

Cairo-Dock

Generated by Doxygen 1.9.8

1 Cairo-Dock's API documentation.	1
1.1 Introduction	2
1.2 Installation	2
1.3 Main structures	3
1.3.1 Objects	3
1.3.2 Managers	3
1.3.3 Containers	3
1.3.4 Icons	3
1.3.5 Dock	3
1.3.6 Desklet	3
1.3.7 Dialog	4
1.3.8 Modules	4
1.3.9 Module-Instances	4
1.3.10 Drawing with cairo/opengl	4
1.3.11 Windows management	4
1.3.12 Display scaling	5
1.4 External Modules	5
1.4.1 Create a new applet	5
1.4.2 First steps	6
1.4.3 Go further	7
1.4.4 How can I take advantage of the OpenGL ?	8
1.4.5 How can I animate my applet to make it more lively ?	8
1.4.6 I have heavy treatments to do, how can I make them without slowing the dock ?	8
1.4.7 Key binding	9
1.4.8 I need more than one icon, how can I easily get more ?	9
1.4.9 How do I provide translations (internationalization) for my applet ?	9
1.5 Advanced fonctionnalités	9
1.5.1 How can I make my own widgets in the config panel ?	9
1.5.2 How can my applet control the window of an application ?	10
1.5.3 How can I render some numerical values on my icon ?	10
1.5.4 How can I make my applet multi-instanciable ?	10
1.5.5 How can I draw anywhere on the dock, not only on my icon ?	10
1.5.6 Applets with advanced initialization steps or providing core functionality	11
1.5.7 Auto-loaded applets	11
2 Data Structure Index	13
2.1 Data Structures	13
3 File Index	17
3.1 File List	17
4 Data Structure Documentation	19
4.1 _CairoDataRenderer Struct Reference	19

4.1.1 Detailed Description	20
4.2 _CairoDataRendererAttribute Struct Reference	20
4.2.1 Detailed Description	21
4.3 _CairoDataRendererInterface Struct Reference	21
4.3.1 Detailed Description	22
4.4 _CairoDesklet Struct Reference	22
4.4.1 Detailed Description	22
4.5 _CairoDeskletAttr Struct Reference	22
4.5.1 Detailed Description	22
4.6 _CairoDeskletDecoration Struct Reference	22
4.6.1 Detailed Description	23
4.7 _CairoDeskletRenderer Struct Reference	23
4.7.1 Detailed Description	23
4.8 _CairoDialog Struct Reference	23
4.8.1 Detailed Description	24
4.9 _CairoDialogDecorator Struct Reference	24
4.9.1 Detailed Description	24
4.10 _CairoDialogRenderer Struct Reference	24
4.10.1 Detailed Description	25
4.11 _CairoDock Struct Reference	25
4.11.1 Detailed Description	27
4.11.2 Field Documentation	27
4.11.2.1 iNumScreen	27
4.12 _CairoDockDesktopEnvBackend Struct Reference	28
4.12.1 Detailed Description	28
4.13 _CairoDockGLConfig Struct Reference	28
4.13.1 Detailed Description	28
4.14 _CairoDockGLFont Struct Reference	28
4.14.1 Detailed Description	28
4.15 _CairoDockGLPath Struct Reference	29
4.15.1 Detailed Description	29
4.16 _CairoDockGroupKeyWidget Struct Reference	29
4.16.1 Detailed Description	29
4.17 _CairoDockGuiBackend Struct Reference	29
4.17.1 Detailed Description	30
4.18 _CairoDockHidingEffect Struct Reference	30
4.18.1 Detailed Description	30
4.19 _CairoDockImageBuffer Struct Reference	30
4.19.1 Detailed Description	31
4.20 _CairoDockPackage Struct Reference	31
4.20.1 Detailed Description	31
4.21 _CairoDockRenderer Struct Reference	32

4.21.1 Detailed Description	32
4.22 _CairoDockTransition Struct Reference	32
4.22.1 Detailed Description	33
4.23 _CairoGraphAttribute Struct Reference	33
4.23.1 Detailed Description	34
4.24 _CairoIconContainerRenderer Struct Reference	34
4.24.1 Detailed Description	34
4.25 _CairoOverlay Struct Reference	34
4.25.1 Detailed Description	35
4.26 _CairoParticle Struct Reference	35
4.26.1 Detailed Description	36
4.27 _CairoParticleSystem Struct Reference	36
4.27.1 Detailed Description	36
4.28 _CairoProgressBarAttribute Struct Reference	36
4.28.1 Detailed Description	37
4.29 _GldiChildProcessManagerBackend Struct Reference	37
4.29.1 Detailed Description	37
4.29.2 Field Documentation	37
4.29.2.1 spawn_app	37
4.30 _GldiContainer Struct Reference	38
4.30.1 Detailed Description	39
4.31 _GldiContainerManagerBackend Struct Reference	39
4.31.1 Detailed Description	39
4.31.2 Field Documentation	39
4.31.2.1 init_layer	39
4.31.2.2 set_keep_below	40
4.31.2.3 move_resize_dock	40
4.31.2.4 update_polling_screen_edge	40
4.31.2.5 dock_handle_leave	40
4.31.2.6 dock_check_if_mouse_inside_linear	40
4.31.2.7 adjust_aimed_point	40
4.32 _GldiDesktopBackground Struct Reference	41
4.32.1 Detailed Description	41
4.33 _GldiDesktopManagerBackend Struct Reference	41
4.33.1 Detailed Description	41
4.34 _GldiManager Struct Reference	41
4.34.1 Detailed Description	42
4.35 _GldiModule Struct Reference	42
4.35.1 Detailed Description	42
4.36 _GldiModuleInstance Struct Reference	42
4.36.1 Detailed Description	43
4.37 _GldiModuleInterface Struct Reference	43

4.37.1 Detailed Description	44
4.37.2 Field Documentation	44
4.37.2.1 initModule	44
4.37.2.2 stopModule	44
4.37.2.3 reloadModule	44
4.37.2.4 read_conf_file	44
4.37.2.5 reset_config	45
4.37.2.6 reset_data	45
4.37.2.7 load_custom_widget	45
4.38 _GldiObject Struct Reference	45
4.38.1 Detailed Description	45
4.39 _GldiObjectManager Struct Reference	45
4.39.1 Detailed Description	46
4.40 _GldiTask Struct Reference	46
4.40.1 Detailed Description	46
4.41 _GldiTextDescription Struct Reference	46
4.41.1 Detailed Description	47
4.42 _GldiVisitCard Struct Reference	47
4.42.1 Detailed Description	47
4.43 _GldiWindowActor Struct Reference	48
4.43.1 Detailed Description	48
4.44 _GldiWindowManagerBackend Struct Reference	48
4.44.1 Detailed Description	48
4.45 _Icon Struct Reference	48
4.45.1 Detailed Description	49
4.46 _IconInterface Struct Reference	49
4.46.1 Detailed Description	50
4.47 GldiAppInfo Struct Reference	50
4.47.1 Detailed Description	50
5 File Documentation	51
5.1 cairo-dock-animations.h File Reference	51
5.1.1 Detailed Description	52
5.1.2 Macro Definition Documentation	52
5.1.2.1 cairo_dock_container_is_animating	52
5.1.2.2 cairo_dock_animation_will_be_visible	52
5.1.2.3 gldi_icon_stop_animation	53
5.1.2.4 cairo_dock_get_animation_delta_t	53
5.1.2.5 cairo_dock_get_slow_animation_delta_t	53
5.1.2.6 cairo_dock_has_transition	53
5.1.2.7 cairo_dock_get_transition_count	54
5.1.2.8 cairo_dock_get_transition_elapsed_time	54

5.1.2.9 cairo_dock_get_transition_fraction	54
5.1.3 Function Documentation	55
5.1.3.1 cairo_dock_pop_up()	55
5.1.3.2 cairo_dock_pop_down()	55
5.1.3.3 cairo_dock_launch_animation()	55
5.1.3.4 gldi_icon_start_animation()	55
5.1.3.5 gldi_icon_request_animation()	56
5.1.3.6 gldi_icon_request_attention()	56
5.1.3.7 gldi_icon_stop_attention()	56
5.1.3.8 cairo_dock_trigger_icon_removal_from_dock()	57
5.1.3.9 cairo_dock_set_transition_on_icon()	57
5.1.3.10 cairo_dock_remove_transition_on_icon()	57
5.2 cairo-dock-applet-canvas.h File Reference	58
5.2.1 Detailed Description	59
5.2.2 Macro Definition Documentation	59
5.2.2.1 CD_APPLET_DEFINE_ALL_BEGIN	59
5.2.2.2 CD_APPLET_DEFINE_END	59
5.2.2.3 CD_APPLET_DEFINITION	60
5.2.2.4 CD_APPLET_DEFINE2_ALL_BEGIN	60
5.2.2.5 CD_APPLET_INIT_ALL_BEGIN	60
5.2.2.6 CD_APPLET_INIT_END	60
5.2.2.7 CD_APPLET_STOP_BEGIN	60
5.2.2.8 CD_APPLET_STOP_END	61
5.2.2.9 CD_APPLET_RELOAD_ALL_BEGIN	61
5.2.2.10 CD_APPLET_RELOAD_END	61
5.2.2.11 CD_APPLET_GET_CONFIG_ALL_BEGIN	61
5.2.2.12 CD_APPLET_GET_CONFIG_END	61
5.2.2.13 CD_APPLET_RESET_CONFIG_ALL_BEGIN	61
5.2.2.14 CD_APPLET_RESET_CONFIG_ALL_END	61
5.2.2.15 CD_APPLET_RESET_DATA_BEGIN	61
5.2.2.16 CD_APPLET_RESET_DATA_ALL_END	62
5.2.2.17 CD_APPLET_ON_CLICK_BEGIN	62
5.2.2.18 CD_APPLET_ON_CLICK_END	62
5.2.2.19 CD_APPLET_ON_BUILD_MENU_BEGIN	62
5.2.2.20 CD_APPLET_ON_BUILD_MENU_END	62
5.2.2.21 CD_APPLET_ON_MIDDLE_CLICK_BEGIN	62
5.2.2.22 CD_APPLET_ON_MIDDLE_CLICK_END	62
5.2.2.23 CD_APPLET_ON_DOUBLE_CLICK_BEGIN	62
5.2.2.24 CD_APPLET_ON_DOUBLE_CLICK_END	63
5.2.2.25 CD_APPLET_ON_DROP_DATA_BEGIN	63
5.2.2.26 CD_APPLET_ON_DROP_DATA_END	63
5.2.2.27 CD_APPLET_ON_SCROLL_BEGIN	63

5.2.2.28 CD_APPLET_ON_SCROLL_END	63
5.2.2.29 CD_APPLET_ON_UPDATE_ICON_BEGIN	63
5.2.2.30 CD_APPLET_ON_UPDATE_ICON_END	63
5.2.2.31 CD_APPLET_SKIP_UPDATE_ICON	63
5.2.2.32 CD_APPLET_STOP_UPDATE_ICON	64
5.2.2.33 CD_APPLET_PAUSE_UPDATE_ICON	64
5.2.2.34 CD_APPLET_REGISTER_FOR_CLICK_EVENT	64
5.2.2.35 CD_APPLET_UNREGISTER_FOR_CLICK_EVENT	64
5.2.2.36 CD_APPLET_REGISTER_FOR_BUILD_MENU_EVENT	64
5.2.2.37 CD_APPLET_UNREGISTER_FOR_BUILD_MENU_EVENT	64
5.2.2.38 CD_APPLET_REGISTER_FOR_MIDDLE_CLICK_EVENT	64
5.2.2.39 CD_APPLET_UNREGISTER_FOR_MIDDLE_CLICK_EVENT	64
5.2.2.40 CD_APPLET_REGISTER_FOR_DOUBLE_CLICK_EVENT	65
5.2.2.41 CD_APPLET_UNREGISTER_FOR_DOUBLE_CLICK_EVENT	65
5.2.2.42 CD_APPLET_REGISTER_FOR_DROP_DATA_EVENT	65
5.2.2.43 CD_APPLET_UNREGISTER_FOR_DROP_DATA_EVENT	65
5.2.2.44 CD_APPLET_REGISTER_FOR_SCROLL_EVENT	65
5.2.2.45 CD_APPLET_UNREGISTER_FOR_SCROLL_EVENT	65
5.2.2.46 CD_APPLET_REGISTER_FOR_UPDATE_ICON_SLOW_EVENT	65
5.2.2.47 CD_APPLET_UNREGISTER_FOR_UPDATE_ICON_SLOW_EVENT	65
5.2.2.48 CD_APPLET_REGISTER_FOR_UPDATE_ICON_EVENT	66
5.2.2.49 CD_APPLET_UNREGISTER_FOR_UPDATE_ICON_EVENT	66
5.3 cairo-dock-applet-facility.h File Reference	66
5.3.1 Detailed Description	68
5.3.2 Macro Definition Documentation	68
5.3.2.1 cairo_dock_set_icon_surface	68
5.3.2.2 CD_CONFIG_GET_BOOLEAN_WITH_DEFAULT	69
5.3.2.3 CD_CONFIG_GET_BOOLEAN	69
5.3.2.4 CD_CONFIG_GET_INTEGER_WITH_DEFAULT	69
5.3.2.5 CD_CONFIG_GET_INTEGER	70
5.3.2.6 CD_CONFIG_GET_DOUBLE_WITH_DEFAULT	70
5.3.2.7 CD_CONFIG_GET_DOUBLE	71
5.3.2.8 CD_CONFIG_GET_INTEGER_LIST	71
5.3.2.9 CD_CONFIG_GET_STRING_WITH_DEFAULT	71
5.3.2.10 CD_CONFIG_GET_STRING	72
5.3.2.11 CD_CONFIG_GET_FILE_PATH	72
5.3.2.12 CD_CONFIG_GET_STRING_LIST_WITH_DEFAULT	73
5.3.2.13 CD_CONFIG_GET_STRING_LIST	73
5.3.2.14 CD_CONFIG_GET_COLOR_RGBA_WITH_DEFAULT	73
5.3.2.15 CD_CONFIG_GET_COLOR_RGBA	74
5.3.2.16 CD_CONFIG_GET_COLOR_RGB_WITH_DEFAULT	74
5.3.2.17 CD_CONFIG_GET_COLOR_RGB	74

5.3.2.18 CD_CONFIG_GET_COLOR	76
5.3.2.19 CD_CONFIG_GET_THEME_PATH	76
5.3.2.20 CD_CONFIG_GET_GAUGE_THEME	76
5.3.2.21 CD_CONFIG_RENAME_GROUP	77
5.3.2.22 CD_APPLET_ADD_SUB_MENU_WITH_IMAGE	77
5.3.2.23 CD_APPLET_ADD_SUB_MENU	77
5.3.2.24 CD_APPLET_ADD_IN_MENU_WITH_TOOLTIP_AND_DATA	78
5.3.2.25 CD_APPLET_ADD_IN_MENU_WITH_STOCK_AND_DATA	78
5.3.2.26 CD_APPLET_ADD_IN_MENU_WITH_DATA	79
5.3.2.27 CD_APPLET_ADD_IN_MENU	79
5.3.2.28 CD_APPLET_ADD_IN_MENU_WITH_STOCK	79
5.3.2.29 CD_APPLET_ADD_SEPARATOR_IN_MENU	80
5.3.2.30 CD_APPLET_POPUP_MENU_ON_MY_ICON	80
5.3.2.31 CD_APPLET_RELOAD_CONFIG_PANEL	80
5.3.2.32 CD_APPLET_RELOAD_CONFIG_PANEL_WITH_PAGE	80
5.3.2.33 CD_APPLET_MY_CONF_FILE	80
5.3.2.34 CD_APPLET_MY_KEY_FILE	80
5.3.2.35 CD_APPLET_MY_CONFIG_CHANGED	81
5.3.2.36 CD_APPLET_MY_CONTAINER_TYPE_CHANGED	81
5.3.2.37 CD_APPLET_MY_OLD_CONTAINER	81
5.3.2.38 CD_APPLET_CLICKED_ICON	81
5.3.2.39 CD_APPLET_CLICKED_CONTAINER	81
5.3.2.40 CD_APPLET_SHIFT_CLICK	81
5.3.2.41 CD_APPLET_CTRL_CLICK	81
5.3.2.42 CD_APPLET_ALT_CLICK	81
5.3.2.43 CD_APPLET_MY_MENU	82
5.3.2.44 CD_APPLET_RECEIVED_DATA	82
5.3.2.45 CD_APPLET_SCROLL_UP	82
5.3.2.46 CD_APPLET_SCROLL_DOWN	82
5.3.2.47 CD_APPLET_BIND_KEY	82
5.3.2.48 CD_APPLET_REDRAW_MY_ICON	83
5.3.2.49 CAIRO_DOCK_REDRAW_MY_CONTAINER	83
5.3.2.50 CD_APPLET_LOAD_SURFACE_FOR_MY_APPLET	83
5.3.2.51 CD_APPLET_LOAD_SURFACE_FOR_MY_APPLET_WITH_DEFAULT	83
5.3.2.52 CD_APPLET_SET_SURFACE_ON_MY_ICON	84
5.3.2.53 CD_APPLET_SET_IMAGE_ON_MY_ICON	84
5.3.2.54 CD_APPLET_SET_USER_IMAGE_ON_MY_ICON	84
5.3.2.55 CD_APPLET_SET_DEFAULT_IMAGE_ON_MY_ICON_IF_NONE	84
5.3.2.56 CD_APPLET_SET_NAME_FOR_MY_ICON	84
5.3.2.57 CD_APPLET_SET_NAME_FOR_MY_ICON_PRINTF	86
5.3.2.58 CD_APPLET_SET_QUICK_INFO_ON_MY_ICON	86
5.3.2.59 CD_APPLET_SET_QUICK_INFO_ON_MY_ICON_PRINTF	86

5.3.2.60 CD_APPLET_SET_HOURS_MINUTES_AS_QUICK_INFO	86
5.3.2.61 CD_APPLET_SET_MINUTES_SECONDES_AS_QUICK_INFO	88
5.3.2.62 CD_APPLET_SET_SIZE_AS_QUICK_INFO	88
5.3.2.63 CD_APPLET_SET_STATIC_ICON	88
5.3.2.64 CD_APPLET_UNSET_STATIC_ICON	88
5.3.2.65 CD_APPLET_SET_ALWAYS_VISIBLE_ICON	88
5.3.2.66 CD_APPLET_ANIMATE_MY_ICON	89
5.3.2.67 CD_APPLET_STOP_ANIMATING_MY_ICON	89
5.3.2.68 CD_APPLET_DEMANDS_ATTENTION	89
5.3.2.69 CD_APPLET_STOP_DEMANDING_ATTENTION	89
5.3.2.70 CD_APPLET_GET_MY_ICON_EXTENT	89
5.3.2.71 CD_APPLET_START_DRAWING_MY_ICON	90
5.3.2.72 CD_APPLET_START_DRAWING_MY_ICON_CAIRO	90
5.3.2.73 CD_APPLET_START_DRAWING_MY_ICON_OR_RETURN	90
5.3.2.74 CD_APPLET_START_DRAWING_MY_ICON_OR_RETURN_CAIRO	90
5.3.2.75 CD_APPLET_FINISH_DRAWING_MY_ICON	90
5.3.2.76 CD_APPLET_FINISH_DRAWING_MY_ICON_CAIRO	91
5.3.2.77 CD_APPLET_ADD_OVERLAY_ON_MY_ICON	91
5.3.2.78 CD_APPLET_PRINT_OVERLAY_ON_MY_ICON	91
5.3.2.79 CD_APPLET_REMOVE_OVERLAY_ON_MY_ICON	91
5.3.2.80 CD_APPLET_ADD_DATA_RENDERER_ON_MY_ICON	92
5.3.2.81 CD_APPLET_RELOAD_MY_DATA_RENDERER	92
5.3.2.82 CD_APPLET_RENDER_NEW_DATA_ON_MY_ICON	92
5.3.2.83 CD_APPLET_REMOVE_MY_DATA_RENDERER	92
5.3.2.84 CD_APPLET_SET_MY_DATA_RENDERER_HISTORY_TO_MAX	92
5.3.2.85 CD_APPLET_MY_CONTAINER_IS_OPENGL	93
5.3.2.86 CD_APPLET_SET_DESKLET_RENDERER_WITH_DATA	93
5.3.2.87 CD_APPLET_SET_DESKLET_RENDERER	93
5.3.2.88 CD_APPLET_SET_STATIC_DESKLET	93
5.3.2.89 CD_APPLET_ALLOW_NO_CLICKABLE_DESKLET	93
5.3.2.90 CD_APPLET_DELETE_MY_ICONS_LIST	93
5.3.2.91 CD_APPLET_REMOVE_ICON_FROM_MY_ICONS_LIST	94
5.3.2.92 CD_APPLET_DETACH_ICON_FROM_MY_ICONS_LIST	94
5.3.2.93 CD_APPLET_LOAD_MY_ICONS_LIST	94
5.3.2.94 CD_APPLET_ADD_ICON_IN_MY_ICONS_LIST	95
5.3.2.95 CD_APPLET_MY_ICONS_LIST	95
5.3.2.96 CD_APPLET_MY_ICONS_LIST_CONTAINER	95
5.3.2.97 CD_APPLET_MANAGE_APPLICATION	95
5.3.2.98 D_	95
5.3.3 Enumeration Type Documentation	96
5.3.3.1 CairoDockInfoDisplay	96
5.3.4 Function Documentation	96

5.3.4.1	cairo_dock_set_icon_surface_full()	96
5.3.4.2	cairo_dock_set_image_on_icon()	96
5.3.4.3	cairo_dock_set_image_on_icon_with_default()	97
5.3.4.4	cairo_dock_get_human_readable_size()	97
5.3.4.5	cairo_dock_play_sound()	97
5.4	cairo-dock-applet-manager.h File Reference	98
5.4.1	Detailed Description	98
5.4.2	Macro Definition Documentation	98
5.4.2.1	GLDI_OBJECT_IS_APPLET_ICON	98
5.5	cairo-dock-applications-manager.h File Reference	98
5.5.1	Detailed Description	99
5.5.2	Macro Definition Documentation	99
5.5.2.1	GLDI_OBJECT_IS_APPLI_ICON	99
5.5.3	Function Documentation	99
5.5.3.1	cairo_dock_start_applications_manager()	99
5.5.3.2	cairo_dock_get_current_applis_list()	99
5.5.3.3	cairo_dock_get_current_active_icon()	100
5.5.3.4	cairo_dock_get_appli_icon()	100
5.5.3.5	cairo_dock_foreach_appli_icon()	100
5.6	cairo-dock-class-manager.h File Reference	101
5.6.1	Detailed Description	101
5.6.2	Function Documentation	101
5.6.2.1	gldi_app_info_new_from_commandline()	101
5.6.2.2	gldi_app_info_launch_action()	102
5.6.2.3	gldi_app_info_launch()	102
5.6.2.4	gldi_app_info_get_desktop_actions()	102
5.6.2.5	gldi_app_info_get_desktop_action_name()	103
5.6.2.6	gldi_app_info_get_supported_types()	103
5.6.2.7	gldi_app_info_from_desktop_app_info()	103
5.6.2.8	gldi_launch_desktop_app_info()	104
5.6.2.9	gldi_app_info_set_run_in_terminal()	104
5.6.2.10	gldi_window_foreach_inhibitor()	104
5.6.2.11	cairo_dock_get_class_app_info()	105
5.6.2.12	cairo_dock_register_class2()	105
5.6.2.13	cairo_dock_register_class()	106
5.6.2.14	cairo_dock_set_data_from_class()	106
5.7	cairo-dock-config.h File Reference	107
5.7.1	Detailed Description	107
5.7.2	Function Documentation	107
5.7.2.1	cairo_dock_load_current_theme()	107
5.7.2.2	cairo_dock_is_loading()	107
5.7.2.3	cairo_dock_decrypt_string()	107

