint</i>	constraint on the height, or 0 to not constraint it.
<i>iLoadingModifier</i>	a mask of different loading modifiers.
<i>fImageWidth</i>	will be filled with the resulting width of the surface (hors zoom).
<i>fImageHeight</i>	will be filled with the resulting height of the surface (hors zoom).
<i>fZoomX</i>	if non NULL, will be filled with the zoom that has been applied on width.
<i>fZoomY</i>	if non NULL, will be filled with the zoom that has been applied on width.

Returns

the newly allocated surface.

5.53.4.4 `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

<i>iWidth</i>	width of the surface.
<i>iHeight</i>	height of the surface.

Returns

the newly allocated surface.

5.53.4.5 `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

<i>cImagePath</i>	complete path to the image.
<i>fMaxScale</i>	maximum zoom of the icon.
<i>iWidthConstraint</i>	constraint on the width, or 0 to not constraint it.
<i>iHeight-Constraint</i>	constraint on the height, or 0 to not constraint it.
<i>iLoadingModifier</i>	a mask of different loading modifiers.
<i>fImageWidth</i>	will be filled with the resulting width of the surface (hors zoom).
<i>fImageHeight</i>	will be filled with the resulting height of the surface (hors zoom).
<i>fZoomX</i>	if non NULL, will be filled with the zoom that has been applied on width.
<i>fZoomY</i>	if non NULL, will be filled with the zoom that has been applied on width.

Returns

the newly allocated surface.

5.53.4.6 `cairo_surface_t* cairo_dock_create_surface_from_image_simple (const gchar * clmageFile, double flmageWidth, double flmageHeight)`

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

<i>clmageFile</i>	path or name of an image.
<i>flmageWidth</i>	the desired surface width.
<i>flmageHeight</i>	the desired surface height.

Returns

the newly allocated surface.

5.53.4.7 `cairo_surface_t* cairo_dock_create_surface_from_icon (const gchar * clmagePath, double flmageWidth, double flmageHeight)`

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

<i>clmagePath</i>	path or name of an image.
<i>flmageWidth</i>	the desired surface width.
<i>flmageHeight</i>	the desired surface height.

Returns

the newly allocated surface.

5.53.4.8 `cairo_surface_t* cairo_dock_create_surface_from_pattern (const gchar * clmageFile, double flmageWidth, double flmageHeight, 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

<i>clmageFile</i>	path or name of an image that will be repeated to fill the surface.
<i>flmageWidth</i>	the desired surface width.
<i>flmageHeight</i>	the desired surface height.
<i>fAlpha</i>	transparency of the pattern (1 means opaque).

Returns

the newly allocated surface.

5.53.4.9 `cairo_surface_t* cairo_dock_rotate_surface (cairo_surface_t * pSurface, double flmageWidth, double flmageHeight, double fRotationAngle)`

Create a surface by rotating another. Only works for 1/4 of rounds.

Parameters

<i>pSurface</i>	surface to rotate.
<i>fImageWidth</i>	the width of the surface.
<i>fImageHeight</i>	the height of the surface.
<i>fRotationAngle</i>	rotation angle to apply, in radians.

Returns

the newly allocated surface.

5.53.4.10 `cairo_surface_t* cairo_dock_create_surface_from_text_full (const gchar * cText, CairoDockLabelDescription * pLabelDescription, double fMaxScale, int iMaxWidth, int * iTextWidth, int * iTextHeight)`

Create a surface representing a text, according to a given text description.

Parameters

<i>cText</i>	the text.
<i>pLabel-Description</i>	description of the text rendering.
<i>fMaxScale</i>	maximum zoom of the text.
<i>iMaxWidth</i>	maximum authorized width for the surface; it will be zoomed in to fits this limit. 0 for no limit.
<i>iTextWidth</i>	will be filled the width of the resulting surface.
<i>iTextHeight</i>	will be filled the height of the resulting surface.

Returns

the newly allocated surface.

5.53.4.11 `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

<i>pSurface</i>	surface to duplicate.
<i>fWidth</i>	the width of the surface.
<i>fHeight</i>	the height of the surface.
<i>fDesiredWidth</i>	desired width of the copy (0 to keep the same size).
<i>fDesiredHeight</i>	desired height of the copy (0 to keep the same size).

Returns

the newly allocated surface.

5.54 cairo-dock-task.h File Reference

Data Structures

- [struct `_CairoDockTask`](#)

Definition of a periodic and asynchronous Task.

Macros

- #define [cairo_dock_new_task](#)(iPeriod, get_data, update, pSharedMemory)
- #define [cairo_dock_get_task_elapsed_time](#)(pTask)

Typedefs

- typedef void(* [CairoDockGetDataAsyncFunc](#))(gpointer pSharedMemory)
Definition of the asynchronous job, that does the heavy part.
- typedef gboolean(* [CairoDockUpdateSyncFunc](#))(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.

Enumerations

- enum [CairoDockFrequencyState](#)
Type of frequency for a periodic task. The frequency of the Task is divided by 2, 4, and 10 for each state.

Functions

- void [cairo_dock_launch_task](#) (CairoDockTask *pTask)
- void [cairo_dock_launch_task_delayed](#) (CairoDockTask *pTask, double fDelay)
- CairoDockTask * [cairo_dock_new_task_full](#) (int iPeriod, [CairoDockGetDataAsyncFunc](#) get_data, [CairoDockUpdateSyncFunc](#) update, GFreeFunc free_data, gpointer pSharedMemory)
- void [cairo_dock_stop_task](#) (CairoDockTask *pTask)
- void [cairo_dock_discard_task](#) (CairoDockTask *pTask)
- void [cairo_dock_free_task](#) (CairoDockTask *pTask)
- gboolean [cairo_dock_task_is_active](#) (CairoDockTask *pTask)
- gboolean [cairo_dock_task_is_running](#) (CairoDockTask *pTask)
- void [cairo_dock_change_task_frequency](#) (CairoDockTask *pTask, int iNewPeriod)
- void [cairo_dock_relaunch_task_immediately](#) (CairoDockTask *pTask, int iNewPeriod)
- void [cairo_dock_downgrade_task_frequency](#) (CairoDockTask *pTask)
- void [cairo_dock_set_normal_task_frequency](#) (CairoDockTask *pTask)

5.54.1 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 [cairo_dock_new_task](#), launch it with [cairo_dock_launch_task](#), and destroy it with [cairo_dock_free_task](#).

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.

5.54.2 Macro Definition Documentation**5.54.2.1 #define cairo_dock_new_task(*iPeriod*, *get_data*, *update*, *pSharedMemory*)**

Create a periodic Task.

Parameters

<i>iPeriod</i>	time between 2 iterations, possibly nul for a Task to be executed once only.
<i>get_data</i>	asynchronous function, which carries out the heavy job parallel to the dock; stores the results in the shared memory.
<i>update</i>	synchronous function, which carries out the update of the dock from the result of the previous function. Returns TRUE to continue, FALSE to stop.
<i>pSharedMemory</i>	structure passed as a parameter of the <i>get_data</i> and <i>update</i> functions. Must not be accessed outside of these functions !

Returns

the newly allocated Task, ready to be launched with [cairo_dock_launch_task](#). Free it with [cairo_dock_free_task](#).

5.54.2.2 #define cairo_dock_get_task_elapsed_time(*pTask*)

Get the time elapsed since the last time the Task has run.

Parameters

<i>pTask</i>	the periodic Task.
--------------	--------------------

5.54.3 Function Documentation**5.54.3.1 void cairo_dock_launch_task (CairoDockTask * *pTask*)**

Launch a periodic Task, beforehand prepared with [cairo_dock_new_task](#). The first iteration is executed immediately. The frequency returns to its normal state.

Parameters

<i>pTask</i>	the periodic Task.
--------------	--------------------

5.54.3.2 void `cairo_dock_launch_task_delayed` (`CairoDockTask * pTask`, double `fDelay`)

Same as above but after a delay.

Parameters

<i>pTask</i>	the periodic Task.
<i>fDelay</i>	delay in ms.

5.54.3.3 `CairoDockTask*` `cairo_dock_new_task_full` (int `iPeriod`, `CairoDockGetDataAsyncFunc` `get_data`, `CairoDockUpdateSyncFunc` `update`, `GFreeFunc` `free_data`, gpointer `pSharedMemory`)

Create a periodic Task.

Parameters

<i>iPeriod</i>	time between 2 iterations, possibly nul for a Task to be executed once only.
<i>get_data</i>	asynchronous function, which carries out the heavy job parallel to the dock; stores the results in the shared memory.
<i>update</i>	synchronous function, which carries out the update of the dock from the result of the previous function. Returns TRUE to continue, FALSE to stop.
<i>free_data</i>	function called when the Task is destroyed, to free the shared memory (optionnal).
<i>pSharedMemory</i>	structure passed as a parameter of the <code>get_data</code> and <code>update</code> functions. Must not be accessed outside of these functions !

Returns

the newly allocated Task, ready to be launched with `cairo_dock_launch_task`. Free it with `cairo_dock_free_task`.

5.54.3.4 void `cairo_dock_stop_task` (`CairoDockTask * 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 `cairo_dock_launch_task`.

Parameters

<i>pTask</i>	the periodic Task.
--------------	--------------------

5.54.3.5 void `cairo_dock_discard_task` (`CairoDockTask * pTask`)

Discard a periodic Task. The asynchronous thread will continue, and the Task will be freed when it ends. Use this function carefully, since you don't know when the free will occur (especially if you've set a `free_data` callback). The Task should be considered as destroyed after a call to this function.

Parameters

<i>pTask</i>	the periodic Task.
--------------	--------------------

5.54.3.6 void `cairo_dock_free_task` (`CairoDockTask * pTask`)

Stop and destroy a periodic Task, freeing all the allocated resources. Unlike `cairo_dock_discard_task`, the task is stopped before being freed, so this is a blocking call. If you want to destroy the task inside the update callback, don't use this function; use `cairo_dock_discard_task` instead.

Parameters

<i>pTask</i>	the periodic Task.
--------------	--------------------

5.54.3.7 gboolean cairo_dock_task_is_active (CairoDockTask * pTask)

Tell if a Task is active, that is to say is periodically called.

Parameters

<i>pTask</i>	the periodic Task.
--------------	--------------------

Returns

TRUE if the Task is active.

5.54.3.8 gboolean cairo_dock_task_is_running (CairoDockTask * pTask)

Tell if a Task is running, that is to say it is either in the thread or waiting for the update.

Parameters

<i>pTask</i>	the periodic Task.
--------------	--------------------

Returns

TRUE if the Task is running.

5.54.3.9 void cairo_dock_change_task_frequency (CairoDockTask * pTask, int iNewPeriod)

Change the frequency of a Task. The next iteration is re-scheduled according to the new period.

Parameters

<i>pTask</i>	the periodic Task.
<i>iNewPeriod</i>	the new period between 2 iterations of the Task, in s.

5.54.3.10 void cairo_dock_relaunch_task_immediately (CairoDockTask * pTask, int iNewPeriod)

Change the frequency of a Task and relaunch it immediately. The next iteration is therefore immediately executed.

Parameters

<i>pTask</i>	the periodic Task.
<i>iNewPeriod</i>	the new period between 2 iterations of the Task, in s, or -1 to let it unchanged.

5.54.3.11 void cairo_dock_downgrade_task_frequency (CairoDockTask * 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

<i>pTask</i>	the periodic Task.
--------------	--------------------

5.54.3.12 void cairo_dock_set_normal_task_frequency (CairoDockTask * pTask)

Set the frequency of the Task to its normal state. This is also done automatically when launching the Task.

Parameters

<i>pTask</i>	the periodic Task.
--------------	--------------------

5.55 cairo-dock-themes-manager.h File Reference

Functions

- gboolean [cairo_dock_export_current_theme](#) (const gchar *cNewThemeName, gboolean bSaveBehavior, gboolean bSaveLaunchers)
- gboolean [cairo_dock_package_current_theme](#) (const gchar *cThemeName)
- 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 bLoadLaunchers)
- CairoDockTask * [cairo_dock_import_theme_async](#) (const gchar *cThemeName, gboolean bLoadBehavior, gboolean bLoadLaunchers, GFunc pCallback, gpointer data)

5.55.1 Detailed Description

This class allows defines the structure of the global theme of the dock (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.

5.55.2 Function Documentation

5.55.2.1 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

<i>cNewThemeName</i>	name to export the theme to.
<i>bSaveBehavior</i>	whether to save the behavior paremeters too.
<i>bSaveLaunchers</i>	whether to save the launchers too.

Returns

TRUE if the theme could be exported successfully.

5.55.2.2 gboolean cairo_dock_package_current_theme (const gchar * *cThemeName*)

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 Home.

Parameters

<i>cThemeName</i>	name of the package.
-------------------	----------------------

Returns

TRUE if the theme could be packaged successfully.

5.55.2.3 gchar* cairo_dock_depackage_theme (const gchar * *cPackagePath*)

Extract a package into the themes folder. Does not load it.

Parameters

<i>cPackagePath</i>	path of a package. If the package is distant, it is first downloaded.
---------------------	---

Returns

the path of the theme folder, or NULL if an error occurred.

5.55.2.4 gboolean cairo_dock_delete_themes (gchar ** *cThemesList*)

Remove some exported themes from the hard-disk.

Parameters

<i>cThemesList</i>	a list of theme names, NULL-terminated.
--------------------	---

Returns

TRUE if the themes has been successfully deleted.