5.7.2.4 cairo_dock_encrypt_string()	108
5.8 cairo-dock-container.h File Reference	108
5.8.1 Detailed Description	109
5.8.2 Macro Definition Documentation	110
5.8.2.1 CAIRO_DOCK_IS_CONTAINER	110
5.8.2.2 gldi_container_enable_drop	110
5.8.3 Enumeration Type Documentation	110
5.8.3.1 GldiContainerNotifications	110
5.8.4 Function Documentation	111
5.8.4.1 gldi_container_reserve_space()	111
5.8.4.2 gldi_container_get_current_desktop_index()	112
5.8.4.3 gldi_container_move()	112
5.8.4.4 gldi_container_is_active()	113
5.8.4.5 gldi_container_present()	113
5.8.4.6 gldi_container_init_layer()	113
5.8.4.7 gldi_container_is_wayland_backend()	114
5.8.4.8 gldi_container_move_resize_dock()	114
5.8.4.9 gldi_container_set_screen()	114
5.8.4.10 gldi_container_move_to_rect()	114
5.8.4.11 gldi_container_calculate_rect()	115
5.8.4.12 gldi_container_calculate_aimed_point()	115
5.8.4.13 gldi_container_calculate_aimed_point_base()	115
5.8.4.14 gldi_container_set_keep_below()	115
5.8.4.15 gldi_container_dock_handle_leave()	116
5.8.4.16 gldi_container_dock_check_if_mouse_inside_linear()	116
5.8.4.17 gldi_container_use_new_positioning_code()	116
5.8.4.18 cairo_dock_redraw_container()	116
5.8.4.19 cairo_dock_redraw_container_area()	116
5.8.4.20 cairo_dock_redraw_icon()	117
5.8.4.21 gldi_container_notify_drop_data()	117
5.8.4.22 gldi_container_build_menu()	117
5.9 cairo-dock-core.h File Reference	118
5.9.1 Detailed Description	118
5.9.2 Function Documentation	118
5.9.2.1 gldi_get_diag_msg()	118
5.10 cairo-dock-data-renderer-manager.h File Reference	118
5.10.1 Detailed Description	118
5.10.2 Macro Definition Documentation	118
5.10.2.1 GLDI_OBJECT_IS_DATA_RENDERER	118
5.10.3 Function Documentation	119
5.10.3.1 cairo_dock_get_default_data_renderer_font()	119
5.11 cairo-dock-data-renderer.h File Reference	119

5.11.1 Detailed Description	120
5.11.2 Macro Definition Documentation	120
5.11.2.1 cairo_dock_get_icon_data_renderer	120
5.11.2.2 CAIRO_DATA_RENDERER	120
5.11.2.3 cairo_data_renderer_get_data	121
5.11.2.4 CAIRO_DATA_RENDERER_ATTRIBUTE	121
5.11.2.5 cairo_data_renderer_get_nb_values	121
5.11.2.6 cairo_data_renderer_get_min_value	122
5.11.2.7 cairo_data_renderer_get_max_value	122
5.11.2.8 cairo_data_renderer_get_value	122
5.11.2.9 cairo_data_renderer_get_current_value	123
5.11.2.10 cairo_data_renderer_get_previous_value	123
5.11.2.11 cairo_data_renderer_get_normalized_value	124
5.11.2.12 cairo_data_renderer_get_normalized_current_value	124
5.11.2.13 cairo_data_renderer_get_normalized_previous_value	124
5.11.2.14 cairo_data_renderer_get_normalized_current_value_with_latency	125
5.11.2.15 cairo_data_renderer_format_value_full	125
5.11.2.16 cairo_data_renderer_format_value	125
5.11.3 Function Documentation	126
5.11.3.1 cairo_dock_get_default_data_renderer_font()	126
5.11.3.2 cairo_dock_add_new_data_renderer_on_icon()	126
5.11.3.3 cairo_dock_render_new_data_on_icon()	126
5.11.3.4 cairo_dock_remove_data_renderer_on_icon()	127
5.11.3.5 cairo_dock_reload_data_renderer_on_icon()	127
5.11.3.6 cairo_dock_resize_data_renderer_history()	127
5.11.3.7 cairo_dock_refresh_data_renderer()	127
5.12 cairo-dock-dbus.h File Reference	128
5.12.1 Detailed Description	128
5.12.2 Function Documentation	128
5.12.2.1 cairo_dock_get_session_connection()	128
5.12.2.2 cairo_dock_register_service_name()	128
5.12.2.3 cairo_dock_dbus_is_enabled()	129
5.12.2.4 cairo_dock_create_new_session_proxy()	129
5.12.2.5 cairo_dock_create_new_system_proxy()	129
5.12.2.6 cairo_dock_dbus_detect_application()	130
5.12.2.7 cairo_dock_dbus_detect_system_application()	130
5.12.2.8 cairo_dock_dbus_get_boolean()	130
5.12.2.9 cairo_dock_dbus_get_uinteger()	131
5.12.2.10 cairo_dock_dbus_get_integer()	131
5.12.2.11 cairo_dock_dbus_get_string()	131
5.12.2.12 cairo_dock_dbus_get_string_list()	132
5.12.2.13 cairo_dock_dbus_get_uchar()	132

5.12.2.14 cairo_dock_dbus_call()	132
5.13 cairo-dock-desklet-factory.h File Reference	133
5.13.1 Detailed Description	134
5.13.2 Macro Definition Documentation	134
5.13.2.1 GLDI_OBJECT_IS_DESKLET	134
5.13.2.2 CAIRO_DESKLET	134
5.13.2.3 gldi_desklet_add_interactive_widget	135
5.13.3 Enumeration Type Documentation	135
5.13.3.1 CairoDeskletVisibility	135
5.13.4 Function Documentation	135
5.13.4.1 gldi_desklet_new()	135
5.13.4.2 gldi_desklet_add_interactive_widget_with_margin()	136
5.13.4.3 gldi_desklet_set_margin()	136
5.13.4.4 gldi_desklet_steal_interactive_widget()	136
5.13.4.5 gldi_desklet_hide()	137
5.13.4.6 gldi_desklet_show()	137
5.13.4.7 gldi_desklet_set_accessibility()	137
5.13.4.8 gldi_desklet_set_sticky()	138
5.13.4.9 gldi_desklet_lock_position()	138
5.14 cairo-dock-desklet-manager.h File Reference	138
5.14.1 Detailed Description	139
5.14.2 Enumeration Type Documentation	139
5.14.2.1 CairoDeskletNotifications	139
5.14.3 Function Documentation	139
5.14.3.1 gldi_desklets_foreach()	139
5.14.3.2 gldi_desklets_foreach_icons()	140
5.14.3.3 gldi_desklets_set_visible()	140
5.14.3.4 gldi_desklets_set_visibility_to_default()	140
5.15 cairo-dock-desktop-file-db.h File Reference	140
5.15.1 Detailed Description	141
5.15.2 Function Documentation	141
5.15.2.1 gldi_desktop_file_db_init()	141
5.15.2.2 gldi_desktop_file_db_stop()	141
5.15.2.3 gldi_desktop_file_db_lookup()	141
5.16 cairo-dock-desktop-manager.h File Reference	142
5.16.1 Detailed Description	142
5.16.2 Enumeration Type Documentation	143
5.16.2.1 CairoDesktopNotifications	143
5.16.3 Function Documentation	144
5.16.3.1 gldi_desktop_manager_register_backend()	144
5.16.3.2 gldi_desktop_present_class()	144
5.16.3.3 gldi_desktop_present_windows()	145

5.16.3.4	gldi_desktop_present_desktops()	145
5.16.3.5	gldi_desktop_show_widget_layer()	145
5.16.3.6	gldi_desktop_set_on_widget_layer()	145
5.16.3.7	gldi_desktop_add_workspace()	146
5.16.3.8	gldi_desktop_remove_last_workspace()	146
5.16.3.9	gldi_desktop_get_current()	146
5.17	cairo-dock-dialog-factory.h File Reference	147
5.17.1	Detailed Description	148
5.17.2	Macro Definition Documentation	148
5.17.2.1	CAIRO_DOCK_IS_DIALOG	148
5.17.2.2	CAIRO_DIALOG	148
5.17.3	Function Documentation	149
5.17.3.1	gldi_dialog_new()	149
5.17.3.2	gldi_dialog_show()	149
5.17.3.3	gldi_dialog_show_temporary_with_icon_printf()	150
5.17.3.4	gldi_dialog_show_temporary_with_icon()	150
5.17.3.5	gldi_dialog_show_temporary()	151
5.17.3.6	gldi_dialog_show_temporary_with_default_icon()	151
5.17.3.7	gldi_dialog_show_with_question()	151
5.17.3.8	gldi_dialog_show_with_entry()	152
5.17.3.9	gldi_dialog_show_with_value()	153
5.17.3.10	gldi_dialog_show_general_message()	153
5.17.3.11	gldi_dialog_show_and_wait()	154
5.17.3.12	gldi_dialog_steal_interactive_widget()	154
5.18	cairo-dock-dialog-manager.h File Reference	155
5.18.1	Detailed Description	155
5.18.2	Function Documentation	155
5.18.2.1	gldi_dialogs_remove_on_icon()	155
5.18.2.2	gldi_dialog_hide()	156
5.18.2.3	gldi_dialog_unhide()	156
5.18.2.4	gldi_dialog_toggle_visibility()	156
5.18.2.5	gldi_dialog_leave()	156
5.19	cairo-dock-dock-facility.h File Reference	157
5.19.1	Detailed Description	157
5.19.2	Macro Definition Documentation	157
5.19.2.1	cairo_dock_get_available_docks_for_icon	157
5.19.3	Function Documentation	158
5.19.3.1	cairo_dock_update_dock_size()	158
5.19.3.2	cairo_dock_calculate_dock_icons()	158
5.19.3.3	cairo_dock_show_subdock()	158
5.19.3.4	cairo_dock_get_available_docks()	158
5.19.3.5	cairo_dock_calculate_icons_positions_at_rest_linear()	159

5.19.3.6	cairo_dock_apply_wave_effect_linear()	159
5.19.3.7	cairo_dock_get_current_dock_width_linear()	159
5.19.3.8	cairo_dock_check_if_mouse_inside_linear()	161
5.19.3.9	cairo_dock_check_can_drop_linear()	161
5.19.3.10	cairo_dock_get_first_drawn_element_linear()	161
5.20	cairo-dock-dock-factory.h File Reference	162
5.20.1	Detailed Description	162
5.20.2	Macro Definition Documentation	162
5.20.2.1	GLDI_OBJECT_IS_DOCK	162
5.20.2.2	CAIRO_DOCK	163
5.20.3	Function Documentation	163
5.20.3.1	gldi_dock_new()	163
5.20.3.2	gldi_subdock_new()	163
5.20.3.3	cairo_dock_remove_icons_from_dock()	164
5.20.3.4	gldi_dock_leave_synthetic()	164
5.20.3.5	gldi_dock_enter_synthetic()	164
5.21	cairo-dock-dock-manager.h File Reference	165
5.21.1	Detailed Description	165
5.21.2	Macro Definition Documentation	165
5.21.2.1	gldi_dock_get_name	165
5.21.3	Enumeration Type Documentation	166
5.21.3.1	CairoDocksNotifications	166
5.21.4	Function Documentation	166
5.21.4.1	gldi_dock_get_readable_name()	166
5.21.4.2	gldi_dock_get()	166
5.21.4.3	cairo_dock_search_icon_pointing_on_dock()	167
5.21.4.4	gldi_dock_rename()	167
5.21.4.5	gldi_docks_foreach()	167
5.21.4.6	gldi_docks_foreach_root()	168
5.21.4.7	gldi_icons_foreach_in_docks()	168
5.21.4.8	cairo_dock_reload_buffers_in_all_docks()	168
5.21.4.9	gldi_dock_add_conf_file_for_name()	169
5.21.4.10	gldi_dock_add_conf_file()	169
5.21.4.11	gldi_docks_redraw_all_root()	169
5.21.4.12	gldi_dock_set_visibility()	169
5.22	cairo-dock-dock-visibility.h File Reference	170
5.22.1	Detailed Description	170
5.22.2	Function Documentation	170
5.22.2.1	gldi_dock_visibility_refresh()	170
5.22.2.2	gldi_dock_has_overlapping_window()	170
5.23	cairo-dock-draw-opengl.h File Reference	170
5.23.1	Detailed Description	171

5.23.2 Macro Definition Documentation	171
5.23.2.1 cairo_dock_create_texture_from_image	171
5.23.2.2 _cairo_dock_delete_texture	171
5.23.2.3 _cairo_dock_enable_texture	172
5.23.2.4 _cairo_dock_disable_texture	172
5.23.2.5 _cairo_dock_set_alpha	172
5.23.2.6 _cairo_dock_set_blend_source	172
5.23.2.7 _cairo_dock_set_blend_alpha	172
5.23.2.8 _cairo_dock_set_blend_over	173
5.23.2.9 _cairo_dock_set_blend_pbuffer	173
5.23.2.10 _cairo_dock_apply_texture_at_size	173
5.23.2.11 _cairo_dock_apply_texture	173
5.23.2.12 _cairo_dock_apply_texture_at_size_with_alpha	173
5.23.3 Function Documentation	175
5.23.3.1 cairo_dock_render_one_icon_opengl()	175
5.23.3.2 cairo_dock_create_texture_from_surface_full()	175
5.23.3.3 cairo_dock_create_texture_from_surface()	176
5.23.3.4 cairo_dock_create_texture_from_raw_data()	176
5.23.3.5 cairo_dock_create_texture_from_image_full()	176
5.23.3.6 cairo_dock_update_icon_texture()	177
5.24 cairo-dock-draw.h File Reference	177
5.24.1 Detailed Description	177
5.24.2 Macro Definition Documentation	177
5.24.2.1 cairo_dock_erase_cairo_context	177
5.24.3 Function Documentation	178
5.24.3.1 cairo_dock_create_drawing_context_generic()	178
5.24.3.2 cairo_dock_create_drawing_context_on_container()	178
5.24.3.3 cairo_dock_create_drawing_context_on_area()	178
5.24.3.4 cairo_dock_draw_rounded_rectangle()	179
5.24.3.5 cairo_dock_draw_icon_cairo()	179
5.24.3.6 cairo_dock_render_one_icon()	179
5.24.3.7 cairo_dock_draw_string()	180
5.25 cairo-dock-file-manager.h File Reference	180
5.25.1 Detailed Description	181
5.25.2 Function Documentation	182
5.25.2.1 cairo_dock_fm_register_vfs_backend()	182
5.25.2.2 cairo_dock_fm_list_directory()	182
5.25.2.3 cairo_dock_fm_measure_directory()	182
5.25.2.4 cairo_dock_fm_get_file_info()	182
5.25.2.5 cairo_dock_fm_get_file_properties()	183
5.25.2.6 cairo_dock_fm_launch_uri()	183
5.25.2.7 cairo_dock_fm_add_monitor_full()	183

5.25.2.8 <code>cairo_dock_fm_remove_monitor_full()</code>	183
5.25.2.9 <code>cairo_dock_fm_mount_full()</code>	183
5.25.2.10 <code>cairo_dock_fm_unmount_full()</code>	184
5.25.2.11 <code>cairo_dock_fm_is_mounted()</code>	184
5.25.2.12 <code>cairo_dock_fm_can_eject()</code>	184
5.25.2.13 <code>cairo_dock_fm_eject_drive()</code>	184
5.25.2.14 <code>cairo_dock_fm_delete_file()</code>	184
5.25.2.15 <code>cairo_dock_fm_rename_file()</code>	184
5.25.2.16 <code>cairo_dock_fm_move_file()</code>	185
5.25.2.17 <code>cairo_dock_fm_create_file()</code>	185
5.25.2.18 <code>cairo_dock_fm_list_apps_for_file()</code>	185
5.25.2.19 <code>cairo_dock_fm_empty_trash()</code>	185
5.25.2.20 <code>cairo_dock_fm_get_trash_path()</code>	185
5.25.2.21 <code>cairo_dock_fm_get_desktop_path()</code>	185
5.25.2.22 <code>cairo_dock_fm_logout()</code>	186
5.25.2.23 <code>cairo_dock_fm_shutdown()</code>	186
5.25.2.24 <code>cairo_dock_fm_reboot()</code>	186
5.25.2.25 <code>cairo_dock_fm_lock_screen()</code>	186
5.25.2.26 <code>cairo_dock_fm_setup_time()</code>	186
5.25.2.27 <code>cairo_dock_fm_show_system_monitor()</code>	186
5.25.2.28 <code>cairo_dock_fm_create_icon_from_URI()</code>	186
5.25.2.29 <code>cairo_dock_get_file_size()</code>	186
5.25.2.30 <code>cairo_dock_fm_get_pid()</code>	187
5.25.2.31 <code>cairo_dock_fm_monitor_pid()</code>	187
5.25.2.32 <code>cairo_dock_fm_add_open_with_submenu()</code>	188
5.26 <code>cairo-dock-gui-factory.h</code> File Reference	188
5.26.1 Detailed Description	190
5.26.2 Enumeration Type Documentation	190
5.26.2.1 <code>CairoDockGUIWidgetType</code>	190
5.26.3 Function Documentation	192
5.26.3.1 <code>cairo_dock_gui_find_group_key_widget_in_list()</code>	192
5.26.3.2 <code>cairo_dock_gui_menu_item_add()</code>	192
5.26.3.3 <code>cairo_dock_gui_image_from_file()</code>	193
5.27 <code>cairo-dock-gui-manager.h</code> File Reference	193
5.27.1 Detailed Description	194
5.27.2 Macro Definition Documentation	194
5.27.2.1 <code>cairo_dock_reload_current_module_widget</code>	194
5.27.3 Function Documentation	194
5.27.3.1 <code>cairo_dock_set_status_message()</code>	194
5.27.3.2 <code>cairo_dock_set_status_message_printf()</code>	195
5.28 <code>cairo-dock-icon-facility.h</code> File Reference	195
5.28.1 Detailed Description	196

5.28.2 Macro Definition Documentation	196
5.28.2.1 cairo_dock_icon_is_being_inserted	196
5.28.2.2 cairo_dock_icon_is_being_removed	196
5.28.2.3 cairo_dock_get_icon_order	196
5.28.2.4 cairo_dock_get_next_element	196
5.28.2.5 cairo_dock_get_previous_element	197
5.28.2.6 cairo_dock_set_icon_static	197
5.28.2.7 cairo_dock_set_icon_always_visible	197
5.28.2.8 gldi_icon_mark_as_launching	198
5.28.2.9 gldi_icon_is_launching	198
5.28.3 Function Documentation	198
5.28.3.1 cairo_dock_get_icon_type()	198
5.28.3.2 cairo_dock_compare_icons_order()	198
5.28.3.3 cairo_dock_compare_icons_name()	199
5.28.3.4 cairo_dock_compare_icons_extension()	199
5.28.3.5 cairo_dock_sort_icons_by_order()	199
5.28.3.6 cairo_dock_sort_icons_by_name()	200
5.28.3.7 cairo_dock_get_first_icon()	200
5.28.3.8 cairo_dock_get_last_icon()	200
5.28.3.9 cairo_dock_get_first_icon_of_group()	201
5.28.3.10 cairo_dock_get_last_icon_of_group()	201
5.28.3.11 cairo_dock_get_first_icon_of_order()	201
5.28.3.12 cairo_dock_get_last_icon_of_order()	203
5.28.3.13 cairo_dock_get_pointed_icon()	203
5.28.3.14 cairo_dock_get_next_icon()	203
5.28.3.15 cairo_dock_get_previous_icon()	204
5.28.3.16 cairo_dock_get_icon_with_command()	204
5.28.3.17 cairo_dock_get_icon_with_base_uri()	204
5.28.3.18 cairo_dock_get_icon_with_name()	205
5.28.3.19 cairo_dock_get_icon_with_subdock()	205
5.28.3.20 cairo_dock_get_icon_extent()	205
5.28.3.21 cairo_dock_get_current_icon_size()	206
5.28.3.22 cairo_dock_compute_icon_area()	206
5.28.3.23 gldi_icon_set_name()	206
5.28.3.24 gldi_icon_set_name_printf()	207
5.28.3.25 gldi_icon_set_quick_info()	207
5.28.3.26 gldi_icon_set_quick_info_printf()	207
5.28.3.27 cairo_dock_icon_buffer_to_cairo()	208
5.28.3.28 cairo_dock_begin_draw_icon()	208
5.28.3.29 cairo_dock_end_draw_icon()	208
5.29 cairo-dock-icon-factory.h File Reference	209
5.29.1 Detailed Description	210

5.29.2 Macro Definition Documentation	210
5.29.2.1 CAIRO_DOCK_IS_ICON	210
5.29.2.2 CAIRO_DOCK_IS_APPLI	210
5.29.2.3 CAIRO_DOCK_IS_APPLET	211
5.29.2.4 CAIRO_DOCK_IS_MULTI_APPLI	211
5.29.2.5 CAIRO_DOCK_IS_AUTOMATIC_SEPARATOR	211
5.29.2.6 CAIRO_DOCK_IS_USER_SEPARATOR	211
5.29.2.7 CAIRO_DOCK_IS_NORMAL_APPLI	212
5.29.2.8 CAIRO_DOCK_IS_DETACHABLE_APPLET	212
5.29.3 Function Documentation	212
5.29.3.1 gldi_icon_new()	212
5.29.3.2 cairo_dock_create_dummy_launcher()	212
5.29.3.3 cairo_dock_load_icon_image()	213
5.29.3.4 cairo_dock_load_icon_text()	213
5.29.3.5 cairo_dock_load_icon_quickinfo()	213
5.29.3.6 cairo_dock_load_icon_buffers()	214
5.30 cairo-dock-icon-manager.h File Reference	214
5.30.1 Detailed Description	214
5.30.2 Enumeration Type Documentation	214
5.30.2.1 CairoIconNotifications	214
5.30.3 Function Documentation	215
5.30.3.1 gldi_icons_foreach()	215
5.30.3.2 cairo_dock_search_icon_size()	215
5.30.3.3 cairo_dock_search_icon_s_path()	215
5.31 cairo-dock-image-buffer.h File Reference	216
5.31.1 Detailed Description	217
5.31.2 Macro Definition Documentation	217
5.31.2.1 cairo_dock_load_image_buffer	217
5.31.2.2 cairo_dock_apply_image_buffer_surface	217
5.31.2.3 cairo_dock_apply_image_buffer_texture	218
5.31.3 Function Documentation	218
5.31.3.1 cairo_dock_search_image_s_path()	218
5.31.3.2 cairo_dock_load_image_buffer_full()	218
5.31.3.3 cairo_dock_load_image_buffer_from_surface()	219
5.31.3.4 cairo_dock_create_image_buffer()	219
5.31.3.5 cairo_dock_unload_image_buffer()	219
5.31.3.6 cairo_dock_free_image_buffer()	220
5.31.3.7 cairo_dock_apply_image_buffer_surface_with_offset()	220
5.31.3.8 cairo_dock_apply_image_buffer_texture_with_offset()	220
5.31.3.9 cairo_dock_apply_image_buffer_surface_at_size()	220
5.31.3.10 cairo_dock_apply_image_buffer_texture_at_size()	221
5.31.3.11 cairo_dock_create_icon_fbo()	221

5.31.3.12 <code>cairo_dock_destroy_icon_fbo()</code>	221
5.31.3.13 <code>cairo_dock_image_buffer_copy_scale()</code>	222
5.32 <code>cairo-dock-indicator-manager.h</code> File Reference	222
5.32.1 Detailed Description	222
5.33 <code>cairo-dock-keybinder.h</code> File Reference	222
5.33.1 Detailed Description	222
5.33.2 Macro Definition Documentation	222
5.33.2.1 <code>gldi_shortkey_could_grab</code>	222
5.33.3 Function Documentation	223
5.33.3.1 <code>gldi_shortkey_new()</code>	223
5.33.3.2 <code>gldi_shortkey_rebind()</code>	223
5.33.3.3 <code>cairo_dock_trigger_shortkey()</code>	224
5.34 <code>cairo-dock-keyfile-utilities.h</code> File Reference	224
5.34.1 Detailed Description	225
5.34.2 Function Documentation	225
5.34.2.1 <code>cairo_dock_open_key_file()</code>	225
5.34.2.2 <code>cairo_dock_write_keys_to_file_full()</code>	225
5.34.2.3 <code>cairo_dock_write_keys_to_new_file()</code>	225
5.34.2.4 <code>cairo_dock_merge_conf_files()</code>	225
5.34.2.5 <code>cairo_dock_upgrade_conf_file_full()</code>	227
5.34.2.6 <code>cairo_dock_get_conf_file_version()</code>	227
5.34.2.7 <code>cairo_dock_conf_file_needs_update()</code>	227
5.34.2.8 <code>cairo_dock_add_remove_element_to_key()</code>	227
5.34.2.9 <code>cairo_dock_add_group_key_to_conf_file()</code>	228
5.34.2.10 <code>cairo_dock_remove_group_key_from_conf_file()</code>	228
5.34.2.11 <code>cairo_dock_update_keyfile()</code>	228
5.35 <code>cairo-dock-launcher-manager.h</code> File Reference	228
5.35.1 Detailed Description	229
5.35.2 Macro Definition Documentation	229
5.35.2.1 <code>GLDI_OBJECT_IS_LAUNCHER_ICON</code>	229
5.35.3 Function Documentation	229
5.35.3.1 <code>gldi_launcher_add_new_full()</code>	229
5.35.3.2 <code>gldi_launcher_add_new()</code>	229
5.36 <code>cairo-dock-manager.h</code> File Reference	230
5.36.1 Detailed Description	230
5.36.2 Macro Definition Documentation	230
5.36.2.1 <code>GLDI_OBJECT_IS_MANAGER</code>	230
5.37 <code>cairo-dock-menu.h</code> File Reference	230
5.37.1 Detailed Description	231
5.37.2 Macro Definition Documentation	231
5.37.2.1 <code>gldi_submenu_new</code>	231
5.37.2.2 <code>gldi_menu_item_new</code>	231

5.37.2.3 gldi_menu_add_sub_menu	232
5.37.3 Function Documentation	232
5.37.3.1 gldi_menu_new()	232
5.37.3.2 gldi_menu_init()	232
5.37.3.3 gldi_menu_popup_full()	233
5.37.3.4 gldi_menu_item_new_full2()	233
5.37.3.5 gldi_menu_item_new_with_action()	233
5.37.3.6 gldi_menu_item_new_with_submenu()	234
5.37.3.7 gldi_menu_item_set_image()	234
5.37.3.8 gldi_menu_item_get_image()	235
5.37.3.9 gldi_menu_add_item()	235
5.37.3.10 gldi_menu_add_item_with_tooltip()	235
5.37.3.11 gldi_menu_add_sub_menu_full()	236
5.37.3.12 gldi_menu_add_separator()	236
5.38 cairo-dock-module-instance-manager.h File Reference	237
5.38.1 Detailed Description	237
5.38.2 Macro Definition Documentation	237
5.38.2.1 GLDI_OBJECT_IS_MODULE_INSTANCE	237
5.38.3 Function Documentation	238
5.38.3.1 gldi_module_instance_new()	238
5.39 cairo-dock-module-manager.h File Reference	238
5.39.1 Detailed Description	239
5.39.2 Macro Definition Documentation	239
5.39.2.1 GLDI_ABI_VERSION	239
5.39.2.2 GLDI_OBJECT_IS_MODULE	239
5.39.3 Function Documentation	239
5.39.3.1 gldi_module_new()	239
5.39.3.2 gldi_module_new_from_so_file()	240
5.39.3.3 gldi_modules_new_from_directory()	240
5.39.3.4 gldi_module_get_config_dir()	240
5.39.3.5 gldi_module_get()	241
5.39.3.6 gldi_modules_load_auto_config()	241
5.39.3.7 gldi_module_activate()	241
5.39.3.8 gldi_module_deactivate()	241
5.40 cairo-dock-object.h File Reference	243
5.40.1 Detailed Description	244
5.40.2 Macro Definition Documentation	244
5.40.2.1 gldi_object_notify	244
5.40.3 Enumeration Type Documentation	244
5.40.3.1 GldiObjectNotifications	244
5.40.4 Function Documentation	244
5.40.4.1 gldi_object_new()	244

5.40.4.2 gldi_object_ref()	245
5.40.4.3 gldi_object_unref()	245
5.40.4.4 gldi_object_delete()	245
5.40.4.5 gldi_object_reload()	246
5.40.4.6 gldi_object_register_notification()	246
5.40.4.7 gldi_object_remove_notification()	246
5.41 cairo-dock-opengl-font.h File Reference	247
5.41.1 Detailed Description	247
5.41.2 Function Documentation	247
5.41.2.1 cairo_dock_create_texture_from_text_simple()	247
5.41.2.2 cairo_dock_load_textured_font()	248
5.41.2.3 cairo_dock_load_textured_font_from_image()	248
5.41.2.4 cairo_dock_free_gl_font()	248
5.41.2.5 cairo_dock_get_gl_text_extent()	249
5.41.2.6 cairo_dock_draw_gl_text()	249
5.41.2.7 cairo_dock_draw_gl_text_at_position()	249
5.41.2.8 cairo_dock_draw_gl_text_in_area()	250
5.41.2.9 cairo_dock_draw_gl_text_at_position_in_area()	250
5.42 cairo-dock-opengl-path.h File Reference	251
5.42.1 Detailed Description	251
5.42.2 Function Documentation	251
5.42.2.1 cairo_dock_new_gl_path()	251
5.42.2.2 cairo_dock_free_gl_path()	252
5.42.2.3 cairo_dock_gl_path_move_to()	252
5.42.2.4 cairo_dock_gl_path_set_extent()	252
5.42.2.5 cairo_dock_gl_path_line_to()	253
5.42.2.6 cairo_dock_gl_path_rel_line_to()	253
5.42.2.7 cairo_dock_gl_path_curve_to()	253
5.42.2.8 cairo_dock_gl_path_rel_curve_to()	255
5.42.2.9 cairo_dock_gl_path_simple_curve_to()	255
5.42.2.10 cairo_dock_gl_path_rel_simple_curve_to()	256
5.42.2.11 cairo_dock_gl_path_arc()	256
5.42.2.12 cairo_dock_stroke_gl_path()	257
5.42.2.13 cairo_dock_fill_gl_path()	257
5.42.2.14 cairo_dock_draw_rounded_rectangle_opengl()	257
5.43 cairo-dock-opengl.h File Reference	258
5.43.1 Detailed Description	258
5.43.2 Macro Definition Documentation	258
5.43.2.1 gldi_gl_container_begin_draw	258
5.43.3 Function Documentation	259
5.43.3.1 gldi_gl_backend_init()	259
5.43.3.2 gldi_gl_init_opengl_context()	259

5.43.3.3 gldi_gl_container_make_current()	259
5.43.3.4 gldi_gl_offscreen_context_make_current()	260
5.43.3.5 gldi_gl_container_begin_draw_full()	260
5.43.3.6 gldi_gl_container_end_draw()	260
5.43.3.7 gldi_gl_container_set_perspective_view()	260
5.43.3.8 gldi_gl_container_set_perspective_view_for_icon()	261
5.43.3.9 gldi_gl_container_set_ortho_view()	261
5.43.3.10 gldi_gl_container_set_ortho_view_for_icon()	261
5.43.3.11 gldi_gl_container_init()	261
5.43.3.12 gldi_gl_container_resized()	262
5.44 cairo-dock-overlay.h File Reference	262
5.44.1 Detailed Description	263
5.44.2 Macro Definition Documentation	263
5.44.2.1 cairo_dock_set_overlay_scale	263
5.44.2.2 cairo_dock_get_overlay_image_buffer	263
5.44.3 Function Documentation	264
5.44.3.1 cairo_dock_add_overlay_from_image()	264
5.44.3.2 cairo_dock_add_overlay_from_surface()	264
5.44.3.3 cairo_dock_add_overlay_from_texture()	265
5.44.3.4 cairo_dock_remove_overlay_at_position()	265
5.44.3.5 cairo_dock_print_overlay_on_icon_from_image()	265
5.44.3.6 cairo_dock_print_overlay_on_icon_from_surface()	267
5.45 cairo-dock-packages.h File Reference	267
5.45.1 Detailed Description	268
5.45.2 Macro Definition Documentation	268
5.45.2.1 cairo_dock_get_url_data	268
5.45.3 Enumeration Type Documentation	269
5.45.3.1 CairoDockPackageType	269
5.45.4 Function Documentation	269
5.45.4.1 cairo_dock_download_file()	269
5.45.4.2 cairo_dock_download_file_in_tmp()	270
5.45.4.3 cairo_dock_download_archive()	270
5.45.4.4 cairo_dock_download_file_async()	270
5.45.4.5 cairo_dock_get_url_data_with_post()	271
5.45.4.6 cairo_dock_get_url_data_async()	271
5.45.4.7 cairo_dock_free_package()	272
5.45.4.8 cairo_dock_list_packages()	272
5.45.4.9 cairo_dock_list_packages_async()	272
5.45.4.10 cairo_dock_get_package_path()	273
5.46 cairo-dock-particle-system.h File Reference	273
5.46.1 Detailed Description	274
5.46.2 Macro Definition Documentation	274

5.46.2.1 cairo_dock_render_particles	274
5.46.3 Function Documentation	274
5.46.3.1 cairo_dock_render_particles_full()	274
5.46.3.2 cairo_dock_create_particle_system()	275
5.46.3.3 cairo_dock_free_particle_system()	275
5.46.3.4 cairo_dock_update_default_particle_system()	275
5.47 cairo-dock-separator-manager.h File Reference	276
5.47.1 Detailed Description	276
5.47.2 Macro Definition Documentation	276
5.47.2.1 GLDI_OBJECT_IS_SEPARATOR_ICON	276
5.48 cairo-dock-stack-icon-manager.h File Reference	276
5.48.1 Detailed Description	277
5.48.2 Macro Definition Documentation	277
5.48.2.1 GLDI_OBJECT_IS_STACK_ICON	277
5.49 cairo-dock-style-facility.h File Reference	277
5.49.1 Detailed Description	278
5.49.2 Function Documentation	278
5.49.2.1 gldi_style_color_shade()	278
5.50 cairo-dock-style-manager.h File Reference	278
5.50.1 Detailed Description	278
5.50.2 Macro Definition Documentation	279
5.50.2.1 gldi_style_colors_set_bg_color	279
5.50.3 Enumeration Type Documentation	279
5.50.3.1 GldiStyleNotifications	279
5.50.4 Function Documentation	279
5.50.4.1 gldi_style_color_get()	279
5.50.4.2 gldi_style_colors_set_bg_color_full()	279
5.50.4.3 gldi_style_colors_set_selected_bg_color()	280
5.50.4.4 gldi_style_colors_set_line_color()	280
5.50.4.5 gldi_style_colors_set_text_color()	280
5.50.4.6 gldi_style_colors_set_separator_color()	280
5.50.4.7 gldi_style_colors_set_child_color()	281
5.50.4.8 gldi_style_colors_paint_bg_color_with_alpha()	281
5.51 cairo-dock-surface-factory.h File Reference	281
5.51.1 Detailed Description	282
5.51.2 Macro Definition Documentation	283
5.51.2.1 cairo_dock_create_surface_for_square_icon	283
5.51.2.2 cairo_dock_create_surface_from_text	283
5.51.3 Enumeration Type Documentation	283
5.51.3.1 CairoDockLoadImageModifier	283
5.51.4 Function Documentation	284
5.51.4.1 cairo_dock_create_surface_from_xicon_buffer()	284

5.51.4.2	cairo_dock_create_surface_from_pixbuf()	284
5.51.4.3	cairo_dock_create_blank_surface_full()	285
5.51.4.4	cairo_dock_load_gdk_pixbuf()	285
5.51.4.5	cairo_dock_load_gdk_pixbuf_with_max_size()	286
5.51.4.6	cairo_dock_create_surface_from_image()	286
5.51.4.7	cairo_dock_create_surface_from_image_simple()	287
5.51.4.8	cairo_dock_create_surface_from_icon()	287
5.51.4.9	cairo_dock_create_surface_from_pattern()	288
5.51.4.10	cairo_dock_rotate_surface()	288
5.51.4.11	cairo_dock_create_surface_from_text_full()	289
5.51.4.12	cairo_dock_duplicate_surface()	289
5.52	cairo-dock-task.h File Reference	290
5.52.1	Detailed Description	291
5.52.2	Macro Definition Documentation	291
5.52.2.1	gldi_task_new	291
5.52.2.2	gldi_task_get_elapsed_time	292
5.52.3	Function Documentation	292
5.52.3.1	gldi_task_launch()	292
5.52.3.2	gldi_task_launch_delayed()	292
5.52.3.3	gldi_task_new_full()	292
5.52.3.4	gldi_task_stop()	293
5.52.3.5	gldi_task_discard()	293
5.52.3.6	gldi_task_free()	294
5.52.3.7	gldi_task_is_active()	294
5.52.3.8	gldi_task_is_running()	294
5.52.3.9	gldi_task_change_frequency()	294
5.52.3.10	gldi_task_change_frequency_and_relaunch()	296
5.52.3.11	gldi_task_downgrade_frequency()	296
5.52.3.12	gldi_task_set_normal_frequency()	296
5.53	cairo-dock-themes-manager.h File Reference	296
5.53.1	Detailed Description	297
5.53.2	Function Documentation	297
5.53.2.1	cairo_dock_update_conf_file()	297
5.53.2.2	cairo_dock_write_keys_to_conf_file()	297
5.53.2.3	cairo_dock_write_keys_to_new_conf_file()	298
5.53.2.4	cairo_dock_export_current_theme()	298
5.53.2.5	cairo_dock_package_current_theme()	298
5.53.2.6	cairo_dock_depackage_theme()	299
5.53.2.7	cairo_dock_delete_themes()	299
5.53.2.8	cairo_dock_import_theme()	299
5.53.2.9	cairo_dock_import_theme_async()	300
5.53.2.10	cairo_dock_set_paths()	300

5.54 cairo-dock-user-icon-manager.h File Reference	301
5.54.1 Detailed Description	301
5.54.2 Macro Definition Documentation	301
5.54.2.1 GLDI_OBJECT_IS_USER_ICON	301
5.55 cairo-dock-utils.h File Reference	301
5.55.1 Detailed Description	302
5.55.2 Enumeration Type Documentation	302
5.55.2.1 GldiLaunchFlags	302
5.55.3 Function Documentation	302
5.55.3.1 cairo_dock_remove_version_from_string()	302
5.55.3.2 cairo_dock_remove_html_spaces()	303
5.55.3.3 cairo_dock_get_version_from_string()	303
5.55.3.4 cairo_dock_string_is_address()	303
5.55.3.5 cairo_dock_launch_command_argv_full()	304
5.55.3.6 cairo_dock_launch_command_argv_full2()	304
5.55.3.7 cairo_dock_get_default_terminal()	305
5.56 cairo-dock-windows-manager.h File Reference	305
5.56.1 Detailed Description	305
5.56.2 Function Documentation	305
5.56.2.1 gldi_windows_manager_register_backend()	305
5.56.2.2 gldi_windows_foreach()	306
5.56.2.3 gldi_windows_find()	306
5.56.2.4 gldi_windows_get_active()	306
5.56.2.5 gldi_window_set_thumbnail_area()	307
5.56.2.6 gldi_window_get_menu_address()	307
5.56.2.7 gldi_window_manager_get_all()	307
5.56.2.8 gldi_window_manager_have_coordinates()	307
5.56.2.9 gldi_window_manager_can_track_workspaces()	308
5.56.2.10 gldi_window_manager_is_position_relative_to_current_viewport()	308
5.56.2.11 gldi_window_manager_can_move_to_desktop()	308
5.57 gldi-icon-names.h File Reference	308
5.57.1 Detailed Description	308
5.58 cairo-dock-cinnamon-integration.h File Reference	308
5.58.1 Detailed Description	308
5.59 cairo-dock-compiz-integration.h File Reference	309
5.59.1 Detailed Description	309
5.60 cairo-dock-default-view.h File Reference	309
5.60.1 Detailed Description	309
5.61 cairo-dock-gauge.h File Reference	309
5.61.1 Detailed Description	309
5.62 cairo-dock-gnome-shell-integration.h File Reference	309
5.62.1 Detailed Description	309

5.63 cairo-dock-graph.h File Reference	309
5.63.1 Detailed Description	310
5.63.2 Enumeration Type Documentation	310
5.63.2.1 CairoDockTypeGraph	310
5.64 cairo-dock-hiding-effect.h File Reference	310
5.64.1 Detailed Description	310
5.65 cairo-dock-icon-container.h File Reference	310
5.65.1 Detailed Description	310
5.66 cairo-dock-kwin-integration.h File Reference	311
5.66.1 Detailed Description	311
5.67 cairo-dock-progressbar.h File Reference	311
5.67.1 Detailed Description	311
5.68 cairo-dock-wayfire-integration.h File Reference	311
5.68.1 Detailed Description	311
Index	313

Chapter 1

Cairo-Dock's API documentation.

[Introduction](#)

[Installation](#)

[Main structures](#)

- [Objects](#)
- [Managers](#)
- [Containers](#)
- [Icons](#)
- [Dock](#)
- [Desklet](#)
- [Dialog](#)
- [Modules](#)
- [Module-Instances](#)
- [Drawing with cairo/opengl](#)
- [Windows management](#)

[External Modules](#)

- [Create a new applet](#)
- [First steps](#)
- [Go further](#)
- [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 ?](#)
- [Key binding](#)
- [I need more than one icon, how can I easily get more ?](#)

- [How do I provide translations \(internationalization\) for my applet ?](#)

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 ?](#)
- [Applets with advanced initialization steps or providing core functionality](#)
- [Auto-loaded applets](#)

1.1 Introduction

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

It is useful if you want to write a plug-in, add new features in the core, or just love C.

Note: to write applets in any language very easily, see <https://github.com/Cairo-Dock/cairo-dock-core/wiki/Writing-an-applet>.

It has a **decentralized conception** and is built of several modules: internal modules ([Managers](#)) and external modules ([Modules](#)) that can extend it.

It also has an [Objects](#) architecture.

1.2 Installation

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

```
### grab the sources of the core
mkdir CD && cd CD
bzip2 -dc cairo-dock-core.tar.gz
### compile 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 -dc cairo-dock-plug-ins.tar.gz
### compile the stable plug-ins and install them
cmake CMakeLists.txt -DCMAKE_INSTALL_PREFIX=/usr
make
sudo make install
```

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

1.3 Main structures

1.3.1 Objects

Any element in *libgldi* is a [_GldiObject](#).

An Object is created by an ObjectManager, which defines the properties and notifications of its children.

It has a reference counter, can be deleted from the current theme, and can be reloaded.

An Object can cast **notifications**; notifications are broadcasted on its ObjectManager.

An ObjectManager can inherit from another ObjectManager; in this case, all methods of the parent ObjectManagers are called recursively, and likewise all notifications on an Object are casted recursively to all parent ObjectManagers.

See [_GldiObject](#) and [cairo-dock-object.h](#) for more details.

1.3.2 Managers

The core is divided in several internal modules, called Managers.

Each Manager manages a set of parameters and objects (for instance, the Dock Manager manages the list of all Docks and their parameters).

See [_GldiManager](#) and [cairo-dock-manager.h](#) for more details.

1.3.3 Containers

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

See [_GldiContainer](#) and [cairo-dock-container.h](#) for more details.

1.3.4 Icons

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

See [_Icon](#) and [cairo-dock-icon-factory.h](#) for more details.

1.3.5 Dock

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

See [_CairoDock](#) and [cairo-dock-dock-factory.h](#) for more details.

1.3.6 Desklet

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

See [_CairoDesklet](#) and [cairo-dock-desklet-factory.h](#) for more details.

1.3.7 Dialog

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

See [_CairoDialog](#) and [cairo-dock-dialog-factory.h](#) for more details.

1.3.8 Modules

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

See [_GldiModule](#) and [cairo-dock-module-manager.h](#) for more details.

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

1.3.9 Module-Instances

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

See [_GldiModuleInstance](#) and [cairo-dock-module-instance-manager.h](#) for more details.

1.3.10 Drawing with cairo/opengl

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

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

1.3.11 Windows management

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

1.3.12 Display scaling

Cairo-Dock now supports integer display scale factors > 1 (fractional scaling is not supported and will likely result in a higher integer scale factor used). Display scale is determined by querying it from GDK using `gdk_window_get_scale_factor ()` of the window to be rendered, or the primary dock (`g_pMainDock`).

Most of the code is independent of the scale factor and uses logical pixels according to how rendering in GTK works. Specifically, the Cairo rendering code does this entirely, as the context supplied by GTK has the proper scale factor set. We need to consider the scale factor in the following cases though:

- **OpenGL rendering:** Contrary to Cairo, we need to manually set up a "scaling" before rendering. This is done by calling `glScalef ()` with the correct scale factor before rendering (so that the rest of the code can work with logical coordinates) and by giving the physical pixel size of surfaces when setting up our viewports (`gldi_gl_container_set_ortho_view ()` and `gldi_gl_container_set_perspective_view ()` in `cairo-dock-opengl.c`).
- **EGL:** when using EGL with Wayland, we need to give the correct pixel size when creating surfaces corresponding to our windows; also, we need to explicitly set the surface scale in some cases (`cairo-dock-egl.c`)
- **Creating surfaces:** when creating Cairo surfaces and OpenGL textures that we draw to, we need to ensure that these have the correct pixel size (i.e. logical size multiplied by the scale factor). For Cairo surfaces, we need to set a scale factor as well with `cairo_surface_set_device_scale ()`. This is done in `cairo-dock-surface-factory.c`, using the scale factor from `g_pMainDock` for now (but could be extended to take the scale factor as a parameter to handle situations when rendering to multiple displays with different scale factors).
- **Loading images:** most importantly, when loading any image (e.g. icons corresponding to windows in the taskbar / applets), we need to ensure that it gets loaded with the correct size, so that it will be rendered without being blurry. This is supported by a combination of multiple components: (1) The `cairo_dock_search_icon_s_path ()` function is used to find the actual image files to load in most cases. This function automatically uses the scale factor of `g_pMainDock` when searching images (with `gtk_icon_theme_lookup_icon_for_scale ()`) to find image files with the correct resolution. (2) Icons to be displayed in the dock are then loaded in their natural size, which should then be large enough to be rendered correctly. (3) In some cases, we load an image at a specified size; we take care to supply the scaled size here. (4) SVG images are loaded by "rendering" them to a Cairo surface with the correct pixel size and scale. E.g. if you just need an icon at a specified size, use `cairo_dock_create_surface_from_icon ()` that will ensure that the correct scale is used.
- **Tracking window locations on X11:** when communicating directly with the X server, we get unscaled coordinates (that correspond to physical pixels on the screen). This is relevant for tracking window locations, checking if they overlap with the dock or sending windows to specific viewport / display. We handle these by dividing the coordinates by the scale factor got from GDK, so that we can work in logical coordinates here as well. Note: this assumes all monitors have the same scale which seems to be the case always for X11.