5.55.2.5 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

<i>cThemeName</i>	name of the theme to import.
<i>bLoadBehavior</i>	whether to import the behavior parameters too.
<i>bLoadLaunchers</i>	whether to import the launchers too.

Returns

TRUE if the theme could be imported successfully.

5.55.2.6 CairoDockTask* 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).

Parameters

<i>cThemeName</i>	name of the theme to import.
<i>bLoadBehavior</i>	whether to import the behavior parameters too.
<i>bLoadLaunchers</i>	whether to import the launchers too.
<i>pCallback</i>	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.
<i>data</i>	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.

5.56 cairo-dock-X-manager.h File Reference

Data Structures

- struct [_CairoDockWMBackend](#)
Definition of the Window Manager backend.
- struct [_CairoDockDesktopBackground](#)
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_SCREEN_GEOMETRY_ALTERED](#),
[NOTIFICATION_DESKTOP_VISIBILITY_CHANGED](#),
[NOTIFICATION_KBD_STATE_CHANGED](#),
[NOTIFICATION_WINDOW_CONFIGURED](#),
[NOTIFICATION_WINDOW_ACTIVATED](#),
[NOTIFICATION_WINDOW_PROPERTY_CHANGED](#) }
signals

Functions

- void [cairo_dock_wm_register_backend](#) (CairoDockWMBackend *pBackend)
- gboolean [cairo_dock_wm_present_class](#) (const gchar *cClass)
- gboolean [cairo_dock_wm_present_windows](#) (void)
- gboolean [cairo_dock_wm_present_desktops](#) (void)

- gboolean `cairo_dock_wm_show_widget_layer` (void)
- gboolean `cairo_dock_wm_set_on_widget_layer` (Window Xid, gboolean bOnWidgetLayer)
- void `cairo_dock_get_current_desktop_and_viewport` (int *iCurrentDesktop, int *iCurrentViewportX, int *iCurrentViewportY)

5.56.1 Detailed Description

This class manages the interactions with X. The X manager will handle signals from X and dispatch them, and manages the screen geometry.

5.56.2 Enumeration Type Documentation

5.56.2.1 enum CairoDesktopNotifications

signals

Enumerator:

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

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

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_WINDOW_CONFIGURED notification called when a window is resized or moved, or when the z-order of windows has changed. data : {Xid, XConfigureEvent or NULL}.

NOTIFICATION_WINDOW_ACTIVATED notification called when the active window has changed. data - : Window* or NULL

NOTIFICATION_WINDOW_PROPERTY_CHANGED notification called when a window's property has changed. data : {Window, Atom, int}

5.56.3 Function Documentation

5.56.3.1 void `cairo_dock_wm_register_backend` (CairoDockWMBackend * *pBackend*)

Register a Window Manager backend, overwriting any previous one.

Parameters

<i>pBackend</i>	a Window Manager backend; the function takes ownership of the pointer.
-----------------	--

5.56.3.2 gboolean `cairo_dock_wm_present_class` (const gchar * *cClass*)

Present all the windows of a given class.

Parameters

<i>cClass</i>	the class.
---------------	------------

Returns

TRUE on success

5.56.3.3 gboolean cairo_dock_wm_present_windows (void)

Present all the windows of the current desktop.

Returns

TRUE on success

5.56.3.4 gboolean cairo_dock_wm_present_desktops (void)

Present all the desktops.

Returns

TRUE on success

5.56.3.5 gboolean cairo_dock_wm_show_widget_layer (void)

Show the Widget Layer.

Returns

TRUE on success

5.56.3.6 gboolean cairo_dock_wm_set_on_widget_layer (Window *Xid*, gboolean *bOnWidgetLayer*)

Set a window to be displayed on the Widget Layer.

Parameters

<i>Xid</i>	X ID of the window.
<i>bOnWidgetLayer</i>	whether to set or unset the option.

Returns

TRUE on success

5.56.3.7 void cairo_dock_get_current_desktop_and_viewport (int * *iCurrentDesktop*, int * *iCurrentViewportX*, int * *iCurrentViewportY*)

Get the current workspace (desktop and viewport).

Parameters

<i>iCurrentDesktop</i>	will be filled with the current desktop number
<i>iCurrent-ViewportX</i>	will be filled with the current horizontal viewport number
<i>iCurrent-ViewportY</i>	will be filled with the current vertical viewport number

5.57 cairo-dock-X-utilities.h File Reference

Functions

- gboolean [cairo_dock_remove_version_from_string](#) (gchar *cString)
- Window * [cairo_dock_get_windows_list](#) (gulong *iNbWindows, gboolean bStackOrder)
gboolean cairo_dock_xwindow_is_on_current_desktop (Window Xid);

5.57.1 Detailed Description

This class provides many utilities functions to interact very specifically on X.

5.57.2 Function Documentation

5.57.2.1 gboolean [cairo_dock_remove_version_from_string](#) (gchar * *cString*)

Remove the version number from a string. Directly modifies the string.

Parameters

<i>cString</i>	a string.
----------------	-----------

Returns

TRUE if a version has been removed.

Index

- [_CairoContainer, 13](#)
- [_CairoDataRenderer, 14](#)
- [_CairoDataRendererAttribute, 15](#)
- [_CairoDataRendererInterface, 16](#)
- [_CairoDesklet, 16](#)
- [_CairoDeskletAttribute, 17](#)
- [_CairoDeskletDecoration, 17](#)
- [_CairoDeskletRenderer, 17](#)
- [_CairoDialog, 18](#)
- [_CairoDialogAttribute, 18](#)
- [_CairoDialogDecorator, 19](#)
- [_CairoDialogRenderer, 19](#)
- [_CairoDock, 19](#)
- [_CairoDockClassAppli, 22](#)
- [_CairoDockDesktopBackground, 22](#)
- [_CairoDockDesktopEnvBackend, 23](#)
- [_CairoDockGLConfig, 23](#)
- [_CairoDockGLFont, 23](#)
- [_CairoDockGLPath, 24](#)
- [_CairoDockGroupKeyWidget, 24](#)
- [_CairoDockGuiBackend, 24](#)
- [_CairoDockHidingEffect, 24](#)
- [_CairoDockImageBuffer, 25](#)
- [_CairoDockLabelDescription, 25](#)
- [_CairoDockModule, 26](#)
- [_CairoDockModuleInstance, 27](#)
- [_CairoDockModuleInterface, 27](#)
- [_CairoDockPackage, 28](#)
- [_CairoDockRenderer, 28](#)
- [_CairoDockTask, 29](#)
- [_CairoDockTransition, 30](#)
- [_CairoDockVisitCard, 31](#)
- [_CairoDockWMBBackend, 31](#)
- [_CairoEmblem, 31](#)
- [_CairoGraphAttribute, 32](#)
- [_CairoIconContainerRenderer, 32](#)
- [_CairoOverlay, 32](#)
- [_CairoParticle, 33](#)
- [_CairoParticleSystem, 34](#)
- [_Icon, 34](#)
- [_IconInterface, 35](#)
- [_cairo_dock_apply_texture](#)
 - [cairo-dock-draw-opengl.h, 122](#)
- [_cairo_dock_apply_texture_at_size](#)
 - [cairo-dock-draw-opengl.h, 121](#)
- [_cairo_dock_apply_texture_at_size_with_alpha](#)
 - [cairo-dock-draw-opengl.h, 122](#)
- [_cairo_dock_delete_texture](#)
 - [cairo-dock-draw-opengl.h, 121](#)
- [_cairo_dock_disable_texture](#)
 - [cairo-dock-draw-opengl.h, 121](#)
- [_cairo_dock_enable_texture](#)
 - [cairo-dock-draw-opengl.h, 121](#)
- [_cairo_dock_set_alpha](#)
 - [cairo-dock-draw-opengl.h, 121](#)
- [_cairo_dock_set_blend_alpha](#)
 - [cairo-dock-draw-opengl.h, 121](#)
- [_cairo_dock_set_blend_over](#)
 - [cairo-dock-draw-opengl.h, 121](#)
- [_cairo_dock_set_blend_pbuffer](#)
 - [cairo-dock-draw-opengl.h, 121](#)
- [_cairo_dock_set_blend_source](#)
 - [cairo-dock-draw-opengl.h, 121](#)
- [CAIRO_DESKLET_KEEP_ABOVE](#)
 - [cairo-dock-desklet-factory.h, 91](#)
- [CAIRO_DESKLET_KEEP_BELOW](#)
 - [cairo-dock-desklet-factory.h, 91](#)
- [CAIRO_DESKLET_NORMAL](#)
 - [cairo-dock-desklet-factory.h, 91](#)
- [CAIRO_DESKLET_ON_WIDGET_LAYER](#)
 - [cairo-dock-desklet-factory.h, 91](#)
- [CAIRO_DESKLET_RESERVE_SPACE](#)
 - [cairo-dock-desklet-factory.h, 91](#)
- [CAIRO_DOCK_ANIMATED_IMAGE](#)
 - [cairo-dock-surface-factory.h, 189](#)
- [CAIRO_DOCK_ANY_PACKAGE](#)
 - [cairo-dock-packages.h, 181](#)
- [CAIRO_DOCK_DISTANT_PACKAGE](#)
 - [cairo-dock-packages.h, 181](#)
- [CAIRO_DOCK_DONT_ZOOM_IN](#)
 - [cairo-dock-surface-factory.h, 188](#)
- [CAIRO_DOCK_FILL_SPACE](#)
 - [cairo-dock-surface-factory.h, 188](#)
- [CAIRO_DOCK_GRAPH_BAR](#)
 - [cairo-dock-graph.h, 133](#)
- [CAIRO_DOCK_GRAPH_CIRCLE](#)
 - [cairo-dock-graph.h, 133](#)
- [CAIRO_DOCK_GRAPH_CIRCLE_PLAIN](#)
 - [cairo-dock-graph.h, 133](#)
- [CAIRO_DOCK_GRAPH_LINE](#)
 - [cairo-dock-graph.h, 133](#)
- [CAIRO_DOCK_GRAPH_PLAIN](#)
 - [cairo-dock-graph.h, 133](#)
- [CAIRO_DOCK_INFO_NONE](#)
 - [cairo-dock-applet-facility.h, 66](#)
- [CAIRO_DOCK_INFO_ON_ICON](#)
 - [cairo-dock-applet-facility.h, 66](#)
- [CAIRO_DOCK_INFO_ON_LABEL](#)

- cairo-dock-applet-facility.h, 66
- CAIRO_DOCK_KEEP_RATIO
 - cairo-dock-surface-factory.h, 188
- CAIRO_DOCK_LOCAL_PACKAGE
 - cairo-dock-packages.h, 181
- CAIRO_DOCK_NEW_PACKAGE
 - cairo-dock-packages.h, 181
- CAIRO_DOCK_ORIENTATION_HFLIP
 - cairo-dock-surface-factory.h, 188
- CAIRO_DOCK_ORIENTATION_ROT_180
 - cairo-dock-surface-factory.h, 188
- CAIRO_DOCK_ORIENTATION_ROT_270
 - cairo-dock-surface-factory.h, 189
- CAIRO_DOCK_ORIENTATION_ROT_90
 - cairo-dock-surface-factory.h, 189
- CAIRO_DOCK_ORIENTATION_ROT_90_HFLIP
 - cairo-dock-surface-factory.h, 189
- CAIRO_DOCK_ORIENTATION_ROT_90_VFLIP
 - cairo-dock-surface-factory.h, 189
- CAIRO_DOCK_ORIENTATION_VFLIP
 - cairo-dock-surface-factory.h, 189
- CAIRO_DOCK_UPDATED_PACKAGE
 - cairo-dock-packages.h, 181
- CAIRO_DOCK_USER_PACKAGE
 - cairo-dock-packages.h, 181
- CAIRO_DOCK_WIDGET_ANIMATION_LIST
 - cairo-dock-gui-factory.h, 135
- CAIRO_DOCK_WIDGET_CHECK_BUTTON
 - cairo-dock-gui-factory.h, 135
- CAIRO_DOCK_WIDGET_CHECK_CONTROL_BUTTON
 - cairo-dock-gui-factory.h, 135
- CAIRO_DOCK_WIDGET_CLASS_SELECTOR
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_COLOR_SELECTOR_RGB
 - cairo-dock-gui-factory.h, 135
- CAIRO_DOCK_WIDGET_COLOR_SELECTOR_RGBA
 - cairo-dock-gui-factory.h, 135
- CAIRO_DOCK_WIDGET_DESKLET_DECORATION_LIST
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_DESKLET_DECORATION_LIST_WITH_DEFAULT
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_DIALOG_DECORATOR_LIST
 - cairo-dock-gui-factory.h, 135
- CAIRO_DOCK_WIDGET_DOCK_LIST
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_EMPTY_WIDGET
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_EXPANDER
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_FILE_SELECTOR
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_FOLDER_SELECTOR
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_FONT_SELECTOR
 - cairo-dock-gui-factory.h, 136
- cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_FRAME
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_HANDBOOK
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_HSCALE_DOUBLE
 - cairo-dock-gui-factory.h, 135
- CAIRO_DOCK_WIDGET_HSCALE_INTEGER
 - cairo-dock-gui-factory.h, 135
- CAIRO_DOCK_WIDGET_ICON_THEME_LIST
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_ICONS_LIST
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_JUMP_TO_MODULE
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_JUMP_TO_MODULE_IF_EXISTS
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_LAUNCH_COMMAND
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_LAUNCH_COMMAND_IF_CONDITION
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_LINK
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_LIST
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_LIST_WITH_ENTRY
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_NUMBERED_CONTROL_LIST
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_NUMBERED_CONTROL_LIST_SELECTIVE
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_NUMBERED_LIST
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_PASSWORD_ENTRY
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_SEPARATOR
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_SHORTKEY_SELECTOR
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_SIZE_INTEGER
 - cairo-dock-gui-factory.h, 135
- CAIRO_DOCK_WIDGET_SOUND_SELECTOR
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_SPIN_DOUBLE
 - cairo-dock-gui-factory.h, 135
- CAIRO_DOCK_WIDGET_SPIN_INTEGER
 - cairo-dock-gui-factory.h, 135
- CAIRO_DOCK_WIDGET_STRING_ENTRY
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_TEXT_LABEL
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_THEME_LIST
 - cairo-dock-gui-factory.h, 135
- CAIRO_DOCK_WIDGET_THEME_LIST_ENTRY