1.4 External Modules

1.4.1 Create a new applet

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

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

Note: when adding a new file, don't forget to add it in the `CMakeLists.txt`.

when changing something in the config file, don't forget to update the version number of the applet, in the main `CMakeLists.txt`.

when changing anything, don't forget to install (*sudo make install*)

1.4.2 First steps

Edit the file `src/applet-inic.c`; the macro `CD_APPLET_DEFINITION2` is a convenient way to define an applet: just fill its name, its category, a brief description, and your name.

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

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

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

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

Edit the file `src/applet-config.c`; In the section `CD_APPLET_GET_CONFIG_BEGIN/CD_APPLET_GET_CONFIG↔_END`, get all your config parameters (don't forget to define them in `applet-struct.h`). Use the `CD_CONFIG_GET_*` macros (defined in [cairo-dock-applet-facility.h](#)) to do so conveniently.

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

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

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

There are other similar sections available:

- `CD_APPLET_ON_MIDDLE_CLICK_BEGIN/ CD_APPLET_ON_MIDDLE_CLICK_END` for the actions on middle click on your icon or one of its sub-dock.

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

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

- `CD_APPLET_REGISTER_FOR_CLICK_EVENT`
- `CD_APPLET_REGISTER_FOR_MIDDLE_CLICK_EVENT`
- `CD_APPLET_REGISTER_FOR_DOUBLE_CLICK_EVENT`
- `CD_APPLET_REGISTER_FOR_SCROLL_EVENT`
- `CD_APPLET_REGISTER_FOR_BUILD_MENU_EVENT`

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

1.4.3 Go further

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

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

- **myIcon** : this is your icon !
- **myContainer** : the container your icon belongs to (a Dock or a Desklet). For convenience, the following 2 parameters are available.
- **myDock** : if your container is a dock, `myDock = myContainer`, otherwise it is `NULL`.
- **myDesklet** : if your container is a desklet, `myDesklet = myContainer`, otherwise it is `NULL`.
- **myConfig** : the structure holding all the parameters you get in your config file. You have to define it in *applet-struct.h*.
- **myData** : the structure holding all the ressources loaded at run-time. You have to define it in *applet-struct.h*.
- **myDrawContext** : a cairo context, if you need to draw on the icon with the libcairo.
- To get values contained inside your **conf file**, you can use the following :
`CD_CONFIG_GET_BOOLEAN` & cie
- To **build your menu**, you can use the following :
`CD_APPLET_ADD_SUB_MENU` & cie
- To directly **set an image on your icon**, you can use the following :
`CD_APPLET_SET_IMAGE_ON_MY_ICON` & cie
- To modify the **label** of your icon, you can use the following :
`CD_APPLET_SET_NAME_FOR_MY_ICON` & cie
- To set a **quick-info** on your icon, you can use the following :
`CD_APPLET_SET_QUICK_INFO_ON_MY_ICON` & cie
- To **create a surface** that fits your icon from an image, you can use the following :
`CD_APPLET_LOAD_SURFACE_FOR_MY_APPLET` & cie
- To trigger the **refresh** of your icon or container after you drew something, you can use the following :
`CD_APPLET_REDRAW_MY_ICON` & `CAIRO_DOCK_REDRAW_MY_CONTAINER`

1.4.4 How can I take advantage of the OpenGL ?

There are 3 cases :

- your applet just has a static icon; there is nothing to take into account, the common functions to set an image or a surface on an icon already handle the texture mapping.
- you draw dynamically on your icon with libcairo (using `myDrawContext`), but you don't want to bother with OpenGL; all you have to do is to call `/ref cairo_dock_update_icon_texture` to update your icon's texture after you drawn your surface. This can be done for occasional drawings, like *Switcher* redrawing its icon each time a window is moved.
- you draw your icon differently whether the dock is in OpenGL mode or not; in this case, you just need to put all the OpenGL commands into a `CD_APPLET_START_DRAWING_MY_ICON/CD_APPLET_FINISH_DRAWING_MY_ICON` section inside your code. If your applet relies on OpenGL for its core function, such that it does not make sense to use it without (e.g. it adds OpenGL effects to the dock or icons etc.), add `CAIRO_DOCK_MODULE_REQUIRES_OPENGL` to the flags (second argument) in `CD_APPLET_DEFINITION2`. This will result in the applet not loaded at all if OpenGL is not available.

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

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

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

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

First you will register to the "update container" notification, with a simple call to `CD_APPLET_REGISTER_FOR_UPDATE_ICON_SLOW` or `CD_APPLET_REGISTER_FOR_UPDATE_ICON_EVENT`, depending on the refresh frequency you need : $\sim 10\text{Hz}$ or $\sim 33\text{Hz}$. 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`.

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

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

You create a Task with `cairo_dock_new_task`, launch it with `cairo_dock_launch_task`, and either cancel it with `cairo_dock_discard_task` or destroy it with `cairo_dock_free_task`.

1.4.7 Key binding

You can bind an action to a shortcut with the following macro: `CD_APPLET_BIND_KEY`.

For instance, the GMenu applet displays the menu on `ctrl+F1`.

You get a `GldiShortkey` that you simply destroy when the applet stops (with [gldi_object_unref](#)).

See [cairo-dock-keybinder.h](#) for more details.

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

In dock mode, your icon can have a sub-dock; in desklet mode, you can load a list of icons into your desklet. Cairo-Dock provides a convenient macro to **quickly load a list of icons** in both cases ↔ : `CD_APPLET_LOAD_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`.

1.4.9 How do I provide translations (internationalization) for my applet ?

Cairo-Dock uses the standard gettext library to provide translated strings in its user interface. Specifically, the `dgettext()` function is used when creating menus, the configuration interface, etc. For plugins that are part of the official plugins package, the "cairo-dock-plugins" message domain is used and this is set by default when loading a plugin. For external plugins, it is recommended to define your own message domain and install translations accordingly (after compiling them with `msgfmt`). Within the plugin code, use `dgettext()` when displaying text to the user. To ensure that translation of configuration options work, you should provide the message domain also when your plugin is loaded. This is achieved by using the `CD_APPLET_DEFINE2_BEGIN` macro (see [Applets with advanced initialization steps or providing core functionality](#)) and setting `pVisitCard->cGettextDomain` to the message domain of your plugin. You should also call the `bindtextdomain()` function at this point to supply where translations are installed.

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_DEFINE2_BEGIN/CD_APPLET_DEFINE2_END` section (see below), 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_gui_find_group_key_widget_in_list](#). To add your custom widgets, insert in the conf file an empty widget (with the prefix `'_'`), then grab it and pack some `GtkWidget` inside. If you want to dynamically alter the config panel (like having a "new" button that would make appear new widgets on click), you can add in the conf file the new widgets, and then call [cairo_dock_reload_current_module_widget](#) to reload the config panel. See the `AlsaMixer` or `Weather` applets for an easy example, and `Clock` or `Mail` for a more advanced example.

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

Say your applet launches an external application that has its own window. It is logical to **make your applet control this application**, rather than letting the Taskbar do. All you need to do is to call the macro [CD_APPLET_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 appli icon.

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

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

Each kind of renderer has a set of attributes that you can use to customize it; you just need to call the [CD_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.

1.5.6 Applets with advanced initialization steps or providing core functionality

Some applets need more control over their life-cycle. Beyond applets that define their own message domain for internationalization (see [How do I provide translations \(internationalization\) for my applet ?](#)), this typically relates to applets that provide some essential functionality for the dock (e.g. renderers, desktop environment integration, etc.). In this case, instead of defining the applet with `CD_APPLET_DEFINITION2`, use the macro pair `CD_APPLET_DEFINE2_BEGIN / CD_APPLET_DEFINE2_END` and

- Add early initialization steps between these; this will be run when the applet is first opened (loaded from disk), regardless whether it is enabled yet. Be careful that the applet's instance variable (`myApplet`) and configuration (`myConfig`) are not available at this point (so you will need to use static variables to save any state). Also, no docks, cairo or OpenGL contexts exist, and the current theme has not been loaded at this point as well. It is thus recommended to keep things here absolute minimal.
- Also you have to manually define the applet's interface here (load, stop, config functions, etc.); see the `GldiModuleInterface` struct from [cairo-dock-module-manager.h](#) for a description of each function and what it does, or use the `CD_APPLET_DEFINE_COMMON_APPLET_INTERFACE` macro that defines these with the usual function names thus you can use the same macros as you would with `CD_APPLET_DEFINITION2`.

1.5.7 Auto-loaded applets

A special class of applets is considered "auto-loaded": this means that the applet is always enabled and cannot be disabled by the user. Note that the `init()` and `stop()` functions are still called as normal whenever Cairo-Dock reloads its full configuration (i.e. when changing the current theme). However, there are slight differences:

- The configuration of auto-loaded applets is read (`CD_APPLET_GET_CONFIG_BEGIN / read_conf_file ()`) earlier than for normal applets. Specifically, this happens before creating any of the docks, but after Cairo-Dock's core configuration has been loaded. This means that such applets should not rely on the existence of any docks, or their drawing context when reading their configuration (this is not recommended anyway for normal applets). On the other hand, such applets can provide parameters when creating docks (e.g. dock-rendering is an example of this).
- Initialization of such applets (`CD_APPLET_INIT_BEGIN / initModule ()`) happens earlier than for normal applets. Specifically, this is done after the main dock has been created and core functionality has started, but before adding any launchers (normal applets' `init` happens after loading launchers).
- Auto-loaded applets are stopped after regular ones.

An applet is defined as auto-loaded if its type is `CAIRO_DOCK_MODULE_IS_PLUGIN` and at least one of the following is true:

- it does not have an `initModule ()` function
- it does not have a `stopModule ()` function
- it extends a core functionality ("manager") of Cairo-Dock: this is achieved by using the `CD_APPLET_EXTEND_MANAGER` macro when the module is read

Note: all of the above need to be defined by using `CD_APPLET_DEFINE2_BEGIN / CD_APPLET_DEFINE2_END`

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

_CairoDataRenderer	Generic DataRenderer. Any implementation of a DataRenderer will derive from this class . . .	19
_CairoDataRendererAttribute	Generic DataRenderer attributes structure. The attributes of any implementation of a DataRenderer will derive from this class	20
_CairoDataRendererInterface	Interface of a DataRenderer	21
_CairoDesklet	Definition of a Desklet, which derives from a Container	22
_CairoDeskletAttr	Configuration attributes of a Desklet	22
_CairoDeskletDecoration	Decoration of a Desklet	22
_CairoDeskletRenderer	Definition of a Desklet's renderer	23
_CairoDialog	Definition of a Dialog	23
_CairoDialogDecorator	Definition of a Dialog/Menu decorator. It draws the frame of the Dialog/Menu	24
_CairoDialogRenderer	Definition of a Dialog renderer. It draws the inside of the Dialog	24
_CairoDock	Definition of a Dock, which derives from a Container	25
_CairoDockDesktopEnvBackend	Definition of the Desktop Environment backend	28
_CairoDockGLConfig	This structure summarizes the available OpenGL configuration on the system	28
_CairoDockGLFont	Structure used to load a font for OpenGL text rendering	28
_CairoDockGLPath	Definition of a CairoDockGLPath	29
_CairoDockGroupKeyWidget	Definition of a widget corresponding to a given (group;key) pair	29
_CairoDockGuiBackend	Definition of the GUI interface for modules	29

_CairoDockHidingEffect	Definition of a Hiding Effect backend (used to provide an animation when the docks hides/shows itself)	30
_CairoDockImageBuffer	Definition of an Image Buffer. It provides an unified interface for a cairo/opengl image buffer . .	30
_CairoDockPackage	Definition of a generic package	31
_CairoDockRenderer	Dock's renderer, also known as 'view'	32
_CairoDockTransition	Transitions are an easy way to set an animation on an Icon to make it change from a state to another	32
_CairoGraphAttribute	Attributes of a Graph	33
_CairoIconContainerRenderer	Definition of an Icon container (= an icon holding a sub-dock) renderer	34
_CairoOverlay	Definition of an Icon Overlay	34
_CairoParticle	A particle of a particle system	35
_CairoParticleSystem	A particle system	36
_CairoProgressBarAttribute	Attributes of a PgrogressBar	36
_GldiChildProcessManagerBackend		37
_GldiContainer	Definition of a Container, whom derive Dock, Desklet, Dialog and FlyingContainer	38
_GldiContainerManagerBackend	Definition of the Container backend. It defines some operations that should be, but are not, provided by GTK	39
_GldiDesktopBackground	Definition of a Desktop Background Buffer. It has a reference count so that it can be shared across all the lib	41
_GldiDesktopManagerBackend	Definition of the Desktop Manager backend	41
_GldiManager	Definition of a Manager	41
_GldiModule	Definition of an external module	42
_GldiModuleInstance	Definition of an instance of a module. A module can be instanciated several times	42
_GldiModuleInterface	Definition of the interface of a module	43
_GldiObject	Definition of an Object	45
_GldiObjectManager	Definition of an ObjectManager	45
_GldiTask	Definition of a periodic and/or asynchronous Task	46
_GldiTextDescription	Description of the rendering of a text	46
_GldiVisitCard	Definition of the visit card of a module. Contains everything that is statically defined for a module	47
_GldiWindowActor	Definition of a window actor	48
_GldiWindowManagerBackend	Definition of the Windows Manager backend	48

_Icon	
Definition of an Icon	48
_IconInterface	
Icon's interface	49
GldiAppInfo	50

Chapter 3

File Index

3.1 File List

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

cairo-dock-animations.h	51
cairo-dock-applet-canvas.h	58
cairo-dock-applet-facility.h	66
cairo-dock-applet-manager.h	98
cairo-dock-applications-manager.h	98
cairo-dock-class-manager.h	101
cairo-dock-config.h	107
cairo-dock-container.h	108
cairo-dock-core.h	118
cairo-dock-data-renderer-manager.h	118
cairo-dock-data-renderer.h	119
cairo-dock-dbus.h	128
cairo-dock-desklet-factory.h	133
cairo-dock-desklet-manager.h	138
cairo-dock-desktop-file-db.h	140
cairo-dock-desktop-manager.h	142
cairo-dock-dialog-factory.h	147
cairo-dock-dialog-manager.h	155
cairo-dock-dock-facility.h	157
cairo-dock-dock-factory.h	162
cairo-dock-dock-manager.h	165
cairo-dock-dock-visibility.h	170
cairo-dock-draw-opengl.h	170
cairo-dock-draw.h	177
cairo-dock-file-manager.h	180
cairo-dock-gui-factory.h	188
cairo-dock-gui-manager.h	193
cairo-dock-icon-facility.h	195
cairo-dock-icon-factory.h	209
cairo-dock-icon-manager.h	214
cairo-dock-image-buffer.h	216
cairo-dock-indicator-manager.h	222
cairo-dock-keybinder.h	222
cairo-dock-keyfile-utilities.h	224
cairo-dock-launcher-manager.h	228

cairo-dock-manager.h	230
cairo-dock-menu.h	230
cairo-dock-module-instance-manager.h	237
cairo-dock-module-manager.h	238
cairo-dock-object.h	243
cairo-dock-opengl-font.h	247
cairo-dock-opengl-path.h	251
cairo-dock-opengl.h	258
cairo-dock-overlay.h	262
cairo-dock-packages.h	267
cairo-dock-particle-system.h	273
cairo-dock-separator-manager.h	276
cairo-dock-stack-icon-manager.h	276
cairo-dock-style-facility.h	277
cairo-dock-style-manager.h	278
cairo-dock-surface-factory.h	281
cairo-dock-task.h	290
cairo-dock-themes-manager.h	296
cairo-dock-user-icon-manager.h	301
cairo-dock-utils.h	301
cairo-dock-windows-manager.h	305
gldi-icon-names.h	308
cairo-dock-cinnamon-integration.h	308
cairo-dock-compiz-integration.h	309
cairo-dock-default-view.h	309
cairo-dock-gauge.h	309
cairo-dock-gnome-shell-integration.h	309
cairo-dock-graph.h	309
cairo-dock-hiding-effect.h	310
cairo-dock-icon-container.h	310
cairo-dock-kwin-integration.h	311
cairo-dock-progressbar.h	311
cairo-dock-wayfire-integration.h	311

Chapter 4

Data Structure Documentation

4.1 `_CairoDataRenderer` Struct Reference

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

```
#include <cairo-dock-data-renderer.h>
```

Data Fields

- [CairoDataRendererInterface](#) **interface**
interface of the Data Renderer.
- CairoDataToRenderer **data**
internal data to be drawn by the renderer.
- gint **iWidth**
size of the drawing area.
- [CairoDataRendererFormatValueFunc](#) **format_value**
specific function to format the values as text.
- gchar **cFormatBuffer** [CAIRO_DOCK_DATA_FORMAT_MAX_LEN+1]
buffer for the text.
- gpointer **pFormatData**
data passed to the format fonction.
- gboolean **bUpdateMinMax**
TRUE <=> the Data Renderer should dynamically update the range of the values.
- gboolean **bWriteValues**
TRUE <=> the Data Renderer should write the values as text itself.
- gint **iLatencyTime**
the time it will take to update to the new value, with a smooth animation (require openGL capacity)
- gint **iRank**
the rank of the renderer, eg the number of values it can display at once (for exemple, 1 for a bar, 2 for a dual-gauge)
- gboolean **bCanRenderValueAsText**
set to TRUE <=> the renderer can draw the values as text itself.
- gboolean **bRotateWithContainer**
set to TRUE <=> the drawing will be rotated if the container is vertical.
- `RendererRotateTheme` **iRotateTheme**
an option to rotate applet, no, automatic or always.

- gboolean **bisRotate**
set to TRUE <=> the theme images are rotated 90° clockwise.
- gboolean **bUseOverlay**
whether the data-renderer draws on an overlay rather than directly on the icon.
- [CairoOverlayPosition](#) **iOverlayPosition**
position of the overlay, in the case the renderer uses one.
- CairoDataRendererText * **pLabels**
an optionnal list of labels to be displayed on the Data Renderer to indicate the nature of each value. Same size as the set of values.
- CairoDataRendererEmblem * **pEmblems**
an optionnal list of emblems to be displayed on the Data Renderer to indicate the nature of each value. Same size as the set of values.
- CairoDataRendererTextParam * **pValuesText**
an optionnal list of text zones to write the values. Same size as the set of values.
- gint **iSmoothAnimationStep**
the animation counter for the smooth movement.
- gdouble **fLatency**
latency due to the smooth movement (0 means the displayed value is the current one, 1 the previous)

4.1.1 Detailed Description

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

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

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

4.2 _CairoDataRendererAttribute Struct Reference

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

```
#include <cairo-dock-data-renderer.h>
```

Data Fields

- const gchar * **cModelName**
name of the model ("gauge", "graph", etc) [mandatory].
- gint **iNbValues**
number of values to represent (for instance 3 for (cpu, mem, swap)) [1 by default and minimum].
- gint **iMemorySize**
number of values to remember over time. For instance graphs can display as much values as the icon's width [2 by default and minimum].
- gdouble * **pMinMaxValues**
an array of pairs of (min,max) values. [optionnal, input values will be considered between 0 and 1 if NULL].
- gboolean **bUpdateMinMax**
whether to automatically update the values' range [false by default].
- gboolean **bWriteValues**

- whether to write the values on the icon. [false by default].*
- **RendererRotateTheme iRotateTheme**
an option to rotate applet, no, automatic or always.
- **gint iLatencyTime**
time needed to update to the new values. The update is smooth in OpenGL mode. [0 by default]
- **CairoDataRendererFormatValueFunc format_value**
a function used to format the values into a string. Only useful if you make te DataRenderer write the values [optionnal, by default the values are formatted with 2 decimals].
- **gpointer pFormatData**
data to be passed to the format function [optionnal].
- **gchar ** cEmblems**
an optionnal list of emblems to draw on the overlay.
- **gchar ** cLabels**
an optionnal list of labels to write on the overlay.

4.2.1 Detailed Description

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

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

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

4.3 _CairoDataRendererInterface Struct Reference

Interface of a DataRenderer.

```
#include <cairo-dock-data-renderer.h>
```

Data Fields

- **CairoDataRendererLoadFunc load**
function that loads anything the DataRenderer will need. It also completes the DataRenderer structure (for instance the text zones).
- **CairoDataRendererRenderFunc render**
function that draws the values with cairo.
- **CairoDataRendererRenderOpenGLFunc render_opengl**
function that draws the values with opengl.
- **CairoDataRendererReloadFunc reload**
function that reloads the DataRenderer's buffers when the icon is resized.
- **CairoDataRendererUnloadFunc unload**
function that unload all the previously allocated buffers.

4.3.1 Detailed Description

Interface of a DataRenderer.

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

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

4.4 `_CairoDesklet` Struct Reference

Definition of a Desklet, which derives from a Container.

```
#include <cairo-dock-desklet-factory.h>
```

4.4.1 Detailed Description

Definition of a Desklet, which derives from a Container.

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

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

4.5 `_CairoDeskletAttr` Struct Reference

Configuration attributes of a Desklet.

```
#include <cairo-dock-desklet-factory.h>
```

4.5.1 Detailed Description

Configuration attributes of a Desklet.

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

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

4.6 `_CairoDeskletDecoration` Struct Reference

Decoration of a Desklet.

```
#include <cairo-dock-desklet-factory.h>
```

4.6.1 Detailed Description

Decoration of a Desklet.

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

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

4.7 _CairoDeskletRenderer Struct Reference

Definition of a Desklet's renderer.

```
#include <cairo-dock-desklet-factory.h>
```

Data Fields

- CairoDeskletRenderFunc **render**
rendering function with libcairo.
- CairoDeskletGLRenderFunc **render_opengl**
rendering function with OpenGL.
- CairoDeskletConfigureRendererFunc **configure**
get the configuration of the renderer from a set of config attributes.
- CairoDeskletLoadRendererDataFunc **load_data**
load the internal data of the renderer.
- CairoDeskletFreeRendererDataFunc **free_data**
free all internal data of the renderer.
- CairoDeskletCalculateIconsFunc **calculate_icons**
define the icons' size and load them.
- CairoDeskletUpdateRendererDataFunc **update**
function called on each iteration of the rendering loop.
- CairoDeskletGLRenderFunc **render_bounding_box**
optionnal rendering function with OpenGL that only draws the bounding boxes of the icons (for picking).
- GList * **pPreDefinedConfigList**
An optionnal list of preset configs.