- cairo-dock-gui-factory.h, 135
- CAIRO_DOCK_WIDGET_THEME_SELECTOR
 - cairo-dock-gui-factory.h, 135
- CAIRO_DOCK_WIDGET_TREE_VIEW_MULTI_CHOICE
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_TREE_VIEW_SORT
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_TREE_VIEW_SORT_AND_MODIFY
 - cairo-dock-gui-factory.h, 136
- CAIRO_DOCK_WIDGET_VIEW_LIST
 - cairo-dock-gui-factory.h, 135
- CAIRO_DATA_RENDERER
 - cairo-dock-data-renderer.h, 80
- CAIRO_DESKLET
 - cairo-dock-desklet-factory.h, 91
- CAIRO_DIALOG
 - cairo-dock-dialog-factory.h, 99
- CAIRO_DOCK
 - cairo-dock-dock-factory.h, 112
- CAIRO_DOCK_IS_DOCK
 - cairo-dock-dock-factory.h, 112
- CD_APPLET_BIND_KEY
 - cairo-dock-applet-facility.h, 57
- CD_APPLET_INIT_END
 - cairo-dock-applet-canvas.h, 43
- CD_APPLET_MY_MENU
 - cairo-dock-applet-facility.h, 57
- CD_APPLET_STOP_END
 - cairo-dock-applet-canvas.h, 43
- cairo-dock-X-manager.h
 - NOTIFICATION_DESKTOP_CHANGED, 200
 - NOTIFICATION_DESKTOP_VISIBILITY_CHANGED, 200
 - NOTIFICATION_KBD_STATE_CHANGED, 200
 - NOTIFICATION_SCREEN_GEOMETRY_ALTERED, 200
 - NOTIFICATION_WINDOW_ACTIVATED, 200
 - NOTIFICATION_WINDOW_CONFIGURED, 200
 - NOTIFICATION_WINDOW_PROPERTY_CHANGED, 200
- cairo-dock-X-manager.h, 199
 - cairo_dock_get_current_desktop_and_viewport, 201
 - cairo_dock_wm_present_class, 200
 - cairo_dock_wm_present_desktops, 201
 - cairo_dock_wm_present_windows, 201
 - cairo_dock_wm_register_backend, 200
 - cairo_dock_wm_set_on_widget_layer, 201
 - cairo_dock_wm_show_widget_layer, 201
 - CairoDesktopNotifications, 200
- cairo-dock-X-utilities.h, 202
 - cairo_dock_remove_version_from_string, 202
- cairo-dock-animations.h, 37
 - cairo_dock_animation_will_be_visible, 38
 - cairo_dock_container_is_animating, 38
 - cairo_dock_get_animation_delta_t, 38
 - cairo_dock_get_slow_animation_delta_t, 38
 - cairo_dock_get_transition_count, 39
 - cairo_dock_get_transition_elapsed_time, 39
 - cairo_dock_get_transition_fraction, 39
 - cairo_dock_has_transition, 39
 - cairo_dock_launch_animation, 40
 - cairo_dock_pop_down, 40
 - cairo_dock_pop_up, 40
 - cairo_dock_remove_transition_on_icon, 41
 - cairo_dock_request_icon_animation, 40
 - cairo_dock_set_transition_on_icon, 41
 - cairo_dock_start_icon_animation, 40
 - cairo_dock_stop_icon_animation, 38
 - cairo_dock_trigger_icon_removal_from_dock, 40
- cairo-dock-applet-canvas.h, 41
- cairo-dock-applet-facility.h
 - CAIRO_DOCK_INFO_NONE, 66
 - CAIRO_DOCK_INFO_ON_ICON, 66
 - CAIRO_DOCK_INFO_ON_LABEL, 66
- cairo-dock-applet-facility.h, 47
 - CD_APPLET_MY_MENU, 57
 - cairo_dock_draw_bar_on_icon, 66
 - cairo_dock_get_human_readable_size, 67
 - cairo_dock_play_sound, 68
 - cairo_dock_set_icon_surface, 50
 - cairo_dock_set_icon_surface_full, 66
 - cairo_dock_set_icon_surface_with_reflect, 67
 - cairo_dock_set_image_on_icon, 67
 - cairo_dock_set_image_on_icon_with_default, 67
 - CairoDockInfoDisplay, 66
 - D_, 66
- cairo-dock-applications-manager.h, 68
 - cairo_dock_foreach_applis, 70
 - cairo_dock_foreach_applis_on_viewport, 70
 - cairo_dock_get_current_active_icon, 69
 - cairo_dock_get_current_active_window, 69
 - cairo_dock_get_current_applis_list, 69
 - cairo_dock_get_icon_with_Xid, 70
 - cairo_dock_search_window_covering_dock, 69
 - cairo_dock_search_window_overlapping_dock, 69
 - cairo_dock_start_applications_manager, 68
- cairo-dock-class-manager.h, 70
 - cairo_dock_register_class, 71
 - cairo_dock_set_data_from_class, 71
- cairo-dock-compiz-integration.h, 71
- cairo-dock-config.h, 71
 - cairo_dock_decrypt_string, 73
 - cairo_dock_encrypt_string, 73
 - cairo_dock_get_pango_weight_from_1_9, 72
 - cairo_dock_get_version_from_string, 72
 - cairo_dock_is_loading, 72
 - cairo_dock_load_config, 72
 - cairo_dock_load_current_theme, 73
- cairo-dock-container.h
 - NOTIFICATION_BUILD_CONTAINER_MENU, 76
 - NOTIFICATION_BUILD_ICON_MENU, 76
 - NOTIFICATION_CLICK_ICON, 76
 - NOTIFICATION_DOUBLE_CLICK_ICON, 76

- NOTIFICATION_DROP_DATA, 76
- NOTIFICATION_ENTER_ICON, 76
- NOTIFICATION_KEY_PRESSED, 76
- NOTIFICATION_MIDDLE_CLICK_ICON, 76
- NOTIFICATION_MOUSE_MOVED, 76
- NOTIFICATION_RENDER, 76
- NOTIFICATION_SCROLL_ICON, 76
- NOTIFICATION_START_DRAG_DATA, 76
- NOTIFICATION_UPDATE, 76
- NOTIFICATION_UPDATE_SLOW, 76
- cairo-dock-container.h, 73
 - cairo_dock_add_in_menu_with_stock_and_data, 78
 - cairo_dock_build_menu, 78
 - cairo_dock_create_sub_menu, 78
 - cairo_dock_finish_container, 76
 - cairo_dock_get_max_scale, 75
 - cairo_dock_init_container, 74
 - cairo_dock_init_container_no_opengl, 75
 - cairo_dock_notify_drop_data, 77
 - cairo_dock_popup_menu_on_container, 75
 - cairo_dock_popup_menu_on_icon, 78
 - cairo_dock_redraw_container, 76
 - cairo_dock_redraw_container_area, 76
 - cairo_dock_redraw_icon, 77
 - cairo_dock_search_container_from_icon, 77
 - cairo_dock_string_is_address, 77
 - CairoContainerNotifications, 76
 - gdi_container_enable_drop, 75
- cairo-dock-core.h, 79
- cairo-dock-data-renderer-manager.h, 79
 - cairo_dock_get_default_data_renderer_font, 79
- cairo-dock-data-renderer.h, 79
 - cairo_data_renderer_format_value, 84
 - cairo_data_renderer_format_value_full, 83
 - cairo_data_renderer_get_current_value, 82
 - cairo_data_renderer_get_data, 81
 - cairo_data_renderer_get_max_value, 81
 - cairo_data_renderer_get_min_value, 81
 - cairo_data_renderer_get_nb_values, 81
 - cairo_data_renderer_get_normalized_current_value, 83
 - cairo_data_renderer_get_normalized_current_value_with_latency, 83
 - cairo_data_renderer_get_normalized_previous_value, 83
 - cairo_data_renderer_get_normalized_value, 82
 - cairo_data_renderer_get_previous_value, 82
 - cairo_data_renderer_get_value, 82
 - cairo_dock_add_new_data_renderer_on_icon, 84
 - cairo_dock_get_default_data_renderer_font, 84
 - cairo_dock_get_icon_data_renderer, 80
 - cairo_dock_refresh_data_renderer, 85
 - cairo_dock_reload_data_renderer_on_icon, 85
 - cairo_dock_remove_data_renderer_on_icon, 84
 - cairo_dock_render_new_data_on_icon, 84
 - cairo_dock_resize_data_renderer_history, 85
- cairo-dock-dbus.h, 85
 - cairo_dock_create_new_session_proxy, 86
 - cairo_dock_create_new_system_proxy, 87
 - cairo_dock_dbus_call, 89
 - cairo_dock_dbus_detect_application, 87
 - cairo_dock_dbus_detect_system_application, 87
 - cairo_dock_dbus_get_boolean, 87
 - cairo_dock_dbus_get_integer, 88
 - cairo_dock_dbus_get_string, 88
 - cairo_dock_dbus_get_string_list, 88
 - cairo_dock_dbus_get_uchar, 89
 - cairo_dock_dbus_get_uinteger, 88
 - cairo_dock_dbus_is_enabled, 86
 - cairo_dock_get_session_connection, 86
 - cairo_dock_register_service_name, 86
- cairo-dock-default-view.h, 89
- cairo-dock-desklet-factory.h
 - CAIRO_DESKLET_KEEP_ABOVE, 91
 - CAIRO_DESKLET_KEEP_BELOW, 91
 - CAIRO_DESKLET_NORMAL, 91
 - CAIRO_DESKLET_ON_WIDGET_LAYER, 91
 - CAIRO_DESKLET_RESERVE_SPACE, 91
- cairo-dock-desklet-factory.h, 89
 - CAIRO_DESKLET, 91
 - cairo_dock_add_interactive_widget_to_desklet, 91
 - cairo_dock_add_interactive_widget_to_desklet_full, 92
 - cairo_dock_configure_desklet, 92
 - cairo_dock_free_desklet, 92
 - cairo_dock_hide_desklet, 93
 - cairo_dock_lock_desklet_position, 94
 - cairo_dock_new_desklet, 92
 - cairo_dock_set_desklet_accessibility, 93
 - cairo_dock_set_desklet_margin, 92
 - cairo_dock_set_desklet_sticky, 93
 - cairo_dock_show_desklet, 93
 - cairo_dock_steal_interactive_widget_from_desklet, 92
 - cairo_dock_zoom_out_desklet, 93
 - CairoDeskletVisibility, 91
- cairo-dock-desklet-manager.h
 - NOTIFICATION_CONFIGURE_DESKLET, 95
 - NOTIFICATION_ENTER_DESKLET, 95
 - NOTIFICATION_LEAVE_DESKLET, 95
 - NOTIFICATION_NEW_DESKLET, 95
- cairo-dock-desklet-manager.h, 94
 - cairo_dock_create_desklet, 95
 - cairo_dock_destroy_desklet, 95
 - cairo_dock_find_clicked_icon_in_desklet, 97
 - cairo_dock_foreach_desklet, 96
 - cairo_dock_foreach_icons_in_desklets, 96
 - cairo_dock_get_desklet_by_Xid, 96
 - cairo_dock_reload_desklets_decorations, 96
 - cairo_dock_set_all_desklets_visible, 96
 - cairo_dock_set_desklets_visibility_to_default, 96
 - CairoDeskletNotifications, 95
- cairo-dock-desktop-file-factory.h, 97
 - cairo_dock_add_desktop_file_from_type, 98
 - cairo_dock_add_desktop_file_from_uri, 97

- cairo_dock_remove_html_spaces, 97
- cairo-dock-dialog-factory.h, 98
 - CAIRO_DIALOG, 99
 - cairo_dock_free_dialog, 100
 - cairo_dock_new_dialog, 100
 - cairo_dock_steal_interactive_widget_from_dialog, 100
- cairo-dock-dialog-manager.h, 100
 - cairo_dock_ask_general_question_and_wait, 108
 - cairo_dock_ask_question_and_wait, 107
 - cairo_dock_build_dialog, 103
 - cairo_dock_dialog_reference, 102
 - cairo_dock_dialog_unreference, 102
 - cairo_dock_get_dialogless_icon_full, 107
 - cairo_dock_hide_dialog, 108
 - cairo_dock_icon_has_dialog, 107
 - cairo_dock_remove_dialog_if_any, 102
 - cairo_dock_remove_dialog_if_any_full, 102
 - cairo_dock_show_demand_and_wait, 106
 - cairo_dock_show_dialog_and_wait, 106
 - cairo_dock_show_dialog_full, 103
 - cairo_dock_show_dialog_with_entry, 105
 - cairo_dock_show_dialog_with_question, 105
 - cairo_dock_show_dialog_with_value, 105
 - cairo_dock_show_general_message, 108
 - cairo_dock_show_temporary_dialog, 104
 - cairo_dock_show_temporary_dialog_with_default_icon, 104
 - cairo_dock_show_temporary_dialog_with_icon, 104
 - cairo_dock_show_temporary_dialog_with_icon - printf, 103
 - cairo_dock_show_value_and_wait, 107
 - cairo_dock_toggle_dialog_visibility, 108
 - cairo_dock_unhide_dialog, 108
- cairo-dock-dock-facility.h, 109
 - cairo_dock_apply_wave_effect_linear, 110
 - cairo_dock_calculate_dock_icons, 109
 - cairo_dock_calculate_icons_positions_at_rest_linear, 110
 - cairo_dock_check_can_drop_linear, 111
 - cairo_dock_check_if_mouse_inside_linear, 110
 - cairo_dock_get_current_dock_width_linear, 110
 - cairo_dock_get_first_drawn_element_linear, 111
 - cairo_dock_show_subdock, 110
 - cairo_dock_update_dock_size, 109
- cairo-dock-dock-factory.h, 111
 - CAIRO_DOCK, 112
 - cairo_dock_add_new_launcher_by_type, 113
 - cairo_dock_add_new_launcher_by_uri, 113
 - cairo_dock_detach_icon_from_dock_full, 114
 - cairo_dock_insert_automatic_separators_in_dock, 114
 - cairo_dock_insert_icon_in_dock, 112
 - cairo_dock_insert_icon_in_dock_full, 114
 - cairo_dock_remove_automatic_separators, 114
 - cairo_dock_remove_icon_from_dock, 113
 - cairo_dock_remove_icons_from_dock, 114
 - cairo_dock_remove_one_icon_from_dock, 112
- cairo-dock-dock-manager.h
 - NOTIFICATION_ENTER_DOCK, 116
 - NOTIFICATION_ICON_MOVED, 116
 - NOTIFICATION_INSERT_ICON, 116
 - NOTIFICATION_LEAVE_DOCK, 116
 - NOTIFICATION_REMOVE_ICON, 116
- cairo-dock-dock-manager.h, 115
 - cairo_dock_add_root_dock_config, 119
 - cairo_dock_add_root_dock_config_for_name, 119
 - cairo_dock_create_dock, 116
 - cairo_dock_create_subdock, 116
 - cairo_dock_destroy_dock, 116
 - cairo_dock_foreach_docks, 118
 - cairo_dock_foreach_icons_in_docks, 118
 - cairo_dock_foreach_root_docks, 118
 - cairo_dock_get_readable_name_for_fock, 117
 - cairo_dock_hide_child_docks, 118
 - cairo_dock_hide_parent_dock, 118
 - cairo_dock_reload_buffers_in_all_docks, 119
 - cairo_dock_reload_one_root_dock, 119
 - cairo_dock_remove_root_dock_config, 119
 - cairo_dock_rename_dock, 118
 - cairo_dock_search_dock_from_name, 117
 - cairo_dock_search_dock_name, 117
 - cairo_dock_search_icon_pointing_on_dock, 117
 - cairo_dock_set_all_views_to_default, 119
 - cairo_dock_set_dock_visibility, 119
 - CairoDocksNotifications, 116
- cairo-dock-draw-opengl.h, 120
 - _cairo_dock_apply_texture, 122
 - _cairo_dock_apply_texture_at_size, 121
 - _cairo_dock_apply_texture_at_size_with_alpha, 122
 - _cairo_dock_delete_texture, 121
 - _cairo_dock_disable_texture, 121
 - _cairo_dock_enable_texture, 121
 - _cairo_dock_set_alpha, 121
 - _cairo_dock_set_blend_alpha, 121
 - _cairo_dock_set_blend_over, 121
 - _cairo_dock_set_blend_pbuffer, 121
 - _cairo_dock_set_blend_source, 121
 - cairo_dock_create_texture_from_image, 120
 - cairo_dock_create_texture_from_image_full, 123
 - cairo_dock_create_texture_from_raw_data, 122
 - cairo_dock_create_texture_from_surface, 122
 - cairo_dock_render_one_icon_opengl, 122
 - cairo_dock_update_icon_texture, 123
 - cairo_dock_update_label_texture, 123
 - cairo_dock_update_quick_info_texture, 123
- cairo-dock-draw.h, 124
 - cairo_dock_create_drawing_context_generic, 124
 - cairo_dock_create_drawing_context_on_area, 125
 - cairo_dock_create_drawing_context_on_container, 125
 - cairo_dock_draw_icon_cairo, 125
 - cairo_dock_draw_rounded_rectangle, 125
 - cairo_dock_draw_string, 126

- cairo_dock_erase_cairo_context, 124
- cairo_dock_render_one_icon, 126
- cairo-dock-emblem.h, 126
 - cairo_dock_draw_emblem_on_icon, 128
 - cairo_dock_free_emblem, 128
 - cairo_dock_make_emblem, 127
 - cairo_dock_make_emblem_from_surface, 127
 - cairo_dock_make_emblem_from_texture, 128
 - cairo_dock_set_emblem_position, 127
- cairo-dock-file-manager.h, 128
 - cairo_dock_fm_add_monitor_full, 130
 - cairo_dock_fm_can_eject, 130
 - cairo_dock_fm_create_file, 131
 - cairo_dock_fm_create_icon_from_URI, 132
 - cairo_dock_fm_delete_file, 131
 - cairo_dock_fm_eject_drive, 130
 - cairo_dock_fm_empty_trash, 131
 - cairo_dock_fm_get_desktop_path, 131
 - cairo_dock_fm_get_file_info, 130
 - cairo_dock_fm_get_file_properties, 130
 - cairo_dock_fm_get_trash_path, 131
 - cairo_dock_fm_is_mounted, 130
 - cairo_dock_fm_launch_uri, 130
 - cairo_dock_fm_list_apps_for_file, 131
 - cairo_dock_fm_list_directory, 129
 - cairo_dock_fm_lock_screen, 131
 - cairo_dock_fm_logout, 131
 - cairo_dock_fm_measure_directory, 130
 - cairo_dock_fm_mount_full, 130
 - cairo_dock_fm_move_file, 131
 - cairo_dock_fm_reboot, 131
 - cairo_dock_fm_register_vfs_backend, 129
 - cairo_dock_fm_remove_monitor_full, 130
 - cairo_dock_fm_rename_file, 131
 - cairo_dock_fm_setup_time, 132
 - cairo_dock_fm_show_system_monitor, 132
 - cairo_dock_fm_shutdown, 131
 - cairo_dock_fm_unmount_full, 130
 - cairo_dock_get_file_size, 132
- cairo-dock-gauge.h, 132
- cairo-dock-graph.h
 - CAIRO_DOCK_GRAPH_BAR, 133
 - CAIRO_DOCK_GRAPH_CIRCLE, 133
 - CAIRO_DOCK_GRAPH_CIRCLE_PLAIN, 133
 - CAIRO_DOCK_GRAPH_LINE, 133
 - CAIRO_DOCK_GRAPH_PLAIN, 133
- cairo-dock-graph.h, 132
 - CairoDockTypeGraph, 133
- cairo-dock-gui-factory.h
 - CAIRO_DOCK_WIDGET_ANIMATION_LIST, 135
 - CAIRO_DOCK_WIDGET_CHECK_BUTTON, 135
 - CAIRO_DOCK_WIDGET_CHECK_CONTROL_BUTTON, 135
 - CAIRO_DOCK_WIDGET_CLASS_SELECTOR, 136
 - CAIRO_DOCK_WIDGET_COLOR_SELECTOR_RGB, 135
 - CAIRO_DOCK_WIDGET_COLOR_SELECTOR_RGBA, 135
 - CAIRO_DOCK_WIDGET_DESKLET_DECORATION_LIST, 136
 - CAIRO_DOCK_WIDGET_DESKLET_DECORATION_LIST_WITH_DEFAULT, 136
 - CAIRO_DOCK_WIDGET_DIALOG_DECORATOR_LIST, 135
 - CAIRO_DOCK_WIDGET_DOCK_LIST, 136
 - CAIRO_DOCK_WIDGET_EMPTY_WIDGET, 136
 - CAIRO_DOCK_WIDGET_EXPANDER, 136
 - CAIRO_DOCK_WIDGET_FILE_SELECTOR, 136
 - CAIRO_DOCK_WIDGET_FOLDER_SELECTOR, 136
 - CAIRO_DOCK_WIDGET_FONT_SELECTOR, 136
 - CAIRO_DOCK_WIDGET_FRAME, 136
 - CAIRO_DOCK_WIDGET_HANDBOOK, 136
 - CAIRO_DOCK_WIDGET_HSCALE_DOUBLE, 135
 - CAIRO_DOCK_WIDGET_HSCALE_INTEGER, 135
 - CAIRO_DOCK_WIDGET_ICON_THEME_LIST, 136
 - CAIRO_DOCK_WIDGET_ICONS_LIST, 136
 - CAIRO_DOCK_WIDGET_JUMP_TO_MODULE, 136
 - CAIRO_DOCK_WIDGET_JUMP_TO_MODULE_IF_EXISTS, 136
 - CAIRO_DOCK_WIDGET_LAUNCH_COMMAND, 136
 - CAIRO_DOCK_WIDGET_LAUNCH_COMMAND_IF_CONDITION, 136
 - CAIRO_DOCK_WIDGET_LINK, 136
 - CAIRO_DOCK_WIDGET_LIST, 136
 - CAIRO_DOCK_WIDGET_LIST_WITH_ENTRY, 136
 - CAIRO_DOCK_WIDGET_NUMBERED_CONTROL_LIST, 136
 - CAIRO_DOCK_WIDGET_NUMBERED_CONTROL_LIST_SELECTIVE, 136
 - CAIRO_DOCK_WIDGET_NUMBERED_LIST, 136
 - CAIRO_DOCK_WIDGET_PASSWORD_ENTRY, 136
 - CAIRO_DOCK_WIDGET_SEPARATOR, 136
 - CAIRO_DOCK_WIDGET_SHORTKEY_SELECTOR, 136
 - CAIRO_DOCK_WIDGET_SIZE_INTEGER, 135
 - CAIRO_DOCK_WIDGET_SOUND_SELECTOR, 136
 - CAIRO_DOCK_WIDGET_SPIN_DOUBLE, 135
 - CAIRO_DOCK_WIDGET_SPIN_INTEGER, 135
 - CAIRO_DOCK_WIDGET_STRING_ENTRY, 136
 - CAIRO_DOCK_WIDGET_TEXT_LABEL, 136
 - CAIRO_DOCK_WIDGET_THEME_LIST, 135
 - CAIRO_DOCK_WIDGET_THEME_LIST_ENTRY, 135