4.7.1 Detailed Description

Definition of a Desklet's renderer.

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

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

4.8 _CairoDialog Struct Reference

Definition of a Dialog.

```
#include <cairo-dock-dialog-factory.h>
```

Data Fields

- [GldiContainer](#) **container**
container.

4.8.1 Detailed Description

Definition of a Dialog.

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

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

4.9 _CairoDialogDecorator Struct Reference

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

```
#include <cairo-dock-dialog-factory.h>
```

Data Fields

- CairoDialogSetDecorationSizeFunc **set_size**
defines the various margins and alignment of the dialog
- CairoDialogRenderDecorationFunc **render**
draw the dialog's frame (outline and background)
- CairoMenuSetupFunc **setup_menu**
defines the GldiMenuParams of the menu (radius, alignment, arrow height)
- CairoMenuRenderFunc **render_menu**
draw the menu's frame (outline and background); in the end, must clip the shape of the frame on the context
- const gchar * **cDisplayedName**
readable name of the decorator

4.9.1 Detailed Description

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

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

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

4.10 _CairoDialogRenderer Struct Reference

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

```
#include <cairo-dock-dialog-factory.h>
```

4.10.1 Detailed Description

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

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

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

4.11 _CairoDock Struct Reference

Definition of a Dock, which derives from a Container.

```
#include <cairo-dock-dock-factory.h>
```

Data Fields

- [GldiContainer](#) **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).
- [gchar](#) * **cDockName**
unique name of the dock
- [CairoDockVisibility](#) **iVisibility**
visibility.
- [gint](#) **iNumScreen**
- [gint](#) **iScreenReq**
number of the screen requested by the user
- [gchar](#) * **cScreenReq**
name of the screen requested by the user – TODO: not used yet, should implement saving screen names !
- [gint](#) **ilconSize**
icon size, as specified in the config of the dock
- [gboolean](#) **bGlobalIconSize**
whether the dock should use the global icons size parameters.
- [gboolean](#) **bGlobalBg**
whether the dock should use the global background parameters.
- [gchar](#) * **cBgImagePath**
path to an image, or NULL
- [gboolean](#) **bBgImageRepeat**
whether to repeat the image as a pattern, or to stretch it to fill the dock.
- [GldiColor](#) **fBgColorBright**
first color of the gradation
- [GldiColor](#) **fBgColorDark**
second color of the gradation
- [CairoDockImageBuffer](#) **backgroundBuffer**
Background image buffer of the dock.

- gdouble **fFoldingFactor**
(un)folding factor, between 0(unfolded) to 1(folded). It's up to the renderer on how to make use of it.
- gdouble **fHideOffset**
counter for auto-hide.
- gdouble **fPostHideOffset**
counter for the post-hiding animation for icons always visible.
- gboolean **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).
- gdouble **iMaxIconHeight**
maximum height of the icons.
- gdouble **fFlatDockWidth**
width of the dock, only taking into account an alignment of the icons.
- guint **iSidMoveResize**
Source ID for window resizing.
- guint **iSidUnhideDelayed**
Source ID for window popping down to the bottom layer.
- guint **iSidLeaveDemand**
Source ID of the timer that delays the "leave" event.
- guint **iSidUpdateWMIcons**
Source ID for pending update of WM icons geometry.
- guint **iSidHideBack**
Source ID for hiding back the dock.
- guint **iSidLoadBg**
Source ID for loading the background.
- guint **iSidDestroyEmptyDock**
Source ID to destroy an empty main dock.
- guint **iSidTestMouseOutside**
Source ID for shrinking down the dock after a mouse event.
- guint **iSidUpdateDockSize**
Source ID for updating the dock's size and icons layout.
- [CairoDockRenderer](#) * **pRenderer**

- current renderer, never NULL.*
- gpointer **pRendererData**
data that can be used by the renderer.
- gboolean **bCanDrop**
Set to TRUE by the renderer if one can drop between 2 icons.
- CairoDockMousePositionType **iMousePositionType**
set by the view to say if the mouse is currently on icons, on the 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).
- cairo_region_t * **pShapeBitmap**
input shape of the window when the dock is at rest.
- cairo_region_t * **pHiddenShapeBitmap**
input shape of the window when the dock is hidden.
- cairo_region_t * **pActiveShapeBitmap**
input shape of the window when the dock is active (NULL to cover all dock).
- gint **iScaleFactor**
last buffer scale factor set by us
- gpointer **pVisibilityData**
any data necessary for the dock visibility backend to work – managed by the backend

4.11.1 Detailed Description

Definition of a Dock, which derives from a Container.

4.11.2 Field Documentation

4.11.2.1 iNumScreen

```
gint _CairoDock::iNumScreen
```

number of the screen the dock is actually placed on (between 0 and g_desktopGeometry.iNbScreens - 1) might differ from iScreenReq if the requested screen is not available; filled out when updating the dock's position

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

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

4.12 **_CairoDockDesktopEnvBackend Struct Reference**

Definition of the Desktop Environment backend.

```
#include <cairo-dock-file-manager.h>
```

4.12.1 Detailed Description

Definition of the Desktop Environment backend.

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

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

4.13 **_CairoDockGLConfig Struct Reference**

This strucure summarizes the available OpenGL configuration on the system.

```
#include <cairo-dock-opengl.h>
```

4.13.1 Detailed Description

This strucure summarizes the available OpenGL configuration on the system.

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

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

4.14 **_CairoDockGLFont Struct Reference**

Structure used to load a font for OpenGL text rendering.

```
#include <cairo-dock-opengl-font.h>
```

4.14.1 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](#)

4.15 _CairoDockGLPath Struct Reference

Definition of a CairoDockGLPath.

```
#include <cairo-dock-opengl-path.h>
```

4.15.1 Detailed Description

Definition of a CairoDockGLPath.

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

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

4.16 _CairoDockGroupKeyWidget Struct Reference

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

```
#include <cairo-dock-gui-factory.h>
```

4.16.1 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](#)

4.17 _CairoDockGuiBackend Struct Reference

Definition of the GUI interface for modules.

```
#include <cairo-dock-gui-manager.h>
```

Data Fields

- void(* **set_status_message_on_gui**)(const gchar *cMessage)
display a message on the GUI.
- void(* **reload_current_widget**)(GldiModuleInstance *pModuleInstance, int iShowPage)
Reload the current config window from the conf file. iShowPage is the page that should be displayed in case the module has several pages, -1 means to keep the current page.
- CairoDockGroupKeyWidget *(* **get_widget_from_name**)(GldiModuleInstance *pModuleInstance, const gchar *cGroupName, const gchar *cKeyName)
retrieve the widgets in the current module window, corresponding to the (group,key) pair in its conf file.

4.17.1 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](#)

4.18 `_CairoDockHidingEffect` Struct Reference

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

```
#include <cairo-dock-animations.h>
```

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.

4.18.1 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](#)

4.19 `_CairoDockImageBuffer` Struct Reference

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

```
#include <cairo-dock-image-buffer.h>
```

4.19.1 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](#)

4.20 _CairoDockPackage Struct Reference

Definition of a generic package.

```
#include <cairo-dock-packages.h>
```

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.

4.20.1 Detailed Description

Definition of a generic package.

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

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

4.21 _CairoDockRenderer Struct Reference

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

```
#include <cairo-dock-dock-factory.h>
```

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.

4.21.1 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](#)

4.22 _CairoDockTransition Struct Reference

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

```
#include <cairo-dock-animations.h>
```

Data Fields

- [CairoDockTransitionRenderFunc](#) **render**
the cairo rendering function.
- [CairoDockTransitionGLRenderFunc](#) **render_opengl**
the openGL rendering function (can be NULL, in which case the texture mapping from the cairo drawing is done automatically).
- gpointer **pUserData**
data passed to the rendering functions.
- GFreeFunc **pFreeUserDataFunc**
function called to destroy the data when the transition is deleted.
- gboolean **bFastPace**
TRUE <=> the transition will be in the fast loop (high frequency refresh).
- gboolean **bRemoveWhenFinished**
TRUE <=> the transition will be destroyed and removed from the icon when finished.
- gint **iDuration**
duration if the transition, in ms. Can be 0 for an endless transition.
- gint **iElapsedTime**
elapsed time since the beginning of the transition, in ms.
- gint **iCount**
number of setps since the beginning of the transition, in ms.
- [GldiContainer](#) * **pContainer**
Container of the Icon.

4.22.1 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](#)

4.23 _CairoGraphAttribute Struct Reference

Attributes of a Graph.

```
#include <cairo-dock-graph.h>
```

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.

4.23.1 Detailed Description

Attributes of a Graph.

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

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

4.24 _CairoIconContainerRenderer Struct Reference

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

```
#include <cairo-dock-icon-factory.h>
```

4.24.1 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](#)

4.25 _CairoOverlay Struct Reference

Definition of an Icon Overlay.

```
#include <cairo-dock-overlay.h>
```

Data Fields

- [GldiObject](#) **object**
object
- [CairoDockImageBuffer](#) **image**
image buffer
- [CairoOverlayPosition](#) **iPosition**
position on the icon
- `gdouble` **fScale**
scale at which to draw the overlay, relatively to the icon (0.5 by default, 1 will cover the whole icon, 0 means to draw at the actual buffer size).
- [Icon](#) * **pIcon**
icon it belongs to.
- `gpointer` **data**
data used to identify an overlay

4.25.1 Detailed Description

Definition of an Icon Overlay.

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

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

4.26 _CairoParticle Struct Reference

A particle of a particle system.

```
#include <cairo-dock-particle-system.h>
```

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.

4.26.1 Detailed Description

A particle of a particle system.

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

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

4.27 _CairoParticleSystem Struct Reference

A particle system.

```
#include <cairo-dock-particle-system.h>
```

4.27.1 Detailed Description

A particle system.

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

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

4.28 _CairoProgressBarAttribute Struct Reference

Attributes of a Pprogressbar.

```
#include <cairo-dock-progressbar.h>
```

Data Fields

- [CairoDataRendererAttribute](#) **rendererAttribute**
General attributes of any DataRenderer.
- gchar * **cImageGradation**
image or NULL
- gdouble * **fColorGradation**
color gradation of the bar (an array of 8 doubles, representing 2 RGBA values) or NULL
- gboolean **bUseCustomPosition**
TRUE to define a custom position (by default it is placed at the middle bottom)
- [CairoOverlayPosition](#) **iCustomPosition**
custom position
- gboolean **bInverted**
invert default colors

4.28.1 Detailed Description

Attributes of a Pprogressbar.

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

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

4.29 _GldiChildProcessManagerBackend Struct Reference

```
#include <cairo-dock-utils.h>
```

Data Fields

- void(* [spawn_app](#))(const gchar *const *args, const gchar *id, const gchar *desc, const gchar *const *env, const gchar *working_dir)

4.29.1 Detailed Description

Simple "backend" for managing processes launched by us. Mainly needed to put newly launched apps in their own systemd scope / cgroup.

4.29.2 Field Documentation

4.29.2.1 spawn_app

```
void(* _GldiChildProcessManagerBackend::spawn_app) (const gchar *const *args, const gchar *id,
const gchar *desc, const gchar *const *env, const gchar *working_dir)
```

Launch a new app based on the given argument vector, performing any system-specific setup necessary.

Parameters

<i>args</i>	argument vector to use
<i>id</i>	a non-NULL identifier for the newly launched process, containing only "safe" characters (currently this means only characters valid in systemd unit names: ASCII letters, digits, ":", "-", "_", ".", and "\"); does not need to be unique @param desc a non-NULL description suitable to display to the user @param env an optional vector of environment variables to set for the launched process; each element should be in the format of "VAR=value"
<i>working_dir</i>	optionally a working directory to set for the newly launched process

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

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

4.30 _GldiContainer Struct Reference

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

```
#include <cairo-dock-container.h>
```

Data Fields

- [GldiObject](#) **object**
object.
- gpointer **pDataSlot** [CAIRO_DOCK_NB_DATA_SLOT]
External data.
- GtkWidget * **pWidget**
window of the container.
- gint **iWidth**
size of the container.
- gint **iWindowPositionX**
position of the container.
- gboolean **bInside**
TRUE is the mouse is inside the container (including the possible sub-widgets).
- [CairoDockTypeHorizontality](#) **bIsHorizontal**
TRUE if the container is horizontal, FALSE if vertical.
- gboolean **bDirectionUp**
TRUE if the container is oriented upwards, FALSE if downwards.
- guint **iSidGLAnimation**
Source ID of the animation loop.
- gint **iAnimationDeltaT**
interval of time between 2 animation steps.
- gint **iMouseX**
X position of the mouse in the container's system of reference.
- gint **iMouseY**
Y position of the mouse in the container's system of reference.
- gdouble **fSmoothScrollAccum**
accumulate smooth scroll events to emulate fixed steps
- guint **iLastScrollTime**
time of last smooth scroll event received (to filter potential duplicates)
- gdouble **fRatio**
zoom applied to the container's elements.
- gboolean **bUseReflect**
TRUE if the container has a reflection power.
- void * **glContext**
OpenGL context (either a GLXContext or EGLContext – both are typedef for a pointer).
- void * **eglSurface**
EGL surface (not needed for GLX).
- 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.
- void * **pMoveToRect**
data for the [gldi_container_move_to_rect\(\)](#) callback if needed
- void * **eglwindow**
a wl_egl_window (needed on Wayland + EGL)

4.30.1 Detailed Description

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

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

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

4.31 _GldiContainerManagerBackend Struct Reference

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

```
#include <cairo-dock-container.h>
```

Data Fields

- void(* [init_layer](#))([GldiContainer](#) *pContainer)
- gboolean(* [is_wayland](#))()
 - return if running on Wayland*
- void(* [set_keep_below](#))([GldiContainer](#) *pContainer, gboolean bKeepBelow)
- void(* [move_resize_dock](#))([CairoDock](#) *pDock)
- void(* [set_input_shape](#))([GldiContainer](#) *pContainer, cairo_region_t *pShape)
 - set input shape on a window (Wayland + EGL needs special treatment)*
- void(* [set_monitor](#))([GldiContainer](#) *pContainer, int iNumScreen)
 - set the monitor (screen) this container should appear – required on Wayland*
- void(* [update_polling_screen_edge](#))()
- gboolean(* [can_reserve_space](#))(int iNumScreen, gboolean bDirectionUp, gboolean bIsHorizontal)
 - determines if it is possible to reserve space for a dock on a given screen with a given orientation; returns TRUE by default*
- void(* [update_mouse_position](#))([GldiContainer](#) *pContainer)
 - update the mouse position based on global coordinates – only supported on X11*
- gboolean(* [dock_handle_leave](#))([CairoDock](#) *pDock, GdkEventCrossing *pEvent)
- void(* [dock_check_if_mouse_inside_linear](#))([CairoDock](#) *pDock)
- void(* [adjust_aimed_point](#))(const [Icon](#) *pIcon, GtkWidget *pWidget, int w, int h, int iMarginPosition, gdouble fAlign, int *iAimedX, int *iAimedY)

4.31.1 Detailed Description

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

4.31.2 Field Documentation

4.31.2.1 init_layer

```
void(* _GldiContainerManagerBackend::init_layer) (GldiContainer *pContainer)
```

extra functionality for Wayland / gtk_layer_shell Initialize layer-shell additions – need to be called before mapping window first

4.31.2.2 set_keep_below

```
void(* _GldiContainerManagerBackend::set_keep_below) (GldiContainer *pContainer, gboolean b↵
KeepBelow)
```

Set to keep the container's GtkWindow below or above other windows. On X11, this calls `gtk_window_set_keep↵
_below()`; on Wayland, this tries to adjust the layer the window appears on.

4.31.2.3 move_resize_dock

```
void(* _GldiContainerManagerBackend::move_resize_dock) (CairoDock *pDock)
```

Move and resize a root dock. On X11, this uses `gdk_window_move_resize()`. On Wayland, this uses `gdk_window↵
_resize()` and layer-shell anchors based on the dock's orientation.

4.31.2.4 update_polling_screen_edge

```
void(* _GldiContainerManagerBackend::update_polling_screen_edge) ()
```

recall hidden dock if the mouse is close to a screen edge update if we need to be looking at the screen edges (for any edge necessary)

4.31.2.5 dock_handle_leave

```
gboolean(* _GldiContainerManagerBackend::dock_handle_leave) (CairoDock *pDock, GdkEventCrossing↵
*pEvent)
```

backend-specific handling of leave / enter events on a dock on Wayland, these update `iMousePositionType` (this is the only place we can do this) the leave event handler should return if the mouse is really outside the dock

4.31.2.6 dock_check_if_mouse_inside_linear

```
void(* _GldiContainerManagerBackend::dock_check_if_mouse_inside_linear) (CairoDock *pDock)
```

check if the mouse is inside the dock (basic case) and update `iMousePositionType` only supported on X11

4.31.2.7 adjust_aimed_point

```
void(* _GldiContainerManagerBackend::adjust_aimed_point) (const Icon *pIcon, GtkWidget *p↵
Widget, int w, int h, int iMarginPosition, gdouble fAlign, int *iAimedX, int *iAimedY)
```

adjust the aimed point (pointed to by a subdock, menu or dialog) `AimedX` and `AimedY` is already set by the caller to a relative position, only corrections need to be done here

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

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

4.32 _GldiDesktopBackground Struct Reference

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

```
#include <cairo-dock-desktop-manager.h>
```

4.32.1 Detailed Description

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

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

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

4.33 _GldiDesktopManagerBackend Struct Reference

Definition of the Desktop Manager backend.

```
#include <cairo-dock-desktop-manager.h>
```

4.33.1 Detailed Description

Definition of the Desktop Manager backend.

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

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

4.34 _GldiManager Struct Reference

Definition of a Manager.

```
#include <cairo-dock-manager.h>
```

Data Fields

- [GldiObject](#) **object**
object
- GldiManagerInitFunc **init**
function called once and for all at the init of the core.
- GldiManagerLoadFunc **load**
function called when loading the current theme, after getting the config
- GldiManagerUnloadFunc **unload**
function called when unloading the current theme, before resetting the config.
- GldiManagerReloadFunc **reload**
function called when reloading a part of the current theme.
- GldiManagerGetConfigFunc **get_config**
function called when getting the config of the current theme, or a part of it.
- GldiManagerResetConfigFunc **reset_config**
function called when resetting the current theme, or a part of it.

4.34.1 Detailed Description

Definition of a Manager.

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

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

4.35 _GldiModule Struct Reference

Definition of an external module.

```
#include <cairo-dock-module-manager.h>
```

Data Fields

- [GldiObject](#) **object**
object
- [GldiModuleInterface](#) * **pInterface**
interface of the module.
- [GldiVisitCard](#) * **pVisitCard**
visit card of the module.
- gchar * **cConfFilePath**
conf file of the module.
- gpointer **handle**
if the module interface is provided by a dynamic library, handle to this library.
- GList * **pInstancesList**
list of instances of the module.

4.35.1 Detailed Description

Definition of an external module.

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

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

4.36 _GldiModuleInstance Struct Reference

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

```
#include <cairo-dock-module-instance-manager.h>
```

Data Fields

- [GldiObject](#) **object**
object
- [GldiModule](#) * **pModule**
the module this instance represents.
- gchar * **cConfFilePath**
conf file of the instance.
- gboolean **bCanDetach**
TRUE if the instance can be detached from docks (desklet mode).
- [Icon](#) * **pIcon**
the icon holding the instance.
- [GldiContainer](#) * **pContainer**
container of the icon.
- [CairoDock](#) * **pDock**
this field repeats the 'pContainer' field if the container is a dock, and is NULL otherwise.
- [CairoDesklet](#) * **pDesklet**
this field repeats the 'pContainer' field if the container is a desklet, and is NULL otherwise.
- cairo_t * **pDrawContext**
a drawing context on the icon.
- gint **iSlotID**
a unique ID to insert external data on icons and containers.
- gpointer **pConfig**
pointer to a structure containing the config parameters of the applet.
- gpointer **pData**
pointer to a structure containing the data of the applet.
- union {
 } **uActive**

indicator whether the module instance has already been activated (init function has been called)

4.36.1 Detailed Description

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

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

- [cairo-dock-module-instance-manager.h](#)

4.37 _GldiModuleInterface Struct Reference

Definition of the interface of a module.

```
#include <cairo-dock-module-manager.h>
```

Data Fields

- void(* [initModule](#))(GldiModuleInstance *pInstance, GKeyFile *pKeyFile)
- void(* [stopModule](#))(GldiModuleInstance *pInstance)
- gboolean(* [reloadModule](#))(GldiModuleInstance *pInstance, [GldiContainer](#) *pOldContainer, GKeyFile *pKeyFile)
- gboolean(* [read_conf_file](#))(GldiModuleInstance *pInstance, GKeyFile *pKeyFile)
- void(* [reset_config](#))(GldiModuleInstance *pInstance)
- void(* [reset_data](#))(GldiModuleInstance *pInstance)
- void(* [load_custom_widget](#))(GldiModuleInstance *pInstance, GKeyFile *pKeyFile, GSList *pWidgetList)

4.37.1 Detailed Description

Definition of the interface of a module.

4.37.2 Field Documentation

4.37.2.1 initModule

```
void(* _GldiModuleInterface::initModule) (GldiModuleInstance *pInstance, GKeyFile *pKeyFile)
```

Function called when the module is activated (e.g. enabled by the user or when loading a theme); normal functionality should be set up here (e.g. signals / callbacks). Note: pKeyFile should NOT be used as it can be NULL (this is currently the case for auto-loaded modules). All config should be read in [read_conf_file](#) () or [reloadModule](#) () below.
-> CD_APPLET_INIT_BEGIN in standard applets

4.37.2.2 stopModule

```
void(* _GldiModuleInterface::stopModule) (GldiModuleInstance *pInstance)
```

Function called when a module is deactivated (e.g. disabled by the user or when changing a theme); this should stop all functions of the module and also free all resources. -> CD_APPLET_STOP_BEGIN in standard applets

4.37.2.3 reloadModule

```
gboolean(* _GldiModuleInterface::reloadModule) (GldiModuleInstance *pInstance, GldiContainer *pOldContainer, GKeyFile *pKeyFile)
```

Function called when important configuration has changed (either for the module or for the dock that could affect this module, e.g. icon theme or icon size). pKeyFile != NULL <=> the configuration of this module has changed (use CD_APPLET_MY_CONFIG_CHANGED) -> CD_APPLET_RELOAD_BEGIN in standard applets

4.37.2.4 read_conf_file

```
gboolean(* _GldiModuleInterface::read_conf_file) (GldiModuleInstance *pInstance, GKeyFile *pKeyFile)
```

Function called to read (all of) a module's config before it is initialized. Normally, this should only read values from the supplied file and store them to be used later (ideally in the instance provided). Accessing other internals (e.g. rendering interface elements based on the config) is not recommended. For auto-loaded modules, this is called before any dock (and thus rendering context) is created. -> CD_APPLET_GET_CONFIG_BEGIN in standard applets

4.37.2.5 reset_config

```
void(* _GldiModuleInterface::reset_config) (GldiModuleInstance *pInstance)
```

Function called after the module has been stopped and should free any data (e.g. strings) in the module's configuration. -> CD_APPLET_RESET_CONFIG_BEGIN in standard applets

4.37.2.6 reset_data

```
void(* _GldiModuleInterface::reset_data) (GldiModuleInstance *pInstance)
```

Function called after module has been stopped to free any data used by it. Note: there is not much difference between using this and the stopModule () function above. -> CD_APPLET_RESET_DATA_BEGIN in standard applets

4.37.2.7 load_custom_widget

```
void(* _GldiModuleInterface::load_custom_widget) (GldiModuleInstance *pInstance, GKeyFile *pKeyFile, GSList *pWidgetList)
```

Functions used for defining custom widgets for configuring the module.