- CAIRO_DOCK_WIDGET_THEME_SELECTOR, 135
- CAIRO_DOCK_WIDGET_TREE_VIEW_MULTI_CHOICE, 136
- CAIRO_DOCK_WIDGET_TREE_VIEW_SORT, 136
- CAIRO_DOCK_WIDGET_TREE_VIEW_SORT_AND_MODIFY, 136
- CAIRO_DOCK_WIDGET_VIEW_LIST, 135
- cairo-dock-gui-factory.h, 133
 - CairoDockGUIWidgetType, 135
- cairo-dock-gui-manager.h, 137
 - cairo_dock_build_generic_gui, 138
 - cairo_dock_build_generic_gui_window, 139
 - cairo_dock_get_group_key_widget_from_name, 138
 - cairo_dock_get_widget_from_name, 138
 - cairo_dock_reload_current_module_widget, 137
 - cairo_dock_reload_generic_gui, 139
 - cairo_dock_set_status_message, 139
 - cairo_dock_set_status_message_printf, 139
- cairo-dock-hiding-effect.h, 140
- cairo-dock-icon-container.h, 140
- cairo-dock-icon-facility.h, 140
 - cairo_dock_compare_icons_extension, 143
 - cairo_dock_compare_icons_name, 142
 - cairo_dock_compare_icons_order, 142
 - cairo_dock_compute_icon_area, 147
 - cairo_dock_get_current_icon_size, 147
 - cairo_dock_get_first_icon, 143
 - cairo_dock_get_first_icon_of_group, 144
 - cairo_dock_get_first_icon_of_order, 144
 - cairo_dock_get_icon_extent, 147
 - cairo_dock_get_icon_order, 141
 - cairo_dock_get_icon_type, 142
 - cairo_dock_get_icon_with_base_uri, 146
 - cairo_dock_get_icon_with_command, 146
 - cairo_dock_get_icon_with_module, 147
 - cairo_dock_get_icon_with_name, 146
 - cairo_dock_get_icon_with_subdock, 146
 - cairo_dock_get_last_icon, 144
 - cairo_dock_get_last_icon_of_group, 144
 - cairo_dock_get_last_icon_of_order, 145
 - cairo_dock_get_next_element, 141
 - cairo_dock_get_next_icon, 145
 - cairo_dock_get_pointed_icon, 145
 - cairo_dock_get_previous_element, 141
 - cairo_dock_get_previous_icon, 145
 - cairo_dock_icon_is_being_inserted, 141
 - cairo_dock_icon_is_being_removed, 141
 - cairo_dock_remove_quick_info, 142
 - cairo_dock_set_icon_always_visible, 142
 - cairo_dock_set_icon_name, 148
 - cairo_dock_set_icon_name_printf, 148
 - cairo_dock_set_icon_static, 141
 - cairo_dock_set_quick_info, 148
 - cairo_dock_set_quick_info_printf, 148
 - cairo_dock_sort_icons_by_name, 143
 - cairo_dock_sort_icons_by_order, 143
 - cairo_dock_update_icon_s_container_name, 147
- cairo-dock-icon-factory.h, 149
 - cairo_dock_load_icon_buffers, 151
 - cairo_dock_load_icon_image, 151
 - cairo_dock_load_icon_quickinfo, 151
 - cairo_dock_load_icon_text, 151
 - cairo_dock_new_icon, 151
- cairo-dock-icon-manager.h
 - NOTIFICATION_PRE_RENDER_ICON, 152
 - NOTIFICATION_RENDER_ICON, 152
 - NOTIFICATION_REQUEST_ICON_ANIMATION, 152
 - NOTIFICATION_STOP_ICON, 152
 - NOTIFICATION_UNFOLD_SUBDOCK, 152
 - NOTIFICATION_UPDATE_ICON, 152
 - NOTIFICATION_UPDATE_ICON_SLOW, 152
- cairo-dock-icon-manager.h, 152
 - cairo_dock_foreach_icons, 153
 - cairo_dock_free_icon, 153
 - cairo_dock_search_icon_s_path, 153
 - cairo_dock_search_icon_size, 153
 - CairoIconNotifications, 152
- cairo-dock-image-buffer.h, 153
 - cairo_dock_colors_differ, 154
 - cairo_dock_colors_rvb_differ, 154
 - cairo_dock_create_image_buffer, 155
 - cairo_dock_free_image_buffer, 156
 - cairo_dock_load_image_buffer, 154
 - cairo_dock_load_image_buffer_from_surface, 155
 - cairo_dock_load_image_buffer_full, 155
 - cairo_dock_search_image_s_path, 155
 - cairo_dock_strings_differ, 154
 - cairo_dock_unload_image_buffer, 156
- cairo-dock-indicator-manager.h, 156
- cairo-dock-keybinder.h, 156
 - cairo_dock_trigger_shortkey, 158
 - cd_keybinder_bind, 157
 - cd_keybinder_could_grab, 157
 - cd_keybinder_rebind, 157
 - cd_keybinder_unbind, 157
- cairo-dock-keyfile-utilities.h, 158
 - cairo_dock_add_group_key_to_conf_file, 159
 - cairo_dock_add_remove_element_to_key, 159
 - cairo_dock_conf_file_needs_update, 159
 - cairo_dock_get_conf_file_version, 159
 - cairo_dock_merge_conf_files, 159
 - cairo_dock_open_key_file, 159
 - cairo_dock_remove_group_key_from_conf_file, 160
 - cairo_dock_update_conf_file, 160
 - cairo_dock_upgrade_conf_file_full, 159
 - cairo_dock_write_keys_to_file, 159
- cairo-dock-kwin-integration.h, 160
- cairo-dock-launcher-factory.h, 160
 - cairo_dock_load_icon_info_from_desktop_file, 161
 - cairo_dock_new_launcher_icon, 161
 - cairo_dock_set_launcher_class, 160

- cairo-dock-launcher-manager.h, 161
 - cairo_dock_create_dummy_launcher, 162
 - cairo_dock_create_icon_from_desktop_file, 162
 - cairo_dock_load_launchers_from_dir, 162
 - cairo_dock_reload_launcher, 162
- cairo-dock-manager.h, 163
- cairo-dock-module-factory.h, 163
 - cairo_dock_activate_module, 164
 - cairo_dock_deactivate_module, 164
 - cairo_dock_deinstanciate_module, 164
 - cairo_dock_reload_module, 164
 - cairo_dock_reload_module_instance, 164
- cairo-dock-module-manager.h, 165
 - cairo_dock_find_module_from_name, 165
 - cairo_dock_load_module, 165
 - cairo_dock_load_modules_in_directory, 165
- cairo-dock-notifications.h, 166
 - cairo_dock_notify_on_object, 166
 - cairo_dock_register_notification_on_object, 167
 - cairo_dock_remove_notification_func_on_object, 167
- cairo-dock-object.h
 - NOTIFICATION_DESTROY, 167
- cairo-dock-object.h, 167
 - CairoObjectNotifications, 167
- cairo-dock-opengl-font.h, 167
 - cairo_dock_create_texture_from_text_simple, 168
 - cairo_dock_draw_gl_text, 170
 - cairo_dock_draw_gl_text_at_position, 170
 - cairo_dock_draw_gl_text_at_position_in_area, 170
 - cairo_dock_draw_gl_text_in_area, 170
 - cairo_dock_free_gl_font, 169
 - cairo_dock_get_gl_text_extent, 169
 - cairo_dock_load_bitmap_font, 168
 - cairo_dock_load_textured_font, 169
 - cairo_dock_load_textured_font_from_image, 169
- cairo-dock-opengl-path.h, 171
 - cairo_dock_draw_rounded_rectangle_opengl, 174
 - cairo_dock_fill_gl_path, 174
 - cairo_dock_free_gl_path, 172
 - cairo_dock_gl_path_arc, 174
 - cairo_dock_gl_path_curve_to, 172
 - cairo_dock_gl_path_line_to, 172
 - cairo_dock_gl_path_move_to, 172
 - cairo_dock_gl_path_rel_curve_to, 173
 - cairo_dock_gl_path_rel_line_to, 172
 - cairo_dock_gl_path_rel_simple_curve_to, 173
 - cairo_dock_gl_path_set_extent, 172
 - cairo_dock_gl_path_simple_curve_to, 173
 - cairo_dock_new_gl_path, 171
 - cairo_dock_stroke_gl_path, 174
- cairo-dock-opengl.h, 175
 - cairo_dock_begin_draw_icon, 175
 - cairo_dock_create_icon_fbo, 175
 - cairo_dock_destroy_icon_fbo, 175
 - cairo_dock_end_draw_icon, 176
 - cairo_dock_initialize_opengl_backend, 175
 - cairo_dock_set_ortho_view, 176
 - cairo_dock_set_perspective_view, 176
 - gldi_glx_apply_desktop_background, 176
 - gldi_glx_init_container, 177
- cairo-dock-overlay.h, 177
 - cairo_dock_add_overlay_from_image, 178
 - cairo_dock_add_overlay_from_surface, 178
 - cairo_dock_add_overlay_from_texture, 178
 - cairo_dock_add_overlay_to_icon, 178
 - cairo_dock_get_overlay_buffer_at_position, 179
 - cairo_dock_print_overlay_on_icon_from_image, 179
 - cairo_dock_print_overlay_on_icon_from_surface, 179
 - cairo_dock_remove_overlay_at_position, 178
- cairo-dock-packages.h
 - CAIRO_DOCK_ANY_PACKAGE, 181
 - CAIRO_DOCK_DISTANT_PACKAGE, 181
 - CAIRO_DOCK_LOCAL_PACKAGE, 181
 - CAIRO_DOCK_NEW_PACKAGE, 181
 - CAIRO_DOCK_UPDATED_PACKAGE, 181
 - CAIRO_DOCK_USER_PACKAGE, 181
- cairo-dock-packages.h, 180
 - cairo_dock_download_archive, 182
 - cairo_dock_download_file, 181
 - cairo_dock_download_file_async, 182
 - cairo_dock_download_file_in_tmp, 182
 - cairo_dock_free_package, 183
 - cairo_dock_get_package_path, 184
 - cairo_dock_get_url_data, 181
 - cairo_dock_get_url_data_async, 183
 - cairo_dock_get_url_data_with_post, 182
 - cairo_dock_list_packages, 183
 - cairo_dock_list_packages_async, 184
 - CairoDockPackageType, 181
- cairo-dock-particle-system.h, 184
 - cairo_dock_create_particle_system, 186
 - cairo_dock_free_particle_system, 186
 - cairo_dock_render_particles, 185
 - cairo_dock_render_particles_full, 185
 - cairo_dock_update_default_particle_system, 186
- cairo-dock-surface-factory.h
 - CAIRO_DOCK_ANIMATED_IMAGE, 189
 - CAIRO_DOCK_DONT_ZOOM_IN, 188
 - CAIRO_DOCK_FILL_SPACE, 188
 - CAIRO_DOCK_KEEP_RATIO, 188
 - CAIRO_DOCK_ORIENTATION_HFLIP, 188
 - CAIRO_DOCK_ORIENTATION_ROT_180, 188
 - CAIRO_DOCK_ORIENTATION_ROT_270, 189
 - CAIRO_DOCK_ORIENTATION_ROT_90, 189
 - CAIRO_DOCK_ORIENTATION_ROT_90_HFLIP, 189
 - CAIRO_DOCK_ORIENTATION_ROT_90_VFLIP, 189
 - CAIRO_DOCK_ORIENTATION_VFLIP, 189
- cairo-dock-surface-factory.h, 186
 - cairo_dock_calculate_constrained_size, 189
 - cairo_dock_create_blank_surface, 190
 - cairo_dock_create_surface_for_square_icon, 188

- cairo_dock_create_surface_from_icon, 191
- cairo_dock_create_surface_from_image, 190
- cairo_dock_create_surface_from_image_simple, 190
- cairo_dock_create_surface_from_pattern, 191
- cairo_dock_create_surface_from_pixbuf, 189
- cairo_dock_create_surface_from_text, 188
- cairo_dock_create_surface_from_text_full, 192
- cairo_dock_create_surface_from_xicon_buffer, 189
- cairo_dock_duplicate_surface, 192
- cairo_dock_rotate_surface, 191
- CairoDockLoadImageModifier, 188
- cairo-dock-task.h, 192
 - cairo_dock_change_task_frequency, 196
 - cairo_dock_discard_task, 195
 - cairo_dock_downgrade_task_frequency, 196
 - cairo_dock_free_task, 195
 - cairo_dock_get_task_elapsed_time, 194
 - cairo_dock_launch_task, 194
 - cairo_dock_launch_task_delayed, 194
 - cairo_dock_new_task, 194
 - cairo_dock_new_task_full, 195
 - cairo_dock_relaunch_task_immediately, 196
 - cairo_dock_set_normal_task_frequency, 197
 - cairo_dock_stop_task, 195
 - cairo_dock_task_is_active, 196
 - cairo_dock_task_is_running, 196
- cairo-dock-themes-manager.h, 197
 - cairo_dock_delete_themes, 198
 - cairo_dock_depackage_theme, 198
 - cairo_dock_export_current_theme, 197
 - cairo_dock_import_theme, 198
 - cairo_dock_import_theme_async, 199
 - cairo_dock_package_current_theme, 198
- cairo_data_renderer_format_value
 - cairo-dock-data-renderer.h, 84
- cairo_data_renderer_format_value_full
 - cairo-dock-data-renderer.h, 83
- cairo_data_renderer_get_current_value
 - cairo-dock-data-renderer.h, 82
- cairo_data_renderer_get_data
 - cairo-dock-data-renderer.h, 81
- cairo_data_renderer_get_max_value
 - cairo-dock-data-renderer.h, 81
- cairo_data_renderer_get_min_value
 - cairo-dock-data-renderer.h, 81
- cairo_data_renderer_get_nb_values
 - cairo-dock-data-renderer.h, 81
- cairo_data_renderer_get_normalized_current_value
 - cairo-dock-data-renderer.h, 83
- cairo_data_renderer_get_normalized_current_value_ -
 - with_latency
 - cairo-dock-data-renderer.h, 83
- cairo_data_renderer_get_normalized_previous_value
 - cairo-dock-data-renderer.h, 83
- cairo_data_renderer_get_normalized_value
 - cairo-dock-data-renderer.h, 82
- cairo_data_renderer_get_previous_value
 - cairo-dock-data-renderer.h, 82
- cairo_data_renderer_get_value
 - cairo-dock-data-renderer.h, 82
- cairo_dock_activate_module
 - cairo-dock-module-factory.h, 164
- cairo_dock_add_desktop_file_from_type
 - cairo-dock-desktop-file-factory.h, 98
- cairo_dock_add_desktop_file_from_uri
 - cairo-dock-desktop-file-factory.h, 97
- cairo_dock_add_group_key_to_conf_file
 - cairo-dock-keyfile-utilities.h, 159
- cairo_dock_add_in_menu_with_stock_and_data
 - cairo-dock-container.h, 78
- cairo_dock_add_interactive_widget_to_desklet
 - cairo-dock-desklet-factory.h, 91
- cairo_dock_add_interactive_widget_to_desklet_full
 - cairo-dock-desklet-factory.h, 92
- cairo_dock_add_new_data_renderer_on_icon
 - cairo-dock-data-renderer.h, 84
- cairo_dock_add_new_launcher_by_type
 - cairo-dock-dock-factory.h, 113
- cairo_dock_add_new_launcher_by_uri
 - cairo-dock-dock-factory.h, 113
- cairo_dock_add_overlay_from_image
 - cairo-dock-overlay.h, 178
- cairo_dock_add_overlay_from_surface
 - cairo-dock-overlay.h, 178
- cairo_dock_add_overlay_from_texture
 - cairo-dock-overlay.h, 178
- cairo_dock_add_overlay_to_icon
 - cairo-dock-overlay.h, 178
- cairo_dock_add_remove_element_to_key
 - cairo-dock-keyfile-utilities.h, 159
- cairo_dock_add_root_dock_config
 - cairo-dock-dock-manager.h, 119
- cairo_dock_add_root_dock_config_for_name
 - cairo-dock-dock-manager.h, 119
- cairo_dock_animation_will_be_visible
 - cairo-dock-animations.h, 38
- cairo_dock_apply_wave_effect_linear
 - cairo-dock-dock-facility.h, 110
- cairo_dock_ask_general_question_and_wait
 - cairo-dock-dialog-manager.h, 108
- cairo_dock_ask_question_and_wait
 - cairo-dock-dialog-manager.h, 107
- cairo_dock_begin_draw_icon
 - cairo-dock-opengl.h, 175
- cairo_dock_build_dialog
 - cairo-dock-dialog-manager.h, 103
- cairo_dock_build_generic_gui
 - cairo-dock-gui-manager.h, 138
- cairo_dock_build_generic_gui_window
 - cairo-dock-gui-manager.h, 139
- cairo_dock_build_menu
 - cairo-dock-container.h, 78
- cairo_dock_calculate_constrained_size
 - cairo-dock-surface-factory.h, 189

- cairo_dock_calculate_dock_icons
 - cairo-dock-dock-facility.h, 109
- cairo_dock_calculate_icons_positions_at_rest_linear
 - cairo-dock-dock-facility.h, 110
- cairo_dock_change_task_frequency
 - cairo-dock-task.h, 196
- cairo_dock_check_can_drop_linear
 - cairo-dock-dock-facility.h, 111
- cairo_dock_check_if_mouse_inside_linear
 - cairo-dock-dock-facility.h, 110
- cairo_dock_colors_differ
 - cairo-dock-image-buffer.h, 154
- cairo_dock_colors_rvb_differ
 - cairo-dock-image-buffer.h, 154
- cairo_dock_compare_icons_extension
 - cairo-dock-icon-facility.h, 143
- cairo_dock_compare_icons_name
 - cairo-dock-icon-facility.h, 142
- cairo_dock_compare_icons_order
 - cairo-dock-icon-facility.h, 142
- cairo_dock_compute_icon_area
 - cairo-dock-icon-facility.h, 147
- cairo_dock_conf_file_needs_update
 - cairo-dock-keyfile-utilities.h, 159
- cairo_dock_configure_desklet
 - cairo-dock-desklet-factory.h, 92
- cairo_dock_container_is_animating
 - cairo-dock-animations.h, 38
- cairo_dock_create_blank_surface
 - cairo-dock-surface-factory.h, 190
- cairo_dock_create_desklet
 - cairo-dock-desklet-manager.h, 95
- cairo_dock_create_dock
 - cairo-dock-dock-manager.h, 116
- cairo_dock_create_drawing_context_generic
 - cairo-dock-draw.h, 124
- cairo_dock_create_drawing_context_on_area
 - cairo-dock-draw.h, 125
- cairo_dock_create_drawing_context_on_container
 - cairo-dock-draw.h, 125
- cairo_dock_create_dummy_launcher
 - cairo-dock-launcher-manager.h, 162
- cairo_dock_create_icon_fbo
 - cairo-dock-opengl.h, 175
- cairo_dock_create_icon_from_desktop_file
 - cairo-dock-launcher-manager.h, 162
- cairo_dock_create_image_buffer
 - cairo-dock-image-buffer.h, 155
- cairo_dock_create_new_session_proxy
 - cairo-dock-dbus.h, 86
- cairo_dock_create_new_system_proxy
 - cairo-dock-dbus.h, 87
- cairo_dock_create_particle_system
 - cairo-dock-particle-system.h, 186
- cairo_dock_create_sub_menu
 - cairo-dock-container.h, 78
- cairo_dock_create_subdock
 - cairo-dock-dock-manager.h, 116
- cairo_dock_create_surface_for_square_icon
 - cairo-dock-surface-factory.h, 188
- cairo_dock_create_surface_from_icon
 - cairo-dock-surface-factory.h, 191
- cairo_dock_create_surface_from_image
 - cairo-dock-surface-factory.h, 190
- cairo_dock_create_surface_from_image_simple
 - cairo-dock-surface-factory.h, 190
- cairo_dock_create_surface_from_pattern
 - cairo-dock-surface-factory.h, 191
- cairo_dock_create_surface_from_pixbuf
 - cairo-dock-surface-factory.h, 189
- cairo_dock_create_surface_from_text
 - cairo-dock-surface-factory.h, 188
- cairo_dock_create_surface_from_text_full
 - cairo-dock-surface-factory.h, 192
- cairo_dock_create_surface_from_xicon_buffer
 - cairo-dock-surface-factory.h, 189
- cairo_dock_create_texture_from_image
 - cairo-dock-draw-opengl.h, 120
- cairo_dock_create_texture_from_image_full
 - cairo-dock-draw-opengl.h, 123
- cairo_dock_create_texture_from_raw_data
 - cairo-dock-draw-opengl.h, 122
- cairo_dock_create_texture_from_surface
 - cairo-dock-draw-opengl.h, 122
- cairo_dock_create_texture_from_text_simple
 - cairo-dock-opengl-font.h, 168
- cairo_dock_dbus_call
 - cairo-dock-dbus.h, 89
- cairo_dock_dbus_detect_application
 - cairo-dock-dbus.h, 87
- cairo_dock_dbus_detect_system_application
 - cairo-dock-dbus.h, 87
- cairo_dock_dbus_get_boolean
 - cairo-dock-dbus.h, 87
- cairo_dock_dbus_get_integer
 - cairo-dock-dbus.h, 88
- cairo_dock_dbus_get_string
 - cairo-dock-dbus.h, 88
- cairo_dock_dbus_get_string_list
 - cairo-dock-dbus.h, 88
- cairo_dock_dbus_get_uchar
 - cairo-dock-dbus.h, 89
- cairo_dock_dbus_get_uinteger
 - cairo-dock-dbus.h, 88
- cairo_dock_dbus_is_enabled
 - cairo-dock-dbus.h, 86
- cairo_dock_deactivate_module
 - cairo-dock-module-factory.h, 164
- cairo_dock_decrypt_string
 - cairo-dock-config.h, 73
- cairo_dock_deinstantiate_module
 - cairo-dock-module-factory.h, 164
- cairo_dock_delete_themes
 - cairo-dock-themes-manager.h, 198
- cairo_dock_depackage_theme
 - cairo-dock-themes-manager.h, 198

- cairo_dock_destroy_desklet
 - cairo-dock-desklet-manager.h, 95
- cairo_dock_destroy_dock
 - cairo-dock-dock-manager.h, 116
- cairo_dock_destroy_icon_fbo
 - cairo-dock-opengl.h, 175
- cairo_dock_detach_icon_from_dock_full
 - cairo-dock-dock-factory.h, 114
- cairo_dock_dialog_reference
 - cairo-dock-dialog-manager.h, 102
- cairo_dock_dialog_unreference
 - cairo-dock-dialog-manager.h, 102
- cairo_dock_discard_task
 - cairo-dock-task.h, 195
- cairo_dock_downgrade_task_frequency
 - cairo-dock-task.h, 196
- cairo_dock_download_archive
 - cairo-dock-packages.h, 182
- cairo_dock_download_file
 - cairo-dock-packages.h, 181
- cairo_dock_download_file_async
 - cairo-dock-packages.h, 182
- cairo_dock_download_file_in_tmp
 - cairo-dock-packages.h, 182
- cairo_dock_draw_bar_on_icon
 - cairo-dock-applet-facility.h, 66
- cairo_dock_draw_emblem_on_icon
 - cairo-dock-emblem.h, 128
- cairo_dock_draw_gl_text
 - cairo-dock-opengl-font.h, 170
- cairo_dock_draw_gl_text_at_position
 - cairo-dock-opengl-font.h, 170
- cairo_dock_draw_gl_text_at_position_in_area
 - cairo-dock-opengl-font.h, 170
- cairo_dock_draw_gl_text_in_area
 - cairo-dock-opengl-font.h, 170
- cairo_dock_draw_icon_cairo
 - cairo-dock-draw.h, 125
- cairo_dock_draw_rounded_rectangle
 - cairo-dock-draw.h, 125
- cairo_dock_draw_rounded_rectangle_opengl
 - cairo-dock-opengl-path.h, 174
- cairo_dock_draw_string
 - cairo-dock-draw.h, 126
- cairo_dock_duplicate_surface
 - cairo-dock-surface-factory.h, 192
- cairo_dock_encrypt_string
 - cairo-dock-config.h, 73
- cairo_dock_end_draw_icon
 - cairo-dock-opengl.h, 176
- cairo_dock_erase_cairo_context
 - cairo-dock-draw.h, 124
- cairo_dock_export_current_theme
 - cairo-dock-themes-manager.h, 197
- cairo_dock_fill_gl_path
 - cairo-dock-opengl-path.h, 174
- cairo_dock_find_clicked_icon_in_desklet
 - cairo-dock-desklet-manager.h, 97
- cairo_dock_find_module_from_name
 - cairo-dock-module-manager.h, 165
- cairo_dock_finish_container
 - cairo-dock-container.h, 76
- cairo_dock_fm_add_monitor_full
 - cairo-dock-file-manager.h, 130
- cairo_dock_fm_can_eject
 - cairo-dock-file-manager.h, 130
- cairo_dock_fm_create_file
 - cairo-dock-file-manager.h, 131
- cairo_dock_fm_create_icon_from_URI
 - cairo-dock-file-manager.h, 132
- cairo_dock_fm_delete_file
 - cairo-dock-file-manager.h, 131
- cairo_dock_fm_eject_drive
 - cairo-dock-file-manager.h, 130
- cairo_dock_fm_empty_trash
 - cairo-dock-file-manager.h, 131
- cairo_dock_fm_get_desktop_path
 - cairo-dock-file-manager.h, 131
- cairo_dock_fm_get_file_info
 - cairo-dock-file-manager.h, 130
- cairo_dock_fm_get_file_properties
 - cairo-dock-file-manager.h, 130
- cairo_dock_fm_get_trash_path
 - cairo-dock-file-manager.h, 131
- cairo_dock_fm_is_mounted
 - cairo-dock-file-manager.h, 130
- cairo_dock_fm_launch_uri
 - cairo-dock-file-manager.h, 130
- cairo_dock_fm_list_apps_for_file
 - cairo-dock-file-manager.h, 131
- cairo_dock_fm_list_directory
 - cairo-dock-file-manager.h, 129
- cairo_dock_fm_lock_screen
 - cairo-dock-file-manager.h, 131
- cairo_dock_fm_logout
 - cairo-dock-file-manager.h, 131
- cairo_dock_fm_measure_directory
 - cairo-dock-file-manager.h, 130
- cairo_dock_fm_mount_full
 - cairo-dock-file-manager.h, 130
- cairo_dock_fm_move_file
 - cairo-dock-file-manager.h, 131
- cairo_dock_fm_reboot
 - cairo-dock-file-manager.h, 131
- cairo_dock_fm_register_vfs_backend
 - cairo-dock-file-manager.h, 129
- cairo_dock_fm_remove_monitor_full
 - cairo-dock-file-manager.h, 130
- cairo_dock_fm_rename_file
 - cairo-dock-file-manager.h, 131
- cairo_dock_fm_setup_time
 - cairo-dock-file-manager.h, 132
- cairo_dock_fm_show_system_monitor
 - cairo-dock-file-manager.h, 132
- cairo_dock_fm_shutdown
 - cairo-dock-file-manager.h, 131