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

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

4.38 _GldiObject Struct Reference

Definition of an Object.

```
#include <cairo-dock-object.h>
```

4.38.1 Detailed Description

Definition of an Object.

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

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

4.39 _GldiObjectManager Struct Reference

Definition of an ObjectManager.

```
#include <cairo-dock-object.h>
```

4.39.1 Detailed Description

Definition of an ObjectManager.

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

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

4.40 _GldiTask Struct Reference

Definition of a periodic and/or asynchronous Task.

```
#include <cairo-dock-task.h>
```

Data Fields

- guint **iPeriod**
interval of time in seconds, 0 if the Task is to run once. Set when creating the task; can change >0 values later, but cannot change between 0 and >0.
- gpointer **pSharedMemory**
structure passed as parameter of the 'get_data' and 'update' functions. Must not be accessed outside of these 2 functions !
- gboolean **bDiscard**
TRUE when the task has been discarded.

4.40.1 Detailed Description

Definition of a periodic and/or asynchronous Task.

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

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

4.41 _GldiTextDescription Struct Reference

Description of the rendering of a text.

```
#include <cairo-dock-style-facility.h>
```

Data Fields

- gchar * **cFont**
font.
- PangoFontDescription * **fd**
pango font
- gint **iSize**
size in pixels
- gboolean **bNoDecorations**
whether to draw the decorations (frame and outline) or not
- gboolean **bUseDefaultColors**
whether to use the default colors or the colors defined below
- GldiColor **fColorStart**
text color
- GldiColor **fBackgroundColor**
background color
- GldiColor **fLineColor**
outline color
- gboolean **bOutlined**
TRUE to stroke the outline of the characters (in black).
- gint **iMargin**
margin around the text, it is also the dimension of the frame if available.
- gboolean **bUseMarkup**
whether to use Pango markups or not (markups are html-like marks, like ...; using markups force you to escape some characters like "&" -> "&")
- gdouble **fMaxRelativeWidth**
maximum width allowed, in ratio of the screen's width. Carriage returns will be inserted if necessary. 0 means no limit.

4.41.1 Detailed Description

Description of the rendering of a text.

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

- [cairo-dock-style-facility.h](#)

4.42 _GldiVisitCard Struct Reference

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

```
#include <cairo-dock-module-manager.h>
```

4.42.1 Detailed Description

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

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

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

4.43 `_GldiWindowActor` Struct Reference

Definition of a window actor.

```
#include <cairo-dock-windows-manager.h>
```

Data Fields

- gboolean **blsHidden**
not used yet...

4.43.1 Detailed Description

Definition of a window actor.

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

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

4.44 `_GldiWindowManagerBackend` Struct Reference

Definition of the Windows Manager backend.

```
#include <cairo-dock-windows-manager.h>
```

4.44.1 Detailed Description

Definition of the Windows Manager backend.

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

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

4.45 `_Icon` Struct Reference

Definition of an Icon.

```
#include <cairo-dock-icon-factory.h>
```

Data Fields

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

4.45.1 Detailed Description

Definition of an Icon.

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

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

4.46 _IconInterface Struct Reference

Icon's interface.

```
#include <cairo-dock-icon-factory.h>
```

Data Fields

- void(* **load_image**)([Icon](#) *icon)
function that loads the icon surface (and optionnally texture).
- void(* **action_on_drag_hover**)([Icon](#) *icon)
function called when the user drag something over the icon for more than 500ms.

4.46.1 Detailed Description

Icon's interface.

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

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

4.47 GldiAppInfo Struct Reference

4.47.1 Detailed Description

for launching apps, wrapping a GAppInfo. This is needed as unfortunately GDesktopAppInfo does not provide all the functionality we need.

[GldiAppInfo](#) derives from GlidObject, so you should use `gldi_object_ref ()` / `gldi_object_unref ()` to manage its lifecycle.

Note: currently, the class-manager automatically creates a [GldiAppInfo](#) for all apps registered with it, filling out the corresponding icon's pAppInfo member as it is created, so there is no need to create this object manually. A constructor is provided only to create a [GldiAppInfo](#) wrapping a custom command to run.

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

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

Chapter 5

File Documentation

5.1 cairo-dock-animations.h File Reference

Data Structures

- struct [_CairoDockTransition](#)

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

- struct [_CairoDockHidingEffect](#)

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

Macros

- #define [cairo_dock_container_is_animating](#)(pContainer)
- #define [cairo_dock_animation_will_be_visible](#)(pDock)
- #define [gldi_icon_stop_animation](#)(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](#) ([GldiContainer](#) *pContainer)
- void [gldi_icon_start_animation](#) ([Icon](#) *icon)
- void [gldi_icon_request_animation](#) ([Icon](#) *plcon, const gchar *cAnimation, int iNbRounds)
- void [gldi_icon_request_attention](#) ([Icon](#) *plcon, const gchar *cAnimation, int iNbRounds)
- void [gldi_icon_stop_attention](#) ([Icon](#) *plcon)
- void [cairo_dock_trigger_icon_removal_from_dock](#) ([Icon](#) *plcon)
- void [cairo_dock_set_transition_on_icon](#) ([Icon](#) *plcon, [GldiContainer](#) *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](#) *plcon)

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 ($\sim 33\text{Hz}$) and slow ($\sim 10\text{Hz}$), 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 `cairo_dock_container_is_animating`

```
#define cairo_dock_container_is_animating(  
    pContainer )
```

Say if a container is currently animated.

Parameters

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

5.1.2.2 `cairo_dock_animation_will_be_visible`

```
#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

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

5.1.2.3 gldi_icon_stop_animation

```
#define gldi_icon_stop_animation(  
    pIcon )
```

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

Parameters

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

5.1.2.4 cairo_dock_get_animation_delta_t

```
#define cairo_dock_get_animation_delta_t(  
    pContainer )
```

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

Parameters

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

5.1.2.5 cairo_dock_get_slow_animation_delta_t

```
#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 cairo_dock_has_transition

```
#define cairo_dock_has_transition(  
    pIcon )
```

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 cairo_dock_get_transition_count

```
#define cairo_dock_get_transition_count(  
    pIcon )
```

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

Parameters

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

Returns

the elapsed number of steps.

5.1.2.8 cairo_dock_get_transition_elapsed_time

```
#define cairo_dock_get_transition_elapsed_time(  
    pIcon )
```

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

Parameters

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

Returns

the elapsed time.

5.1.2.9 cairo_dock_get_transition_fraction

```
#define cairo_dock_get_transition_fraction(  
    pIcon )
```

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

Parameters

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

Returns

the elapsed time in [0,1].

5.1.3 Function Documentation

5.1.3.1 cairo_dock_pop_up()

```
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 cairo_dock_pop_down()

```
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 cairo_dock_launch_animation()

```
void cairo_dock_launch_animation (
    GldiContainer * pContainer )
```

Launch the animation of a Container.

Parameters

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

5.1.3.4 gldi_icon_start_animation()

```
void gldi_icon_start_animation (
    Icon * icon )
```

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

Parameters

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

5.1.3.5 gldi_icon_request_animation()

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

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

Parameters

<i>plcon</i>	the icon to animate.
<i>cAnimation</i>	name of the animation.
<i>iNbRounds</i>	number of rounds the animation will be played.

5.1.3.6 gldi_icon_request_attention()

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

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

Parameters

<i>plcon</i>	the icon
<i>cAnimation</i>	an animation name, or NULL or "default" to use the default attention animation
<i>iNbRounds</i>	number of rounds, or ≤ 0 for an endles animation

5.1.3.7 gldi_icon_stop_attention()

```
void gldi_icon_stop_attention (
    Icon * pIcon )
```

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

Parameters

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

5.1.3.8 cairo_dock_trigger_icon_removal_from_dock()

```
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
--------------	--------------------

5.1.3.9 cairo_dock_set_transition_on_icon()

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

Set a Transition on an Icon.

Parameters

<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 (optionnal).

5.1.3.10 cairo_dock_remove_transition_on_icon()

```
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_DEFINE2_ALL_BEGIN](#)(cName, iFlags, _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 CD_APPLET_DEFINE_ALL_BEGIN

```
#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>_iAppletCategory</code>	Categorie de l'applet (CAIRO_DOCK_CATEGORY_ACCESSORY, CAIRO_DOCK_CATEGORY_DESKTOP, CAIRO_DOCK_CATEGORY_CONTROLER)
<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 CD_APPLET_DEFINE_END

```
#define CD_APPLET_DEFINE_END
```

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

5.2.2.3 CD_APPLET_DEFINITION

```
#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 CD_APPLET_DEFINE2_ALL_BEGIN

```
#define CD_APPLET_DEFINE2_ALL_BEGIN(
    cName,
    iFlags,
    _iAppletCategory,
    _cDescription,
    _cAuthor )
```

New type of applet definition. Uses the pre_init function only for filling out the visit card and the post_load function for all other init.

5.2.2.5 CD_APPLET_INIT_ALL_BEGIN

```
#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

<i>pApplet</i>	une instance du module.
----------------	-------------------------

5.2.2.6 CD_APPLET_INIT_END

```
#define CD_APPLET_INIT_END
```

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

5.2.2.7 CD_APPLET_STOP_BEGIN

```
#define CD_APPLET_STOP_BEGIN
```

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

5.2.2.8 CD_APPLET_STOP_END

```
#define CD_APPLET_STOP_END
```

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

5.2.2.9 CD_APPLET_RELOAD_ALL_BEGIN

```
#define CD_APPLET_RELOAD_ALL_BEGIN
```

Debut de la fonction de rechargement de l'applet.

5.2.2.10 CD_APPLET_RELOAD_END

```
#define CD_APPLET_RELOAD_END
```

Fin de la fonction de rechargement de l'applet.

5.2.2.11 CD_APPLET_GET_CONFIG_ALL_BEGIN

```
#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).

5.2.2.12 CD_APPLET_GET_CONFIG_END

```
#define CD_APPLET_GET_CONFIG_END
```

Fin de la fonction de configuration de l'applet.

5.2.2.13 CD_APPLET_RESET_CONFIG_ALL_BEGIN

```
#define CD_APPLET_RESET_CONFIG_ALL_BEGIN
```

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

5.2.2.14 CD_APPLET_RESET_CONFIG_ALL_END

```
#define CD_APPLET_RESET_CONFIG_ALL_END
```

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

5.2.2.15 CD_APPLET_RESET_DATA_BEGIN

```
#define CD_APPLET_RESET_DATA_BEGIN
```

Debut de la fonction de liberation des donnees internes.

5.2.2.16 CD_APPLET_RESET_DATA_ALL_END

```
#define CD_APPLET_RESET_DATA_ALL_END
```

Fin de la fonction de liberation des donnees internes.

5.2.2.17 CD_APPLET_ON_CLICK_BEGIN

```
#define CD_APPLET_ON_CLICK_BEGIN
```

Debut de la fonction de notification au clic gauche.

5.2.2.18 CD_APPLET_ON_CLICK_END

```
#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.

5.2.2.19 CD_APPLET_ON_BUILD_MENU_BEGIN

```
#define CD_APPLET_ON_BUILD_MENU_BEGIN
```

Debut de la fonction de notification de construction du menu.

5.2.2.20 CD_APPLET_ON_BUILD_MENU_END

```
#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.

5.2.2.21 CD_APPLET_ON_MIDDLE_CLICK_BEGIN

```
#define CD_APPLET_ON_MIDDLE_CLICK_BEGIN
```

Debut de la fonction de notification du clic du milieu.

5.2.2.22 CD_APPLET_ON_MIDDLE_CLICK_END

```
#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.

5.2.2.23 CD_APPLET_ON_DOUBLE_CLICK_BEGIN

```
#define CD_APPLET_ON_DOUBLE_CLICK_BEGIN
```

Debut de la fonction de notification du clic du milieu.

5.2.2.24 CD_APPLET_ON_DOUBLE_CLICK_END

```
#define CD_APPLET_ON_DOUBLE_CLICK_END
```

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

5.2.2.25 CD_APPLET_ON_DROP_DATA_BEGIN

```
#define CD_APPLET_ON_DROP_DATA_BEGIN
```

Début de la fonction de notification du glisse-dépose.

5.2.2.26 CD_APPLET_ON_DROP_DATA_END

```
#define CD_APPLET_ON_DROP_DATA_END
```

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

5.2.2.27 CD_APPLET_ON_SCROLL_BEGIN

```
#define CD_APPLET_ON_SCROLL_BEGIN
```

Début de la fonction de notification au scroll.

5.2.2.28 CD_APPLET_ON_SCROLL_END

```
#define CD_APPLET_ON_SCROLL_END
```

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

5.2.2.29 CD_APPLET_ON_UPDATE_ICON_BEGIN

```
#define CD_APPLET_ON_UPDATE_ICON_BEGIN
```

Début de la fonction de notification d'update icon.

5.2.2.30 CD_APPLET_ON_UPDATE_ICON_END

```
#define CD_APPLET_ON_UPDATE_ICON_END
```

Fin de la fonction de notification d'update icon.

5.2.2.31 CD_APPLET_SKIP_UPDATE_ICON

```
#define CD_APPLET_SKIP_UPDATE_ICON
```

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

5.2.2.32 CD_APPLET_STOP_UPDATE_ICON

```
#define CD_APPLET_STOP_UPDATE_ICON
```

Quit the update function immediately with no more updates.

5.2.2.33 CD_APPLET_PAUSE_UPDATE_ICON

```
#define CD_APPLET_PAUSE_UPDATE_ICON
```

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

5.2.2.34 CD_APPLET_REGISTER_FOR_CLICK_EVENT

```
#define CD_APPLET_REGISTER_FOR_CLICK_EVENT
```

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

5.2.2.35 CD_APPLET_UNREGISTER_FOR_CLICK_EVENT

```
#define CD_APPLET_UNREGISTER_FOR_CLICK_EVENT
```

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

5.2.2.36 CD_APPLET_REGISTER_FOR_BUILD_MENU_EVENT

```
#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.

5.2.2.37 CD_APPLET_UNREGISTER_FOR_BUILD_MENU_EVENT

```
#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.

5.2.2.38 CD_APPLET_REGISTER_FOR_MIDDLE_CLICK_EVENT

```
#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.

5.2.2.39 CD_APPLET_UNREGISTER_FOR_MIDDLE_CLICK_EVENT

```
#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.

5.2.2.40 CD_APPLET_REGISTER_FOR_DOUBLE_CLICK_EVENT

```
#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.

5.2.2.41 CD_APPLET_UNREGISTER_FOR_DOUBLE_CLICK_EVENT

```
#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.

5.2.2.42 CD_APPLET_REGISTER_FOR_DROP_DATA_EVENT

```
#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.

5.2.2.43 CD_APPLET_UNREGISTER_FOR_DROP_DATA_EVENT

```
#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.

5.2.2.44 CD_APPLET_REGISTER_FOR_SCROLL_EVENT

```
#define CD_APPLET_REGISTER_FOR_SCROLL_EVENT
```

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

5.2.2.45 CD_APPLET_UNREGISTER_FOR_SCROLL_EVENT

```
#define CD_APPLET_UNREGISTER_FOR_SCROLL_EVENT
```

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

5.2.2.46 CD_APPLET_REGISTER_FOR_UPDATE_ICON_SLOW_EVENT

```
#define CD_APPLET_REGISTER_FOR_UPDATE_ICON_SLOW_EVENT
```

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

5.2.2.47 CD_APPLET_UNREGISTER_FOR_UPDATE_ICON_SLOW_EVENT

```
#define CD_APPLET_UNREGISTER_FOR_UPDATE_ICON_SLOW_EVENT
```

Unregister the applet from the slow rendering loop.

5.2.2.48 CD_APPLET_REGISTER_FOR_UPDATE_ICON_EVENT

```
#define CD_APPLET_REGISTER_FOR_UPDATE_ICON_EVENT
```

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

5.2.2.49 CD_APPLET_UNREGISTER_FOR_UPDATE_ICON_EVENT

```
#define CD_APPLET_UNREGISTER_FOR_UPDATE_ICON_EVENT
```

Unregister the applet from the fast rendering loop.

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

Macros

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

- #define `CD_APPLET_POPUP_MENU_ON_MY_ICON`(pMenu)
- #define `CD_APPLET_RELOAD_CONFIG_PANEL`
- #define `CD_APPLET_RELOAD_CONFIG_PANEL_WITH_PAGE`(iNumPage)
- #define `CD_APPLET_MY_CONF_FILE`
- #define `CD_APPLET_MY_KEY_FILE`
- #define `CD_APPLET_MY_CONFIG_CHANGED`
- #define `CD_APPLET_MY_CONTAINER_TYPE_CHANGED`
- #define `CD_APPLET_MY_OLD_CONTAINER`
- #define `CD_APPLET_CLICKED_ICON`
- #define `CD_APPLET_CLICKED_CONTAINER`
- #define `CD_APPLET_SHIFT_CLICK`
- #define `CD_APPLET_CTRL_CLICK`
- #define `CD_APPLET_ALT_CLICK`
- #define `CD_APPLET_MY_MENU`
- #define `CD_APPLET_RECEIVED_DATA`
- #define `CD_APPLET_SCROLL_UP`
- #define `CD_APPLET_SCROLL_DOWN`
- #define `CD_APPLET_BIND_KEY`(cShortKey, cDescription, cGroupName, cKeyName, handler)
- #define `CD_APPLET_REDRAW_MY_ICON`
- #define `CAIRO_DOCK_REDRAW_MY_CONTAINER`
- #define `CD_APPLET_LOAD_SURFACE_FOR_MY_APPLET`(clImagePath)
- #define `CD_APPLET_LOAD_SURFACE_FOR_MY_APPLET_WITH_DEFAULT`(cUserImageName, cDefaultLocalImageName)
- #define `CD_APPLET_SET_SURFACE_ON_MY_ICON`(pSurface)
- #define `CD_APPLET_SET_IMAGE_ON_MY_ICON`(clconName)
- #define `CD_APPLET_SET_USER_IMAGE_ON_MY_ICON`(clconName, cDefaultLocalImageName)
- #define `CD_APPLET_SET_DEFAULT_IMAGE_ON_MY_ICON_IF_NONE`
- #define `CD_APPLET_SET_NAME_FOR_MY_ICON`(clconName)
- #define `CD_APPLET_SET_NAME_FOR_MY_ICON_PRINTF`(clconNameFormat, ...)
- #define `CD_APPLET_SET_QUICK_INFO_ON_MY_ICON`(cQuickInfo)
- #define `CD_APPLET_SET_QUICK_INFO_ON_MY_ICON_PRINTF`(cQuickInfoFormat, ...)
- #define `CD_APPLET_SET_HOURS_MINUTES_AS_QUICK_INFO`(iTimeInSeconds)
- #define `CD_APPLET_SET_MINUTES_SECONDES_AS_QUICK_INFO`(iTimeInSeconds)
- #define `CD_APPLET_SET_SIZE_AS_QUICK_INFO`(iSizeInBytes)
- #define `CD_APPLET_SET_STATIC_ICON`
- #define `CD_APPLET_UNSET_STATIC_ICON`
- #define `CD_APPLET_SET_ALWAYS_VISIBLE_ICON`(bAlwaysVisible)
- #define `CD_APPLET_ANIMATE_MY_ICON`(cAnimationName, iAnimationLength)
- #define `CD_APPLET_STOP_ANIMATING_MY_ICON`
- #define `CD_APPLET_DEMANDS_ATTENTION`(cAnimationName, iAnimationLength)
- #define `CD_APPLET_STOP_DEMANDING_ATTENTION`
- #define `CD_APPLET_GET_MY_ICON_EXTENT`(iWidthPtr, iHeightPtr)
- #define `CD_APPLET_START_DRAWING_MY_ICON`
- #define `CD_APPLET_START_DRAWING_MY_ICON_CAIRO`
- #define `CD_APPLET_START_DRAWING_MY_ICON_OR_RETURN`(...)
- #define `CD_APPLET_START_DRAWING_MY_ICON_OR_RETURN_CAIRO`(...)
- #define `CD_APPLET_FINISH_DRAWING_MY_ICON`
- #define `CD_APPLET_FINISH_DRAWING_MY_ICON_CAIRO`
- #define `CD_APPLET_ADD_OVERLAY_ON_MY_ICON`(clmageFile, iPosition)
- #define `CD_APPLET_PRINT_OVERLAY_ON_MY_ICON`(clmageFile, iPosition)
- #define `CD_APPLET_REMOVE_OVERLAY_ON_MY_ICON`(iPosition)
- #define `CD_APPLET_ADD_DATA_RENDERER_ON_MY_ICON`(pAttr)
- #define `CD_APPLET_RELOAD_MY_DATA_RENDERER`(...)
- #define `CD_APPLET_RENDER_NEW_DATA_ON_MY_ICON`(pValues)
- #define `CD_APPLET_REMOVE_MY_DATA_RENDERER`

- `#define CD_APPLET_SET_MY_DATA_RENDERER_HISTORY_TO_MAX`
- `#define CD_APPLET_MY_CONTAINER_IS_OPENGL`
- `#define CD_APPLET_SET_DESKLET_RENDERER_WITH_DATA(cRendererName, pConfig)`
- `#define CD_APPLET_SET_DESKLET_RENDERER(cRendererName)`
- `#define CD_APPLET_SET_STATIC_DESKLET`
- `#define CD_APPLET_ALLOW_NO_CLICKABLE_DESKLET`
- `#define CD_APPLET_DELETE_MY_ICONS_LIST`
- `#define CD_APPLET_REMOVE_ICON_FROM_MY_ICONS_LIST(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` ,
`CAIRO_DOCK_NB_INFO_DISPLAY` }

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)
- gboolean `cairo_dock_set_image_on_icon` (cairo_t *plconContext, const gchar *clconName, Icon *plcon, GldiContainer *pContainer)
- void `cairo_dock_set_image_on_icon_with_default` (cairo_t *plconContext, const gchar *cImage, Icon *plcon, GldiContainer *pContainer, const gchar *cDefaultImagePath)
- gchar * `cairo_dock_get_human_readable_size` (long long int iSizeInBytes)
- void `cairo_dock_play_sound` (const gchar *cSoundPath)

5.3.1 Detailed Description

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

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

5.3.2 Macro Definition Documentation

5.3.2.1 `cairo_dock_set_icon_surface`

```
#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>plconContext</i>	the drawing context; is not altered by the function.
<i>pSurface</i>	the surface to apply.
<i>plcon</i>	the icon.

5.3.2.2 CD_CONFIG_GET_BOOLEAN_WITH_DEFAULT

```
#define CD_CONFIG_GET_BOOLEAN_WITH_DEFAULT(  
    cGroupName,  
    cKeyName,  
    bDefaultValue )
```

The following macros provide a convenient way to read configuration options for applets. They should only be used within the CD_APPLET_GET_CONFIG_BEGIN / CD_APPLET_GET_CONFIG_END section of an applet (usually defined in applet-config.c). 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 CD_CONFIG_GET_BOOLEAN

```
#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 CD_CONFIG_GET_INTEGER_WITH_DEFAULT

```
#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 CD_CONFIG_GET_INTEGER

```
#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 CD_CONFIG_GET_DOUBLE_WITH_DEFAULT

```
#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 CD_CONFIG_GET_DOUBLE

```
#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 CD_CONFIG_GET_INTEGER_LIST

```
#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 CD_CONFIG_GET_STRING_WITH_DEFAULT

```
#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 CD_CONFIG_GET_STRING

```
#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 CD_CONFIG_GET_FILE_PATH

```
#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 tfile if none is specified in the conf file.

Returns

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

5.3.2.12 CD_CONFIG_GET_STRING_LIST_WITH_DEFAULT

```
#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 CD_CONFIG_GET_STRING_LIST

```
#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 CD_CONFIG_GET_COLOR_RGBA_WITH_DEFAULT

```
#define CD_CONFIG_GET_COLOR_RGBA_WITH_DEFAULT(  
    cGroupName,  
    cKeyName,  
    pColorBuffer,  
    pDefaultColor )
```

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

Parameters

<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 CD_CONFIG_GET_COLOR_RGBA

```
#define CD_CONFIG_GET_COLOR_RGBA(  
    cGroupName,  
    cKeyName,  
    pColorBuffer )
```

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

Parameters

<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 CD_CONFIG_GET_COLOR_RGB_WITH_DEFAULT

```
#define CD_CONFIG_GET_COLOR_RGB_WITH_DEFAULT(  
    cGroupName,  
    cKeyName,  
    pColorBuffer,  
    pDefaultColor )
```

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

Parameters

<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 CD_CONFIG_GET_COLOR_RGB

```
#define CD_CONFIG_GET_COLOR_RGB(  
    cGroupName,  
    cKeyName,  
    pColorBuffer )
```

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

Parameters

<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 CD_CONFIG_GET_COLOR

```
#define CD_CONFIG_GET_COLOR(
    cGroupName,
    cKeyName,
    pColor )
```

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

Parameters

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

5.3.2.19 CD_CONFIG_GET_THEME_PATH

```
#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.

5.3.2.20 CD_CONFIG_GET_GAUGE_THEME

```
#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).

Returns

Path to the theme, in a newly allocated string.

5.3.2.21 CD_CONFIG_RENAME_GROUP

```
#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.

5.3.2.22 CD_APPLET_ADD_SUB_MENU_WITH_IMAGE

```
#define CD_APPLET_ADD_SUB_MENU_WITH_IMAGE(  
    cLabel,  
    pMenu,  
    cImage )
```

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>cImage</i>	name of an image (can be a path or a GtkStock).

Returns

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

5.3.2.23 CD_APPLET_ADD_SUB_MENU

```
#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.

5.3.2.24 CD_APPLET_ADD_IN_MENU_WITH_TOOLTIP_AND_DATA

```
#define CD_APPLET_ADD_IN_MENU_WITH_TOOLTIP_AND_DATA(
    cLabel,
    gtkStock,
    cToolTip,
    pCallback,
    pMenu,
    pData )
```

Create and add an entry to a menu, with an icon and a tooltip. It is recommended to use this function to add a tooltip instead of `gtk_widget_set_tooltip_text ()` as on Wayland and `gtk-layer-shell` there seems to be a race condition with GTK internals that can result in an attempt to re-show the tooltip after the menu has been closed, that leads to a protocol error and crash; see <https://github.com/wmww/gtk-layer-shell/issues/207>. This function takes care to keep the tooltip hidden when the menu has been closed.

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.

5.3.2.25 CD_APPLET_ADD_IN_MENU_WITH_STOCK_AND_DATA

```
#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.

5.3.2.26 CD_APPLET_ADD_IN_MENU_WITH_DATA

```
#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.

5.3.2.27 CD_APPLET_ADD_IN_MENU

```
#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.

5.3.2.28 CD_APPLET_ADD_IN_MENU_WITH_STOCK

```
#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.

5.3.2.29 CD_APPLET_ADD_SEPARATOR_IN_MENU

```
#define CD_APPLET_ADD_SEPARATOR_IN_MENU (
    pMenu )
```

Create and add a separator to a menu.

5.3.2.30 CD_APPLET_POPUP_MENU_ON_MY_ICON

```
#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
<i>pEvent</i>	GTK event which is the trigger (if not the currently processed event)

5.3.2.31 CD_APPLET_RELOAD_CONFIG_PANEL

```
#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.

5.3.2.32 CD_APPLET_RELOAD_CONFIG_PANEL_WITH_PAGE

```
#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.

5.3.2.33 CD_APPLET_MY_CONF_FILE

```
#define CD_APPLET_MY_CONF_FILE
```

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

5.3.2.34 CD_APPLET_MY_KEY_FILE

```
#define CD_APPLET_MY_KEY_FILE
```

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

5.3.2.35 CD_APPLET_MY_CONFIG_CHANGED

```
#define CD_APPLET_MY_CONFIG_CHANGED
```

TRUE if the conf file has changed before the reload.

5.3.2.36 CD_APPLET_MY_CONTAINER_TYPE_CHANGED

```
#define CD_APPLET_MY_CONTAINER_TYPE_CHANGED
```

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

5.3.2.37 CD_APPLET_MY_OLD_CONTAINER

```
#define CD_APPLET_MY_OLD_CONTAINER
```

The previous Container.

5.3.2.38 CD_APPLET_CLICKED_ICON

```
#define CD_APPLET_CLICKED_ICON
```

The clicked Icon.

5.3.2.39 CD_APPLET_CLICKED_CONTAINER

```
#define CD_APPLET_CLICKED_CONTAINER
```

The clicked Container.

5.3.2.40 CD_APPLET_SHIFT_CLICK

```
#define CD_APPLET_SHIFT_CLICK
```

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

5.3.2.41 CD_APPLET_CTRL_CLICK

```
#define CD_APPLET_CTRL_CLICK
```

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

5.3.2.42 CD_APPLET_ALT_CLICK

```
#define CD_APPLET_ALT_CLICK
```

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

5.3.2.43 CD_APPLET_MY_MENU

```
#define CD_APPLET_MY_MENU
```

Main menu of the applet.

5.3.2.44 CD_APPLET_RECEIVED_DATA

```
#define CD_APPLET_RECEIVED_DATA
```

Data received after a drop occurred (string).

5.3.2.45 CD_APPLET_SCROLL_UP

```
#define CD_APPLET_SCROLL_UP
```

TRUE if the user scrolled up.

5.3.2.46 CD_APPLET_SCROLL_DOWN

```
#define CD_APPLET_SCROLL_DOWN
```

TRUE if the user scrolled down.

5.3.2.47 CD_APPLET_BIND_KEY

```
#define CD_APPLET_BIND_KEY(  
    cShortcut,  
    cDescription,  
    cGroupName,  
    cKeyName,  
    handler )
```

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

Parameters

<i>cShortcut</i>	a keyboard 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 shortcut.

5.3.2.48 CD_APPLET_REDRAW_MY_ICON

```
#define CD_APPLET_REDRAW_MY_ICON
```

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

5.3.2.49 CAIRO_DOCK_REDRAW_MY_CONTAINER

```
#define CAIRO_DOCK_REDRAW_MY_CONTAINER
```

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

5.3.2.50 CD_APPLET_LOAD_SURFACE_FOR_MY_APPLET

```
#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.

5.3.2.51 CD_APPLET_LOAD_SURFACE_FOR_MY_APPLET_WITH_DEFAULT

```
#define CD_APPLET_LOAD_SURFACE_FOR_MY_APPLET_WITH_DEFAULT(  
    cUserNameImageName,  
    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>cUserNameImageName</i>	name or path of an user image.
<i>cDefaultLocalImageName</i>	default image

Returns

the newly allocated surface.

5.3.2.52 CD_APPLET_SET_SURFACE_ON_MY_ICON

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

5.3.2.53 CD_APPLET_SET_IMAGE_ON_MY_ICON

```
#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>clconName</i>	name of an icon or path to an image.
------------------	--------------------------------------

5.3.2.54 CD_APPLET_SET_USER_IMAGE_ON_MY_ICON

```
#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>clconName</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).

5.3.2.55 CD_APPLET_SET_DEFAULT_IMAGE_ON_MY_ICON_IF_NONE

```
#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.

5.3.2.56 CD_APPLET_SET_NAME_FOR_MY_ICON

```
#define CD_APPLET_SET_NAME_FOR_MY_ICON(  
    cIconName )
```


Set a new label on the applet's icon.

Parameters

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

5.3.2.57 CD_APPLET_SET_NAME_FOR_MY_ICON_PRINTF

```
#define CD_APPLET_SET_NAME_FOR_MY_ICON_PRINTF(
    cIconNameFormat,
    ... )
```

Set a new label on the applet's icon.

Parameters

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

5.3.2.58 CD_APPLET_SET_QUICK_INFO_ON_MY_ICON

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

5.3.2.59 CD_APPLET_SET_QUICK_INFO_ON_MY_ICON_PRINTF

```
#define CD_APPLET_SET_QUICK_INFO_ON_MY_ICON_PRINTF(
    cQuickInfoFormat,
    ... )
```

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

Parameters

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

5.3.2.60 CD_APPLET_SET_HOURS_MINUTES_AS_QUICK_INFO

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

5.3.2.61 CD_APPLET_SET_MINUTES_SECONDES_AS_QUICK_INFO

```
#define CD_APPLET_SET_MINUTES_SECONDES_AS_QUICK_INFO(
    iTimeInSeconds )
```

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

Parameters

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

5.3.2.62 CD_APPLET_SET_SIZE_AS_QUICK_INFO

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

5.3.2.63 CD_APPLET_SET_STATIC_ICON

```
#define CD_APPLET_SET_STATIC_ICON
```

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

5.3.2.64 CD_APPLET_UNSET_STATIC_ICON

```
#define CD_APPLET_UNSET_STATIC_ICON
```

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

5.3.2.65 CD_APPLET_SET_ALWAYS_VISIBLE_ICON

```
#define CD_APPLET_SET_ALWAYS_VISIBLE_ICON(
    bAlwaysVisible )
```

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

5.3.2.66 CD_APPLET_ANIMATE_MY_ICON

```
#define CD_APPLET_ANIMATE_MY_ICON(  
    cAnimationName,  
    iAnimationLength )
```

Launch an animation on the applet's icon.

Parameters

<i>cAnimationName</i>	name of the animation.
<i>iAnimationLength</i>	number of rounds the animation should be played.

5.3.2.67 CD_APPLET_STOP_ANIMATING_MY_ICON

```
#define CD_APPLET_STOP_ANIMATING_MY_ICON
```

Stop any animation on the applet's icon.

5.3.2.68 CD_APPLET_DEMANDS_ATTENTION

```
#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>cAnimationName</i>	name of the animation.
<i>iAnimationLength</i>	number of rounds the animation should be played, or 0 for an endless animation.

5.3.2.69 CD_APPLET_STOP_DEMANDING_ATTENTION

```
#define CD_APPLET_STOP_DEMANDING_ATTENTION
```

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

5.3.2.70 CD_APPLET_GET_MY_ICON_EXTENT

```
#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.

5.3.2.71 CD_APPLET_START_DRAWING_MY_ICON

```
#define CD_APPLET_START_DRAWING_MY_ICON
```

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

5.3.2.72 CD_APPLET_START_DRAWING_MY_ICON_CAIRO

```
#define CD_APPLET_START_DRAWING_MY_ICON_CAIRO
```

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

5.3.2.73 CD_APPLET_START_DRAWING_MY_ICON_OR_RETURN

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

5.3.2.74 CD_APPLET_START_DRAWING_MY_ICON_OR_RETURN_CAIRO

```
#define CD_APPLET_START_DRAWING_MY_ICON_OR_RETURN_CAIRO(  
    ... )
```

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

Parameters

...	value to return in case of failure.
-----	-------------------------------------

5.3.2.75 CD_APPLET_FINISH_DRAWING_MY_ICON

```
#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 CD_APPLET_FINISH_DRAWING_MY_ICON_CAIRO

```
#define CD_APPLET_FINISH_DRAWING_MY_ICON_CAIRO
```

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

5.3.2.77 CD_APPLET_ADD_OVERLAY_ON_MY_ICON

```
#define CD_APPLET_ADD_OVERLAY_ON_MY_ICON(  
    cImageFile,  
    iPosition )
```

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

Parameters

<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

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

5.3.2.78 CD_APPLET_PRINT_OVERLAY_ON_MY_ICON

```
#define CD_APPLET_PRINT_OVERLAY_ON_MY_ICON(  
    cImageFile,  
    iPosition )
```

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

Parameters

<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 printed.

5.3.2.79 CD_APPLET_REMOVE_OVERLAY_ON_MY_ICON

```
#define CD_APPLET_REMOVE_OVERLAY_ON_MY_ICON(  
    iPosition )
```

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

Parameters

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

5.3.2.80 CD_APPLET_ADD_DATA_RENDERER_ON_MY_ICON

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

5.3.2.81 CD_APPLET_RELOAD_MY_DATA_RENDERER

```
#define CD_APPLET_RELOAD_MY_DATA_RENDERER(  
    ... )
```

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

5.3.2.82 CD_APPLET_RENDER_NEW_DATA_ON_MY_ICON

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

5.3.2.83 CD_APPLET_REMOVE_MY_DATA_RENDERER

```
#define CD_APPLET_REMOVE_MY_DATA_RENDERER
```

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

5.3.2.84 CD_APPLET_SET_MY_DATA_RENDERER_HISTORY_TO_MAX

```
#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.

5.3.2.85 CD_APPLET_MY_CONTAINER_IS_OPENGL

```
#define CD_APPLET_MY_CONTAINER_IS_OPENGL
```

Say if the applet's container currently supports OpenGL.

5.3.2.86 CD_APPLET_SET_DESKLET_RENDERER_WITH_DATA

```
#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.

5.3.2.87 CD_APPLET_SET_DESKLET_RENDERER

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

5.3.2.88 CD_APPLET_SET_STATIC_DESKLET

```
#define CD_APPLET_SET_STATIC_DESKLET
```

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

5.3.2.89 CD_APPLET_ALLOW_NO_CLICKABLE_DESKLET

```
#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.

5.3.2.90 CD_APPLET_DELETE_MY_ICONS_LIST

```
#define CD_APPLET_DELETE_MY_ICONS_LIST
```

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

5.3.2.91 CD_APPLET_REMOVE_ICON_FROM_MY_ICONS_LIST

```
#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>plcon</i>	the icon to remove.
--------------	---------------------

Returns

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

5.3.2.92 CD_APPLET_DETACH_ICON_FROM_MY_ICONS_LIST

```
#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>plcon</i>	the icon to remove.
--------------	---------------------

Returns

whether the icon has been removed or not.

5.3.2.93 CD_APPLET_LOAD_MY_ICONS_LIST

```
#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>plconList</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.

5.3.2.94 CD_APPLET_ADD_ICON_IN_MY_ICONS_LIST

```
#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>plcon</i>	an icon.
--------------	----------

5.3.2.95 CD_APPLET_MY_ICONS_LIST

```
#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.

5.3.2.96 CD_APPLET_MY_ICONS_LIST_CONTAINER

```
#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.

5.3.2.97 CD_APPLET_MANAGE_APPLICATION

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

5.3.2.98 D_

```
#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 CairoDockInfoDisplay

enum `CairoDockInfoDisplay`

type of possible display on a Icon.

Enumerator

<code>CAIRO_DOCK_INFO_NONE</code>	don't display anything.
<code>CAIRO_DOCK_INFO_ON_ICON</code>	display info on the icon (as quick-info).
<code>CAIRO_DOCK_INFO_ON_LABEL</code>	display on the label of the icon.

5.3.4 Function Documentation

5.3.4.1 `cairo_dock_set_icon_surface_full()`

```
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>plconContext</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>plcon</i>	the icon.

5.3.4.2 `cairo_dock_set_image_on_icon()`

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

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

Parameters

<i>plconContext</i>	the drawing context; is not altered by the function.
---------------------	--

Parameters

<i>clconName</i>	name or path to an icon image.
<i>plcon</i>	the icon.
<i>pContainer</i>	the container of the icon.

Returns

TRUE if everything went smoothly.

5.3.4.3 cairo_dock_set_image_on_icon_with_default()

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

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

Parameters

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

5.3.4.4 cairo_dock_get_human_readable_size()

```
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.

5.3.4.5 cairo_dock_play_sound()

```
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-applet-manager.h File Reference

Macros

- `#define GLDI_OBJECT_IS_APPLET_ICON(obj)`

5.4.1 Detailed Description

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

5.4.2 Macro Definition Documentation

5.4.2.1 GLDI_OBJECT_IS_APPLET_ICON

```
#define GLDI_OBJECT_IS_APPLET_ICON(  
    obj )
```

Say if an object is a AppletIcon.

Parameters

<i>obj</i>	the object.
------------	-------------

Returns

TRUE if the object is a AppletIcon.

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

Macros

- `#define GLDI_OBJECT_IS_APPLI_ICON(obj)`

Functions

- void [cairo_dock_start_applications_manager](#) ([CairoDock](#) *pDock)
- GList * [cairo_dock_get_current_applis_list](#) (void)
- Icon * [cairo_dock_get_current_active_icon](#) (void)
- Icon * [cairo_dock_get_appli_icon](#) ([GldiWindowActor](#) *actor)
- void [cairo_dock_foreach_appli_icon](#) ([GldilConFunc](#) pFunction, gpointer pUserData)

5.5.1 Detailed Description

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

5.5.2 Macro Definition Documentation

5.5.2.1 GLDI_OBJECT_IS_APPLI_ICON

```
#define GLDI_OBJECT_IS_APPLI_ICON(  
    obj )
```

Say if an object is an AppliIcon.

Parameters

<i>obj</i>	the object.
------------	-------------

Returns

TRUE if the object is a AppliIcon.

5.5.3 Function Documentation

5.5.3.1 cairo_dock_start_applications_manager()

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

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

Parameters

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

5.5.3.2 cairo_dock_get_current_applis_list()

```
GList * cairo_dock_get_current_applis_list (  
    void )
```

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

Returns

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

5.5.3.3 cairo_dock_get_current_active_icon()

```
Icon * cairo_dock_get_current_active_icon (
    void )
```

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

Returns

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

5.5.3.4 cairo_dock_get_appli_icon()

```
Icon * cairo_dock_get_appli_icon (
    GldiWindowActor * actor )
```

Get the icon of a given window, if any.

Parameters

<i>actor</i>	the window actor
--------------	------------------

Returns

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

5.5.3.5 cairo_dock_foreach_appli_icon()

```
void cairo_dock_foreach_appli_icon (
    GldiIconFunc pFunction,
    gpointer pUserData )
```

Run a function on all Appli icons.

Parameters

<i>pFunction</i>	function to be called
<i>pUserData</i>	data passed to the function.

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

Functions

- [GldiAppInfo](#) * [gldi_app_info_new_from_commandline](#) (const gchar *cCmdline, const gchar *cName, const gchar *cWorkingDir, gboolean bNeedsTerminal)
- void [gldi_app_info_launch_action](#) ([GldiAppInfo](#) *app, const gchar *cAction)
- void [gldi_app_info_launch](#) ([GldiAppInfo](#) *app, const gchar *const *uris)
- const gchar *const * [gldi_app_info_get_desktop_actions](#) ([GldiAppInfo](#) *app)
- gchar * [gldi_app_info_get_desktop_action_name](#) ([GldiAppInfo](#) *app, const gchar *cAction)
- const gchar ** [gldi_app_info_get_supported_types](#) ([GldiAppInfo](#) *app)
- [GldiAppInfo](#) * [gldi_app_info_from_desktop_app_info](#) (GDesktopAppInfo *pDesktopAppInfo)
- void [gldi_launch_desktop_app_info](#) (GDesktopAppInfo *pDesktopAppInfo, const gchar *const *uris)
- void [gldi_app_info_set_run_in_terminal](#) ([GldiAppInfo](#) *app, gboolean bNeedsTerminal)
- void [gldi_window_foreach_inhibitor](#) ([GldiWindowActor](#) *actor, GldiIconRFunc callback, gpointer data)
- [GldiAppInfo](#) * [cairo_dock_get_class_app_info](#) (const gchar *cClass)
- gchar * [cairo_dock_register_class2](#) (const gchar *cSearchTerm, const gchar *cWmClass, gboolean bCreateAlways, gboolean blsDesktopFile)
- gchar * [cairo_dock_register_class](#) (const gchar *cSearchTerm)
- void [cairo_dock_set_data_from_class](#) (const gchar *cClass, [Icon](#) *plcon)

5.6.1 Detailed Description

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

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.

5.6.2 Function Documentation

5.6.2.1 [gldi_app_info_new_from_commandline\(\)](#)

```
GldiAppInfo * gldi\_app\_info\_new\_from\_commandline (
    const gchar * cCmdline,
    const gchar * cName,
    const gchar * cWorkingDir,
    gboolean bNeedsTerminal )
```

Create a [GldiAppInfo](#) that can be used to start the given command.

Parameters

<i>cCmdline</i>	Command to launch in the format of the XDG Desktop Entry specification.
<i>cName</i>	Descriptive name that is suitable to be displayed to the user.
<i>cWorkingDir</i>	Optionally, a directory where the command should be launched.
<i>bNeedsTerminal</i>	Whether the command should be launched in a terminal.

Returns

the newly created [GldiAppInfo](#) or NULL if there was an error parsing cCmdline.

5.6.2.2 gldi_app_info_launch_action()

```
void gldi_app_info_launch_action (
    GldiAppInfo * app,
    const gchar * cAction )
```

Launch one of the extra actions supported by this app.

Parameters

<i>app</i>	a GldiAppInfo corresponding to an installed app.
<i>cAction</i>	one of the additional actions supported by the app. The must be one of the entries returned by gldi_app_info_get_desktop_actions() .

Note: if the app supports Dbus activation, it will be used instead of directly launching it. See here for more details: <https://specifications.freedesktop.org/desktop-entry-spec/latest/dbus.html>

Apps that do not support Dbus activation might be launched directly as a child process of Cairo-Dock, or indirectly via the session manager if available (i.e. systemd on Linux).

5.6.2.3 gldi_app_info_launch()

```
void gldi_app_info_launch (
    GldiAppInfo * app,
    const gchar *const * uris )
```

Launch an application with an optional list of URIs or files to open.

Parameters

<i>app</i>	a GldiAppInfo corresponding to an installed app or available command.
<i>uris</i>	a NULL-terminated list of file names or URIs to provide as parameters or NULL.

Note: if the app supports Dbus activation, it will be used instead of directly launching it. See here for more details: <https://specifications.freedesktop.org/desktop-entry-spec/latest/dbus.html>

Apps that do not support Dbus activation might be launched directly as a child process of Cairo-Dock, or indirectly via the session manager if available (i.e. systemd on Linux).

5.6.2.4 gldi_app_info_get_desktop_actions()