cairo_dock_fm_unmount_full
 cairo-dock-file-manager.h, 130
 cairo_dock_foreach_applis
 cairo-dock-applications-manager.h, 70
 cairo_dock_foreach_applis_on_viewport
 cairo-dock-applications-manager.h, 70
 cairo_dock_foreach_desklet
 cairo-dock-desklet-manager.h, 96
 cairo_dock_foreach_docks
 cairo-dock-dock-manager.h, 118
 cairo_dock_foreach_icons
 cairo-dock-icon-manager.h, 153
 cairo_dock_foreach_icons_in_desklets
 cairo-dock-desklet-manager.h, 96
 cairo_dock_foreach_icons_in_docks
 cairo-dock-dock-manager.h, 118
 cairo_dock_foreach_root_docks
 cairo-dock-dock-manager.h, 118
 cairo_dock_free_desklet
 cairo-dock-desklet-factory.h, 92
 cairo_dock_free_dialog
 cairo-dock-dialog-factory.h, 100
 cairo_dock_free_emblem
 cairo-dock-emblem.h, 128
 cairo_dock_free_gl_font
 cairo-dock-opengl-font.h, 169
 cairo_dock_free_gl_path
 cairo-dock-opengl-path.h, 172
 cairo_dock_free_icon
 cairo-dock-icon-manager.h, 153
 cairo_dock_free_image_buffer
 cairo-dock-image-buffer.h, 156
 cairo_dock_free_package
 cairo-dock-packages.h, 183
 cairo_dock_free_particle_system
 cairo-dock-particle-system.h, 186
 cairo_dock_free_task
 cairo-dock-task.h, 195
 cairo_dock_get_animation_delta_t
 cairo-dock-animations.h, 38
 cairo_dock_get_conf_file_version
 cairo-dock-keyfile-utilities.h, 159
 cairo_dock_get_current_active_icon
 cairo-dock-applications-manager.h, 69
 cairo_dock_get_current_active_window
 cairo-dock-applications-manager.h, 69
 cairo_dock_get_current_applis_list
 cairo-dock-applications-manager.h, 69
 cairo_dock_get_current_desktop_and_viewport
 cairo-dock-X-manager.h, 201
 cairo_dock_get_current_dock_width_linear
 cairo-dock-dock-facility.h, 110
 cairo_dock_get_current_icon_size
 cairo-dock-icon-facility.h, 147
 cairo_dock_get_default_data_renderer_font
 cairo-dock-data-renderer-manager.h, 79
 cairo-dock-data-renderer.h, 84
 cairo_dock_get_desklet_by_Xid
 cairo-dock-desklet-manager.h, 96
 cairo_dock_get_dialogless_icon_full
 cairo-dock-dialog-manager.h, 107
 cairo_dock_get_file_size
 cairo-dock-file-manager.h, 132
 cairo_dock_get_first_drawn_element_linear
 cairo-dock-dock-facility.h, 111
 cairo_dock_get_first_icon
 cairo-dock-icon-facility.h, 143
 cairo_dock_get_first_icon_of_group
 cairo-dock-icon-facility.h, 144
 cairo_dock_get_first_icon_of_order
 cairo-dock-icon-facility.h, 144
 cairo_dock_get_gl_text_extent
 cairo-dock-opengl-font.h, 169
 cairo_dock_get_group_key_widget_from_name
 cairo-dock-gui-manager.h, 138
 cairo_dock_get_human_readable_size
 cairo-dock-applet-facility.h, 67
 cairo_dock_get_icon_data_renderer
 cairo-dock-data-renderer.h, 80
 cairo_dock_get_icon_extent
 cairo-dock-icon-facility.h, 147
 cairo_dock_get_icon_order
 cairo-dock-icon-facility.h, 141
 cairo_dock_get_icon_type
 cairo-dock-icon-facility.h, 142
 cairo_dock_get_icon_with_Xid
 cairo-dock-applications-manager.h, 70
 cairo_dock_get_icon_with_base_uri
 cairo-dock-icon-facility.h, 146
 cairo_dock_get_icon_with_command
 cairo-dock-icon-facility.h, 146
 cairo_dock_get_icon_with_module
 cairo-dock-icon-facility.h, 147
 cairo_dock_get_icon_with_name
 cairo-dock-icon-facility.h, 146
 cairo_dock_get_icon_with_subdock
 cairo-dock-icon-facility.h, 146
 cairo_dock_get_last_icon
 cairo-dock-icon-facility.h, 144
 cairo_dock_get_last_icon_of_group
 cairo-dock-icon-facility.h, 144
 cairo_dock_get_last_icon_of_order
 cairo-dock-icon-facility.h, 145
 cairo_dock_get_max_scale
 cairo-dock-container.h, 75
 cairo_dock_get_next_element
 cairo-dock-icon-facility.h, 141
 cairo_dock_get_next_icon
 cairo-dock-icon-facility.h, 145
 cairo_dock_get_overlay_buffer_at_position
 cairo-dock-overlay.h, 179
 cairo_dock_get_package_path
 cairo-dock-packages.h, 184
 cairo_dock_get_pango_weight_from_1_9
 cairo-dock-config.h, 72
 cairo_dock_get_pointed_icon

- cairo-dock-icon-facility.h, 145
- cairo_dock_get_previous_element
 - cairo-dock-icon-facility.h, 141
- cairo_dock_get_previous_icon
 - cairo-dock-icon-facility.h, 145
- cairo_dock_get_readable_name_for_fock
 - cairo-dock-dock-manager.h, 117
- cairo_dock_get_session_connection
 - cairo-dock-dbus.h, 86
- cairo_dock_get_slow_animation_delta_t
 - cairo-dock-animations.h, 38
- cairo_dock_get_task_elapsed_time
 - cairo-dock-task.h, 194
- cairo_dock_get_transition_count
 - cairo-dock-animations.h, 39
- cairo_dock_get_transition_elapsed_time
 - cairo-dock-animations.h, 39
- cairo_dock_get_transition_fraction
 - cairo-dock-animations.h, 39
- cairo_dock_get_url_data
 - cairo-dock-packages.h, 181
- cairo_dock_get_url_data_async
 - cairo-dock-packages.h, 183
- cairo_dock_get_url_data_with_post
 - cairo-dock-packages.h, 182
- cairo_dock_get_version_from_string
 - cairo-dock-config.h, 72
- cairo_dock_get_widget_from_name
 - cairo-dock-gui-manager.h, 138
- cairo_dock_gl_path_arc
 - cairo-dock-opengl-path.h, 174
- cairo_dock_gl_path_curve_to
 - cairo-dock-opengl-path.h, 172
- cairo_dock_gl_path_line_to
 - cairo-dock-opengl-path.h, 172
- cairo_dock_gl_path_move_to
 - cairo-dock-opengl-path.h, 172
- cairo_dock_gl_path_rel_curve_to
 - cairo-dock-opengl-path.h, 173
- cairo_dock_gl_path_rel_line_to
 - cairo-dock-opengl-path.h, 172
- cairo_dock_gl_path_rel_simple_curve_to
 - cairo-dock-opengl-path.h, 173
- cairo_dock_gl_path_set_extent
 - cairo-dock-opengl-path.h, 172
- cairo_dock_gl_path_simple_curve_to
 - cairo-dock-opengl-path.h, 173
- cairo_dock_has_transition
 - cairo-dock-animations.h, 39
- cairo_dock_hide_child_docks
 - cairo-dock-dock-manager.h, 118
- cairo_dock_hide_desklet
 - cairo-dock-desklet-factory.h, 93
- cairo_dock_hide_dialog
 - cairo-dock-dialog-manager.h, 108
- cairo_dock_hide_parent_dock
 - cairo-dock-dock-manager.h, 118
- cairo_dock_icon_has_dialog
 - cairo-dock-dialog-manager.h, 107
- cairo_dock_icon_is_being_inserted
 - cairo-dock-icon-facility.h, 141
- cairo_dock_icon_is_being_removed
 - cairo-dock-icon-facility.h, 141
- cairo_dock_import_theme
 - cairo-dock-themes-manager.h, 198
- cairo_dock_import_theme_async
 - cairo-dock-themes-manager.h, 199
- cairo_dock_init_container
 - cairo-dock-container.h, 74
- cairo_dock_init_container_no_opengl
 - cairo-dock-container.h, 75
- cairo_dock_initialize_opengl_backend
 - cairo-dock-opengl.h, 175
- cairo_dock_insert_automatic_separators_in_dock
 - cairo-dock-dock-factory.h, 114
- cairo_dock_insert_icon_in_dock
 - cairo-dock-dock-factory.h, 112
- cairo_dock_insert_icon_in_dock_full
 - cairo-dock-dock-factory.h, 114
- cairo_dock_is_loading
 - cairo-dock-config.h, 72
- cairo_dock_launch_animation
 - cairo-dock-animations.h, 40
- cairo_dock_launch_task
 - cairo-dock-task.h, 194
- cairo_dock_launch_task_delayed
 - cairo-dock-task.h, 194
- cairo_dock_list_packages
 - cairo-dock-packages.h, 183
- cairo_dock_list_packages_async
 - cairo-dock-packages.h, 184
- cairo_dock_load_bitmap_font
 - cairo-dock-opengl-font.h, 168
- cairo_dock_load_config
 - cairo-dock-config.h, 72
- cairo_dock_load_current_theme
 - cairo-dock-config.h, 73
- cairo_dock_load_icon_buffers
 - cairo-dock-icon-factory.h, 151
- cairo_dock_load_icon_image
 - cairo-dock-icon-factory.h, 151
- cairo_dock_load_icon_info_from_desktop_file
 - cairo-dock-launcher-factory.h, 161
- cairo_dock_load_icon_quickinfo
 - cairo-dock-icon-factory.h, 151
- cairo_dock_load_icon_text
 - cairo-dock-icon-factory.h, 151
- cairo_dock_load_image_buffer
 - cairo-dock-image-buffer.h, 154
- cairo_dock_load_image_buffer_from_surface
 - cairo-dock-image-buffer.h, 155
- cairo_dock_load_image_buffer_full
 - cairo-dock-image-buffer.h, 155
- cairo_dock_load_launchers_from_dir
 - cairo-dock-launcher-manager.h, 162
- cairo_dock_load_module

- cairo-dock-module-manager.h, 165
- cairo_dock_load_modules_in_directory
 - cairo-dock-module-manager.h, 165
- cairo_dock_load_textured_font
 - cairo-dock-opengl-font.h, 169
- cairo_dock_load_textured_font_from_image
 - cairo-dock-opengl-font.h, 169
- cairo_dock_lock_desklet_position
 - cairo-dock-desklet-factory.h, 94
- cairo_dock_make_emblem
 - cairo-dock-emblem.h, 127
- cairo_dock_make_emblem_from_surface
 - cairo-dock-emblem.h, 127
- cairo_dock_make_emblem_from_texture
 - cairo-dock-emblem.h, 128
- cairo_dock_merge_conf_files
 - cairo-dock-keyfile-utilities.h, 159
- cairo_dock_new_desklet
 - cairo-dock-desklet-factory.h, 92
- cairo_dock_new_dialog
 - cairo-dock-dialog-factory.h, 100
- cairo_dock_new_gl_path
 - cairo-dock-opengl-path.h, 171
- cairo_dock_new_icon
 - cairo-dock-icon-factory.h, 151
- cairo_dock_new_launcher_icon
 - cairo-dock-launcher-factory.h, 161
- cairo_dock_new_task
 - cairo-dock-task.h, 194
- cairo_dock_new_task_full
 - cairo-dock-task.h, 195
- cairo_dock_notify_drop_data
 - cairo-dock-container.h, 77
- cairo_dock_notify_on_object
 - cairo-dock-notifications.h, 166
- cairo_dock_open_key_file
 - cairo-dock-keyfile-utilities.h, 159
- cairo_dock_package_current_theme
 - cairo-dock-themes-manager.h, 198
- cairo_dock_play_sound
 - cairo-dock-applet-facility.h, 68
- cairo_dock_pop_down
 - cairo-dock-animations.h, 40
- cairo_dock_pop_up
 - cairo-dock-animations.h, 40
- cairo_dock_popup_menu_on_container
 - cairo-dock-container.h, 75
- cairo_dock_popup_menu_on_icon
 - cairo-dock-container.h, 78
- cairo_dock_print_overlay_on_icon_from_image
 - cairo-dock-overlay.h, 179
- cairo_dock_print_overlay_on_icon_from_surface
 - cairo-dock-overlay.h, 179
- cairo_dock_redraw_container
 - cairo-dock-container.h, 76
- cairo_dock_redraw_container_area
 - cairo-dock-container.h, 76
- cairo_dock_redraw_icon
 - cairo-dock-container.h, 77
- cairo_dock_refresh_data_renderer
 - cairo-dock-data-renderer.h, 85
- cairo_dock_register_class
 - cairo-dock-class-manager.h, 71
- cairo_dock_register_notification_on_object
 - cairo-dock-notifications.h, 167
- cairo_dock_register_service_name
 - cairo-dock-dbus.h, 86
- cairo_dock_relaunch_task_immediately
 - cairo-dock-task.h, 196
- cairo_dock_reload_buffers_in_all_docks
 - cairo-dock-dock-manager.h, 119
- cairo_dock_reload_current_module_widget
 - cairo-dock-gui-manager.h, 137
- cairo_dock_reload_data_renderer_on_icon
 - cairo-dock-data-renderer.h, 85
- cairo_dock_reload_desklets_decorations
 - cairo-dock-desklet-manager.h, 96
- cairo_dock_reload_generic_gui
 - cairo-dock-gui-manager.h, 139
- cairo_dock_reload_launcher
 - cairo-dock-launcher-manager.h, 162
- cairo_dock_reload_module
 - cairo-dock-module-factory.h, 164
- cairo_dock_reload_module_instance
 - cairo-dock-module-factory.h, 164
- cairo_dock_reload_one_root_dock
 - cairo-dock-dock-manager.h, 119
- cairo_dock_remove_automatic_separators
 - cairo-dock-dock-factory.h, 114
- cairo_dock_remove_data_renderer_on_icon
 - cairo-dock-data-renderer.h, 84
- cairo_dock_remove_dialog_if_any
 - cairo-dock-dialog-manager.h, 102
- cairo_dock_remove_dialog_if_any_full
 - cairo-dock-dialog-manager.h, 102
- cairo_dock_remove_group_key_from_conf_file
 - cairo-dock-keyfile-utilities.h, 160
- cairo_dock_remove_html_spaces
 - cairo-dock-desktop-file-factory.h, 97
- cairo_dock_remove_icon_from_dock
 - cairo-dock-dock-factory.h, 113
- cairo_dock_remove_icons_from_dock
 - cairo-dock-dock-factory.h, 114
- cairo_dock_remove_notification_func_on_object
 - cairo-dock-notifications.h, 167
- cairo_dock_remove_one_icon_from_dock
 - cairo-dock-dock-factory.h, 112
- cairo_dock_remove_overlay_at_position
 - cairo-dock-overlay.h, 178
- cairo_dock_remove_quick_info
 - cairo-dock-icon-facility.h, 142
- cairo_dock_remove_root_dock_config
 - cairo-dock-dock-manager.h, 119
- cairo_dock_remove_transition_on_icon
 - cairo-dock-animations.h, 41
- cairo_dock_remove_version_from_string

- cairo-dock-X-utilities.h, 202
- cairo_dock_rename_dock
 - cairo-dock-dock-manager.h, 118
- cairo_dock_render_new_data_on_icon
 - cairo-dock-data-renderer.h, 84
- cairo_dock_render_one_icon
 - cairo-dock-draw.h, 126
- cairo_dock_render_one_icon_opengl
 - cairo-dock-draw-opengl.h, 122
- cairo_dock_render_particles
 - cairo-dock-particle-system.h, 185
- cairo_dock_render_particles_full
 - cairo-dock-particle-system.h, 185
- cairo_dock_request_icon_animation
 - cairo-dock-animations.h, 40
- cairo_dock_resize_data_renderer_history
 - cairo-dock-data-renderer.h, 85
- cairo_dock_rotate_surface
 - cairo-dock-surface-factory.h, 191
- cairo_dock_search_container_from_icon
 - cairo-dock-container.h, 77
- cairo_dock_search_dock_from_name
 - cairo-dock-dock-manager.h, 117
- cairo_dock_search_dock_name
 - cairo-dock-dock-manager.h, 117
- cairo_dock_search_icon_pointing_on_dock
 - cairo-dock-dock-manager.h, 117
- cairo_dock_search_icon_s_path
 - cairo-dock-icon-manager.h, 153
- cairo_dock_search_icon_size
 - cairo-dock-icon-manager.h, 153
- cairo_dock_search_image_s_path
 - cairo-dock-image-buffer.h, 155
- cairo_dock_search_window_covering_dock
 - cairo-dock-applications-manager.h, 69
- cairo_dock_search_window_overlapping_dock
 - cairo-dock-applications-manager.h, 69
- cairo_dock_set_all_desklets_visible
 - cairo-dock-desklet-manager.h, 96
- cairo_dock_set_all_views_to_default
 - cairo-dock-dock-manager.h, 119
- cairo_dock_set_data_from_class
 - cairo-dock-class-manager.h, 71
- cairo_dock_set_desklet_accessibility
 - cairo-dock-desklet-factory.h, 93
- cairo_dock_set_desklet_margin
 - cairo-dock-desklet-factory.h, 92
- cairo_dock_set_desklet_sticky
 - cairo-dock-desklet-factory.h, 93
- cairo_dock_set_desklets_visibility_to_default
 - cairo-dock-desklet-manager.h, 96
- cairo_dock_set_dock_visibility
 - cairo-dock-dock-manager.h, 119
- cairo_dock_set_emblem_position
 - cairo-dock-emblem.h, 127
- cairo_dock_set_icon_always_visible
 - cairo-dock-icon-facility.h, 142
- cairo_dock_set_icon_name
 - cairo-dock-icon-facility.h, 148
- cairo_dock_set_icon_name_printf
 - cairo-dock-icon-facility.h, 148
- cairo_dock_set_icon_static
 - cairo-dock-icon-facility.h, 141
- cairo_dock_set_icon_surface
 - cairo-dock-applet-facility.h, 50
- cairo_dock_set_icon_surface_full
 - cairo-dock-applet-facility.h, 66
- cairo_dock_set_icon_surface_with_reflect
 - cairo-dock-applet-facility.h, 67
- cairo_dock_set_image_on_icon
 - cairo-dock-applet-facility.h, 67
- cairo_dock_set_image_on_icon_with_default
 - cairo-dock-applet-facility.h, 67
- cairo_dock_set_launcher_class
 - cairo-dock-launcher-factory.h, 160
- cairo_dock_set_normal_task_frequency
 - cairo-dock-task.h, 197
- cairo_dock_set_ortho_view
 - cairo-dock-opengl.h, 176
- cairo_dock_set_perspective_view
 - cairo-dock-opengl.h, 176
- cairo_dock_set_quick_info
 - cairo-dock-icon-facility.h, 148
- cairo_dock_set_quick_info_printf
 - cairo-dock-icon-facility.h, 148
- cairo_dock_set_status_message
 - cairo-dock-gui-manager.h, 139
- cairo_dock_set_status_message_printf
 - cairo-dock-gui-manager.h, 139
- cairo_dock_set_transition_on_icon
 - cairo-dock-animations.h, 41
- cairo_dock_show_demand_and_wait
 - cairo-dock-dialog-manager.h, 106
- cairo_dock_show_desklet
 - cairo-dock-desklet-factory.h, 93
- cairo_dock_show_dialog_and_wait
 - cairo-dock-dialog-manager.h, 106
- cairo_dock_show_dialog_full
 - cairo-dock-dialog-manager.h, 103
- cairo_dock_show_dialog_with_entry
 - cairo-dock-dialog-manager.h, 105
- cairo_dock_show_dialog_with_question
 - cairo-dock-dialog-manager.h, 105
- cairo_dock_show_dialog_with_value
 - cairo-dock-dialog-manager.h, 105
- cairo_dock_show_general_message
 - cairo-dock-dialog-manager.h, 108
- cairo_dock_show_subdock
 - cairo-dock-dock-facility.h, 110
- cairo_dock_show_temporary_dialog
 - cairo-dock-dialog-manager.h, 104
- cairo_dock_show_temporary_dialog_with_default_icon
 - cairo-dock-dialog-manager.h, 104
- cairo_dock_show_temporary_dialog_with_icon
 - cairo-dock-dialog-manager.h, 104
- cairo_dock_show_temporary_dialog_with_icon_printf

- cairo-dock-dialog-manager.h, 103
- cairo_dock_show_value_and_wait
 - cairo-dock-dialog-manager.h, 107
- cairo_dock_sort_icons_by_name
 - cairo-dock-icon-facility.h, 143
- cairo_dock_sort_icons_by_order
 - cairo-dock-icon-facility.h, 143
- cairo_dock_start_applications_manager
 - cairo-dock-applications-manager.h, 68
- cairo_dock_start_icon_animation
 - cairo-dock-animations.h, 40
- cairo_dock_steal_interactive_widget_from_desklet
 - cairo-dock-desklet-factory.h, 92
- cairo_dock_steal_interactive_widget_from_dialog
 - cairo-dock-dialog-factory.h, 100
- cairo_dock_stop_icon_animation
 - cairo-dock-animations.h, 38
- cairo_dock_stop_task
 - cairo-dock-task.h, 195
- cairo_dock_string_is_adress
 - cairo-dock-container.h, 77
- cairo_dock_strings_differ
 - cairo-dock-image-buffer.h, 154
- cairo_dock_stroke_gl_path
 - cairo-dock-opengl-path.h, 174
- cairo_dock_task_is_active
 - cairo-dock-task.h, 196
- cairo_dock_task_is_running
 - cairo-dock-task.h, 196
- cairo_dock_toggle_dialog_visibility
 - cairo-dock-dialog-manager.h, 108
- cairo_dock_trigger_icon_removal_from_dock
 - cairo-dock-animations.h, 40
- cairo_dock_trigger_shortkey
 - cairo-dock-keybinder.h, 158
- cairo_dock_unhide_dialog
 - cairo-dock-dialog-manager.h, 108
- cairo_dock_unload_image_buffer
 - cairo-dock-image-buffer.h, 156
- cairo_dock_update_conf_file
 - cairo-dock-keyfile-utilities.h, 160
- cairo_dock_update_default_particle_system
 - cairo-dock-particle-system.h, 186
- cairo_dock_update_dock_size
 - cairo-dock-dock-facility.h, 109
- cairo_dock_update_icon_s_container_name
 - cairo-dock-icon-facility.h, 147
- cairo_dock_update_icon_texture
 - cairo-dock-draw-opengl.h, 123
- cairo_dock_update_label_texture
 - cairo-dock-draw-opengl.h, 123
- cairo_dock_update_quick_info_texture
 - cairo-dock-draw-opengl.h, 123
- cairo_dock_upgrade_conf_file_full
 - cairo-dock-keyfile-utilities.h, 159
- cairo_dock_wm_present_class
 - cairo-dock-X-manager.h, 200
- cairo_dock_wm_present_desktops
 - cairo-dock-X-manager.h, 201
- cairo_dock_wm_present_windows
 - cairo-dock-X-manager.h, 201
- cairo_dock_wm_register_backend
 - cairo-dock-X-manager.h, 200
- cairo_dock_wm_set_on_widget_layer
 - cairo-dock-X-manager.h, 201
- cairo_dock_wm_show_widget_layer
 - cairo-dock-X-manager.h, 201
- cairo_dock_write_keys_to_file
 - cairo-dock-keyfile-utilities.h, 159
- cairo_dock_zoom_out_desklet
 - cairo-dock-desklet-factory.h, 93
- CairoContainerNotifications
 - cairo-dock-container.h, 76
- CairoDeskletNotifications
 - cairo-dock-desklet-manager.h, 95
- CairoDeskletVisibility
 - cairo-dock-desklet-factory.h, 91
- CairoDesktopNotifications
 - cairo-dock-X-manager.h, 200
- CairoDockGUIWidgetType
 - cairo-dock-gui-factory.h, 135
- CairoDockInfoDisplay
 - cairo-dock-applet-facility.h, 66
- CairoDockLoadImageModifier
 - cairo-dock-surface-factory.h, 188
- CairoDockPackageType
 - cairo-dock-packages.h, 181
- CairoDockTypeGraph
 - cairo-dock-graph.h, 133
- CairoDocksNotifications
 - cairo-dock-dock-manager.h, 116
- CairoIconNotifications
 - cairo-dock-icon-manager.h, 152
- CairoObjectNotifications
 - cairo-dock-object.h, 167
- cd_keybinder_bind
 - cairo-dock-keybinder.h, 157
- cd_keybinder_could_grab
 - cairo-dock-keybinder.h, 157
- cd_keybinder_rebind
 - cairo-dock-keybinder.h, 157
- cd_keybinder_unbind
 - cairo-dock-keybinder.h, 157
- D_
 - cairo-dock-applet-facility.h, 66
- gldi_container_enable_drop
 - cairo-dock-container.h, 75
- gldi_glx_apply_desktop_background
 - cairo-dock-opengl.h, 176
- gldi_glx_init_container
 - cairo-dock-opengl.h, 177
- NOTIFICATION_BUILD_CONTAINER_MENU
 - cairo-dock-container.h, 76
- NOTIFICATION_BUILD_ICON_MENU

- cairo-dock-container.h, [76](#)
- NOTIFICATION_CLICK_ICON
 - cairo-dock-container.h, [76](#)
- NOTIFICATION_CONFIGURE_DESKLET
 - cairo-dock-desklet-manager.h, [95](#)
- NOTIFICATION_DESKTOP_CHANGED
 - cairo-dock-X-manager.h, [200](#)
- NOTIFICATION_DESKTOP_VISIBILITY_CHANGED
 - cairo-dock-X-manager.h, [200](#)
- NOTIFICATION_DESTROY
 - cairo-dock-object.h, [167](#)
- NOTIFICATION_DOUBLE_CLICK_ICON
 - cairo-dock-container.h, [76](#)
- NOTIFICATION_DROP_DATA
 - cairo-dock-container.h, [76](#)
- NOTIFICATION_ENTER_DESKLET
 - cairo-dock-desklet-manager.h, [95](#)
- NOTIFICATION_ENTER_DOCK
 - cairo-dock-dock-manager.h, [116](#)
- NOTIFICATION_ENTER_ICON
 - cairo-dock-container.h, [76](#)
- NOTIFICATION_ICON_MOVED
 - cairo-dock-dock-manager.h, [116](#)
- NOTIFICATION_INSERT_ICON
 - cairo-dock-dock-manager.h, [116](#)
- NOTIFICATION_KBD_STATE_CHANGED
 - cairo-dock-X-manager.h, [200](#)
- NOTIFICATION_KEY_PRESSED
 - cairo-dock-container.h, [76](#)
- NOTIFICATION_LEAVE_DESKLET
 - cairo-dock-desklet-manager.h, [95](#)
- NOTIFICATION_LEAVE_DOCK
 - cairo-dock-dock-manager.h, [116](#)
- NOTIFICATION_MIDDLE_CLICK_ICON
 - cairo-dock-container.h, [76](#)
- NOTIFICATION_MOUSE_MOVED
 - cairo-dock-container.h, [76](#)
- NOTIFICATION_NEW_DESKLET
 - cairo-dock-desklet-manager.h, [95](#)
- NOTIFICATION_PRE_RENDER_ICON
 - cairo-dock-icon-manager.h, [152](#)
- NOTIFICATION_REMOVE_ICON
 - cairo-dock-dock-manager.h, [116](#)
- NOTIFICATION_RENDER
 - cairo-dock-container.h, [76](#)
- NOTIFICATION_RENDER_ICON
 - cairo-dock-icon-manager.h, [152](#)
- NOTIFICATION_REQUEST_ICON_ANIMATION
 - cairo-dock-icon-manager.h, [152](#)
- NOTIFICATION_SCREEN_GEOMETRY_ALTERED
 - cairo-dock-X-manager.h, [200](#)
- NOTIFICATION_SCROLL_ICON
 - cairo-dock-container.h, [76](#)
- NOTIFICATION_START_DRAG_DATA
 - cairo-dock-container.h, [76](#)
- NOTIFICATION_STOP_ICON
 - cairo-dock-icon-manager.h, [152](#)
- NOTIFICATION_UNFOLD_SUBDOCK
 - cairo-dock-icon-manager.h, [152](#)
- NOTIFICATION_UPDATE
 - cairo-dock-container.h, [76](#)
- NOTIFICATION_UPDATE_ICON
 - cairo-dock-icon-manager.h, [152](#)
- NOTIFICATION_UPDATE_ICON_SLOW
 - cairo-dock-icon-manager.h, [152](#)
- NOTIFICATION_UPDATE_SLOW
 - cairo-dock-container.h, [76](#)
- NOTIFICATION_WINDOW_ACTIVATED
 - cairo-dock-X-manager.h, [200](#)
- NOTIFICATION_WINDOW_CONFIGURED
 - cairo-dock-X-manager.h, [200](#)
- NOTIFICATION_WINDOW_PROPERTY_CHANGED
 - cairo-dock-X-manager.h, [200](#)