```
const gchar *const * gldi_app_info_get_desktop_actions (
    GldiAppInfo * app )
```

Get a list of additional actions supported by this app. See <https://specifications.freedesktop.org/desktop-entry-spec/latest/extra-actions.html> for a description of extra actions.

Parameters

<i>app</i>	a GldiAppInfo corresponding to an installed app.
------------	--

Returns

The list of additional action names as a NULL-terminated array or NULL if no additional actions are supported.
The returned list and its contents are owned by the instance and should not be modified or freed by the caller.

5.6.2.5 gldi_app_info_get_desktop_action_name()

```
gchar * gldi_app_info_get_desktop_action_name (
    GldiAppInfo * app,
    const gchar * cAction )
```

Get the name of an additional action supported by this app that is suitable to display to the user (i.e. translated whenever possible).

Parameters

<i>app</i>	a GldiAppInfo corresponding to an installed app.
<i>cAction</i>	one of the additional actions supported by the app. The must be one of the entries returned by <code>gldi_app_info_get_desktop_actions()</code> .

Returns

the name of the action in a newly allocated string or NULL if *cAction* is invalid. The caller takes ownership of the return value and is responsible for freeing it.

5.6.2.6 gldi_app_info_get_supported_types()

```
const gchar ** gldi_app_info_get_supported_types (
    GldiAppInfo * app )
```

Get the list of mime types that this app supports or NULL if unknown.

5.6.2.7 gldi_app_info_from_desktop_app_info()

```
GldiAppInfo * gldi_app_info_from_desktop_app_info (
    GDesktopAppInfo * pDesktopAppInfo )
```

Get a [GldiAppInfo](#) corresponding to the given `GDesktopAppInfo`. Also registers this app with the class manager if this has not been done before.

Parameters

<i>pDesktopAppInfo</i>	a <code>GDesktopAppInfo</code> representing an app installed on the system.
------------------------	---

Returns

a [GldiAppInfo](#) that can be used to launch this app or NULL if pAppInfo is invalid. A reference is added and the caller should call `gldi_object_unref ()` when done using it.

5.6.2.8 gldi_launch_desktop_app_info()

```
void gldi_launch_desktop_app_info (
    GDesktopAppInfo * pDesktopAppInfo,
    const gchar *const * uris )
```

Launch an application given by a GDesktopAppInfo with an optional list of URIs or files to open. The app will be registered with the class manager if it has not been seen yet.

Parameters

<i>app</i>	a GDesktopAppInfo corresponding to an installed app.
<i>uris</i>	a NULL-terminated list of file names or URIs to provide as parameters or NULL.

Note: if the app supports DBus activation, it will be used instead of directly launching it. See here for more details: <https://specifications.freedesktop.org/desktop-entry-spec/latest/dbus.html>

Apps that do not support DBus activation might be launched directly as a child process of Cairo-Dock, or indirectly via the session manager if available (i.e. systemd on Linux).

5.6.2.9 gldi_app_info_set_run_in_terminal()

```
void gldi_app_info_set_run_in_terminal (
    GldiAppInfo * app,
    gboolean bNeedsTerminal )
```

Override the setting whether this app needs to run in a terminal.

5.6.2.10 gldi_window_foreach_inhibitor()

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

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

Parameters

<i>actor</i>	the window actor
<i>callback</i>	function to be called
<i>data</i>	data passed to the callback

5.6.2.11 cairo_dock_get_class_app_info()

```
GldiAppInfo * cairo_dock_get_class_app_info (
    const gchar * cClass )
```

Get the app info associated with this class as a [GldiAppInfo](#) object.

Parameters

<i>the</i>	class name to search for
------------	--------------------------

Returns

The app info or NULL if unknown. The caller should call `gldi_object_ref ()` on the return value if it wishes to hang on to it.

5.6.2.12 cairo_dock_register_class2()

```
gchar * cairo_dock_register_class2 (
    const gchar * cSearchTerm,
    const gchar * cWmClass,
    gboolean bCreateAlways,
    gboolean bIsDesktopFile )
```

Register an application class from apps installed on the system or find an already registered one.

Parameters

<i>cSearchTerm</i>	query to search for among installed apps (see below for details).
<i>cWmClass</i>	StartupWMClass key from a custom launcher to add as an additional key to find this class later.
<i>bCreateAlways</i>	if TRUE, a new class is always created with cSearchTerm as its key.

Returns

the class ID in a newly allocated string (can be used to retrieve class properties later).

The `cSearchTerm` supplied to this function should be either:

- a desktop path which is opened if it exists; if not, the file basename is searched among the desktop IDs of installed apps in a case-insensitive way, but no other heuristics is attempted
- a desktop file ID (i.e. a string not starting with "/" and ending with ".desktop"); it is searched among the desktop IDs of installed apps (case-insensitive); if not found, a heuristic search is also attempted
- a class name / app-id (from the StartupWMClass key of a launcher) or a command name; this is assumed to be already lowercase, and it is searched among installed apps using heuristics

Heuristics applied to search are the following:

- search among desktop file IDs by applying common suffices (org.gnome., org.kde., org.freedesktop)

- duplicating the name (e.g. "firefox" -> "firefox_firefox.desktop", required for Snap)
- searching the content of the StartupWMClass or the Exec key (if StartupWMClass is not present) in the .desktop file

The cWmClass parameter is not used for searching among installed apps, but it is added without any modification as an additional key for this class for later retrieval (so it is useful if the app uses a WMClass / app-id that is not possible to find with our heuristics).

The bCreateAlways controls whether cSearchTerm is always used to create a class:

- if bCreateAlways == FALSE, and no result is found, no class is created and NULL is returned
- if bCreateAlways == TRUE, and an app is found, a class is created as normal, but it is also ensured that cSearchTerm is added as a key for retrieval (in this case, the return value will be cSearchTerm)
- if bCreateAlways == TRUE, and no result is found, a "dummy" class is created and registered; this should be used as a last resort to ensure that a launcher has a class registered
- if blsDesktopFile == TRUE, cSearchTerm is assumed to be the name of a .desktop file and is processed accordingly (the .desktop suffix removed and converted to lowercase)

5.6.2.13 cairo_dock_register_class()

```
gchar * cairo_dock_register_class (
    const gchar * cSearchTerm )
```

Register an application class from apps installed on the system.

Parameters

<i>cSearchTerm</i>	query to search for among installed apps
--------------------	--

Returns

the class ID in a newly allocated string (can be used to retrieve class properties later).

This function behaves as `cairo_dock_register_class2(cSearchTerm, NULL, FALSE)`.

5.6.2.14 cairo_dock_set_data_from_class()

```
void cairo_dock_set_data_from_class (
    const gchar * cClass,
    Icon * pIcon )
```

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

Parameters

<i>cClass</i>	the class name
<i>pIcon</i>	the icon

5.7 cairo-dock-config.h File Reference

Functions

- void [cairo_dock_load_current_theme](#) (void)
- gboolean [cairo_dock_is_loading](#) (void)
- void [cairo_dock_decrypt_string](#) (const gchar *cEncryptedString, gchar **cDecryptedString)
- void [cairo_dock_encrypt_string](#) (const gchar *cDecryptedString, gchar **cEncryptedString)

5.7.1 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 Function Documentation

5.7.2.1 [cairo_dock_load_current_theme\(\)](#)

```
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.

5.7.2.2 [cairo_dock_is_loading\(\)](#)

```
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).

5.7.2.3 [cairo_dock_decrypt_string\(\)](#)

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

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

Parameters

<i>cEncryptedString</i>	the encrypted string.
<i>cDecryptedString</i>	the decrypted string.

5.7.2.4 cairo_dock_encrypt_string()

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

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

Parameters

<i>cDecryptedString</i>	the decrypted string.
<i>cEncryptedString</i>	the encrypted string.

5.8 cairo-dock-container.h File Reference

Data Structures

- struct [_GldiContainer](#)
Definition of a Container, whom derive Dock, Desklet, Dialog and FlyingContainer.
- struct [_GldiContainerManagerBackend](#)
Definition of the Container backend. It defines some operations that should be, but are not, provided by GTK.

Macros

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

Enumerations

- enum [GldiContainerNotifications](#) {
[NOTIFICATION_BUILD_CONTAINER_MENU](#) ,
[NOTIFICATION_BUILD_ICON_MENU](#) ,
[NOTIFICATION_CLICK_ICON](#) ,
[NOTIFICATION_DOUBLE_CLICK_ICON](#) ,
[NOTIFICATION_MIDDLE_CLICK_ICON](#) ,
[NOTIFICATION_SCROLL_ICON](#) ,
[NOTIFICATION_SMOOTH_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](#) ,
[NOTIFICATION_DROP_DATA_SELECTION](#) ,
NB_NOTIFICATIONS_CONTAINER }
signals
- enum [CairoDockTypeHorizontality](#)
Main orientation of a container.

Functions

- void [gldi_container_reserve_space](#) ([GldiContainer](#) *pContainer, int left, int right, int top, int bottom, int left_start_y, int left_end_y, int right_start_y, int right_end_y, int top_start_x, int top_end_x, int bottom_start_x, int bottom_end_x)
- gboolean [gldi_container_can_reserve_space](#) (int iNumScreen, gboolean bDirectionUp, gboolean bIsHorizontal)
determines if it is possible to reserve space for a dock on a given screen with a given orientation
- int [gldi_container_get_current_desktop_index](#) ([GldiContainer](#) *pContainer)
- void [gldi_container_move](#) ([GldiContainer](#) *pContainer, int iNumDesktop, int iAbsolutePositionX, int iAbsolutePositionY)
- gboolean [gldi_container_is_active](#) ([GldiContainer](#) *pContainer)
- void [gldi_container_present](#) ([GldiContainer](#) *pContainer)
- void [gldi_container_init_layer](#) ([GldiContainer](#) *pContainer)
- gboolean [gldi_container_is_wayland_backend](#) ()
- void [gldi_container_move_resize_dock](#) ([CairoDock](#) *pDock)
- void [gldi_container_set_screen](#) ([GldiContainer](#) *pContainer, int iNumScreen)
- void [gldi_container_move_to_rect](#) ([GldiContainer](#) *pContainer, const [GdkRectangle](#) *rect, [GdkGravity](#) rect_anchor, [GdkGravity](#) window_anchor, [GdkAnchorHints](#) anchor_hints, gdouble rel_anchor_dx, gdouble rel_anchor_dy)
- void [gldi_container_calculate_rect](#) (const [GldiContainer](#) *pContainer, const [Icon](#) *pPointedIcon, [GdkRectangle](#) *rect, [GdkGravity](#) *rect_anchor, [GdkGravity](#) *window_anchor, gboolean bSkipLabel)
- void [gldi_container_calculate_aimed_point](#) (const [Icon](#) *pIcon, [GtkWidget](#) *pWidget, int w, int h, int iMarginPosition, gdouble fAlign, int *iAimedX, int *iAimedY)
- void [gldi_container_calculate_aimed_point_base](#) (int w, int h, int iMarginPosition, gdouble fAlign, int *iAimedX, int *iAimedY)
- void [gldi_container_update_polling_screen_edge](#) (void)
update looking at the screen edges (for any edge necessary)
- gboolean [gldi_container_can_poll_screen_edge](#) (void)
check whether we can detect the mouse hitting the screen edges (for the purpose of recalling hidden docks)
- void [gldi_container_set_keep_below](#) ([GldiContainer](#) *pContainer, gboolean bKeepBelow)
- gboolean [gldi_container_dock_handle_leave](#) ([CairoDock](#) *pDock, [GdkEventCrossing](#) *pEvent)
- void [gldi_container_dock_check_if_mouse_inside_linear](#) ([CairoDock](#) *pDock)
- gboolean [gldi_container_use_new_positioning_code](#) ()
- void [cairo_dock_redraw_container](#) ([GldiContainer](#) *pContainer)
- void [cairo_dock_redraw_container_area](#) ([GldiContainer](#) *pContainer, [GdkRectangle](#) *pArea)
- void [cairo_dock_redraw_icon](#) ([Icon](#) *icon)
- [GdkAtom](#) [gldi_container_icon_dnd_atom](#) (void)
Get the [GdkAtom](#) used internally from dragging icons between docks.
- void [gldi_container_notify_drop_data](#) ([GldiContainer](#) *pContainer, gchar *cReceivedData, [Icon](#) *pPointedIcon, double fOrder)
- [GtkWidget](#) * [gldi_container_build_menu](#) ([GldiContainer](#) *pContainer, [Icon](#) *icon)

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.

5.8.2 Macro Definition Documentation

5.8.2.1 CAIRO_DOCK_IS_CONTAINER

```
#define CAIRO_DOCK_IS_CONTAINER(  
    obj )
```

Say if an object is a Container.

Parameters

<i>obj</i>	the object.
------------	-------------

Returns

TRUE if the object is a Container.

5.8.2.2 gldi_container_enable_drop

```
#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.

5.8.3 Enumeration Type Documentation

5.8.3.1 GldiContainerNotifications

```
enum GldiContainerNotifications
```

signals

Enumerator

NOTIFICATION_BUILD_CONTAINER_MENU	notification called when the menu is being built on a container. data : {Icon, GldiContainer, GtkMenu, gboolean*}
NOTIFICATION_BUILD_ICON_MENU	notification called when the menu is being built on an icon (possibly NULL). data : {Icon, GldiContainer, GtkMenu}
NOTIFICATION_CLICK_ICON	notification called when use clicks on an icon data : {Icon, CairoDock, int}

Enumerator

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 a container. data : {Icon, CairoContainer, int iDirection, int bEmulated} Note: Icon is the icon under the mouse or can be NULL if the mouse is not over any icon. Currently it is only emitted on docks and desklets. iDirection is either GDK_SCROLL_UP or GDK_SCROLL_DOWN; bEmulated is TRUE if this event is synthesized based on a series of GDK_SCROLL_SMOOTH events received earlier (so it can be ignored if those were handled)
NOTIFICATION_SMOOTH_SCROLL_ICON	notification called when the user scrolls on a container and a GDK_SCROLL_SMOOTH event was delivered data : {Icon, CairoContainer, gdouble delta_x, gdouble delta_y}
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} only called if the below NOTIFICATION_DROP_DATA_SELECTION was not handled
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.
NOTIFICATION_DROP_DATA_SELECTION	notification called when something is dropped, using the original data received data: GtkSelectionData*, Icon, double*, CairoDock, gboolean* bHandled

5.8.4 Function Documentation

5.8.4.1 gldi_container_reserve_space()

```
void gldi_container_reserve_space (
    GldiContainer * pContainer,
    int left,
    int right,
    int top,
    int bottom,
    int left_start_y,
    int left_end_y,
    int right_start_y,
    int right_end_y,
    int top_start_x,
```

```

int top_end_x,
int bottom_start_x,
int bottom_end_x )

```

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

Parameters

<i>pContainer</i>	the container
<i>left</i>	
<i>right</i>	
<i>top</i>	
<i>bottom</i>	
<i>left_start_y</i>	
<i>left_end_y</i>	
<i>right_start_y</i>	
<i>right_end_y</i>	
<i>top_start_x</i>	
<i>top_end_x</i>	
<i>bottom_start_x</i>	
<i>bottom_end_x</i>	

5.8.4.2 gldi_container_get_current_desktop_index()

```

int gldi_container_get_current_desktop_index (
    GldiContainer * pContainer )

```

Get the desktop and viewports a Container is placed on.

Parameters

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

Returns

an index representing the desktop and viewports.

5.8.4.3 gldi_container_move()

```

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

```

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

Parameters

<i>pContainer</i>	the container
<i>iNumDesktop</i>	desktop number
<i>iAbsolutePositionX</i>	horizontal position on the virtual screen
<i>iAbsolutePositionY</i>	vertical position on the virtual screen

5.8.4.4 gldi_container_is_active()

```
gboolean gldi_container_is_active (
    GldiContainer * pContainer )
```

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

Parameters

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

Returns

TRUE if the Container is the current active window.

5.8.4.5 gldi_container_present()

```
void gldi_container_present (
    GldiContainer * pContainer )
```

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

Parameters

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

5.8.4.6 gldi_container_init_layer()

```
void gldi_container_init_layer (
    GldiContainer * pContainer )
```

Make this container a layer-shell surface. This can be used to properly position a dock on the screen on wlroots-based Wayland compositors

Parameters

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

See here for more details: <https://github.com/swaywm/wlr-protocols/blob/master/unstable/wlr-layer-shell-unstable-v1.xml>

Below functions provide basic functionality to position the dock and place it above / below other windows.

5.8.4.7 `gldi_container_is_wayland_backend()`

```
gboolean gldi_container_is_wayland_backend ( )
```

determine if the display server is Wayland; this can be used by e.g. positioning code that needs to work differently under Wayland; ideally, code that needs to depend on this could be moved to the backends, but for now, that seems too complicated

5.8.4.8 `gldi_container_move_resize_dock()`

```
void gldi_container_move_resize_dock (
    CairoDock * pDock )
```

Move and resize a root dock. On X11, this uses `gdk_window_move_resize()`. On Wayland, this uses `gdk_window_move_resize()` and layer-shell anchors based on the dock's orientation.

5.8.4.9 `gldi_container_set_screen()`

```
void gldi_container_set_screen (
    GldiContainer * pContainer,
    int iNumScreen )
```

Move the dock to the given screen – only used on Wayland. On X11, this is handled by adding an offset based on a global coordinate system in `gldi_container_move_resize_dock()`.

5.8.4.10 `gldi_container_move_to_rect()`

```
void gldi_container_move_to_rect (
    GldiContainer * pContainer,
    const GdkRectangle * rect,
    GdkGravity rect_anchor,
    GdkGravity window_anchor,
    GdkAnchorHints anchor_hints,
    gdouble rel_anchor_dx,
    gdouble rel_anchor_dy )
```

Wrapper around `gdk_window_move_to_rect()` that can be called anytime. Originally, `gdk_window_move_to_rect()` can only be called after the container's window has been realized (has been associated with a `GdkWindow`). On the other hand, on Wayland with layer-shell, this needs to be set up before the container's window is mapped (it is not possible to move a popup after it was mapped). See <https://developer.gnome.org/gdk3/stable/gdk3-Windows.html#gdk-window-move-to-rect> for the description of the parameters used, except for the anchors which are interpreted as relative values compared to the width and height of the corresponding `GdkWindow`.

5.8.4.11 `gldi_container_calculate_rect()`

```
void gldi_container_calculate_rect (
    const GldiContainer * pContainer,
    const Icon * pPointedIcon,
    GdkRectangle * rect,
    GdkGravity * rect_anchor,
    GdkGravity * window_anchor,
    gboolean bSkipLabel )
```

Calculate the parameters to pass to [gldi_container_move_to_rect\(\)](#) to position a child container on the given `pContainer`, pointing to `pPointedIcon`. This can be used for subdocks, dialogs and menus. The `bSkipLabel` parameter controls whether to leave space for an icon's label on a horizontal dock (should be TRUE for subdocks and dialogs, FALSE for menus).

5.8.4.12 `gldi_container_calculate_aimed_point()`

```
void gldi_container_calculate_aimed_point (
    const Icon * pIcon,
    GtkWidget * pWidget,
    int w,
    int h,
    int iMarginPosition,
    gdouble fAlign,
    int * iAimedX,
    int * iAimedY )
```

Calculate the aimed point of sub-containers (menus and dialogs), based on relative positioning. This can be used to point an arrow to the corresponding icon. Works for menus and dialogs. Parameters: `pIcon` – the icon that is pointed by the newly placed container `pWidget` – GtkWidget of the new container `w`, `h` – width and height of the new container `iMarginPosition` – which side the margin (and the arrow) should be: 0: bottom; 1: top; 2: right; 3: left `iAimedX`, `iAimedY` – result is stored here: on Wayland, this is relative to the parent container (if exists, otherwise, relative to `pWidget`) on X11, this is in global coordinates

5.8.4.13 `gldi_container_calculate_aimed_point_base()`

```
void gldi_container_calculate_aimed_point_base (
    int w,
    int h,
    int iMarginPosition,
    gdouble fAlign,
    int * iAimedX,
    int * iAimedY )
```

Helper for the above, calculates position along the midpoint of the given edge.

5.8.4.14 `gldi_container_set_keep_below()`

```
void gldi_container_set_keep_below (
    GldiContainer * pContainer,
    gboolean bKeepBelow )
```

Set to keep the container's GtkWindow below or above other windows. On X11, this calls `gtk_window_set_keep_below()`; on Wayland, this tries to adjust the layer the window appears on.

5.8.4.15 `gldi_container_dock_handle_leave()`

```
gboolean gldi_container_dock_handle_leave (
    CairoDock * pDock,
    GdkEventCrossing * pEvent )
```

extras required for tracking mouse position: backend-specific handling of leave / enter events on a dock on Wayland, these update `iMousePositionType` (this is the only place we can do this) the leave event handler should return if the mouse is really outside the dock

5.8.4.16 `gldi_container_dock_check_if_mouse_inside_linear()`

```
void gldi_container_dock_check_if_mouse_inside_linear (
    CairoDock * pDock )
```

check if the mouse is inside the dock (basic case) and update `iMousePositionType` only supported on X11

5.8.4.17 `gldi_container_use_new_positioning_code()`

```
gboolean gldi_container_use_new_positioning_code ( )
```

return whether new code (using `gdk_window_move_to_rect()` and friends) should be used to position subdocks, menus and dialogs on Wayland, it always returns TRUE, on X11, it is based on the setting in `System / X11_new↔_rendering_code`

5.8.4.18 `cairo_dock_redraw_container()`

```
void cairo_dock_redraw_container (
    GldiContainer * pContainer )
```

Clear and trigger the redraw of a Container.

Parameters

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

5.8.4.19 `cairo_dock_redraw_container_area()`

```
void cairo_dock_redraw_container_area (
    GldiContainer * 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.

5.8.4.20 cairo_dock_redraw_icon()

```
void cairo_dock_redraw_icon (
    Icon * icon )
```

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

Parameters

<i>icon</i>	l'icone a retracer.
-------------	---------------------

5.8.4.21 gldi_container_notify_drop_data()

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

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 LAST_ORDER if the drop occurred between 2 icons.
<i>pContainer</i>	the container of the icon

5.8.4.22 gldi_container_build_menu()

```
GtkWidget * gldi_container_build_menu (
    GldiContainer * pContainer,
    Icon * icon )
```

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

Functions

- `gchar * gldi_get_diag_msg (void)`

5.9.1 Detailed Description

This class instanciates the different core managers.

5.9.2 Function Documentation

5.9.2.1 gldi_get_diag_msg()

```
gchar * gldi_get_diag_msg (  
    void )
```

Get some basic info about the features supported by Cairo-Dock and detected at runtime. Returns a dynamically allocated string that should be freed by the caller after using it.

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

Macros

- `#define GLDI_OBJECT_IS_DATA_RENDERER(obj)`

Functions

- `CairoDockGLFont * cairo_dock_get_default_data_renderer_font (void)`

5.10.1 Detailed Description

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

5.10.2 Macro Definition Documentation

5.10.2.1 GLDI_OBJECT_IS_DATA_RENDERER

```
#define GLDI_OBJECT_IS_DATA_RENDERER(  
    obj )
```

Say if an object is a DataRenderer.

Parameters

<i>obj</i>	the object.
------------	-------------

Returns

TRUE if the object is a DataRenderer.

5.10.3 Function Documentation

5.10.3.1 cairo_dock_get_default_data_renderer_font()

```
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** *pIcon, **GldiContainer** *pContainer, **CairoDataRendererAttribute** *pAttribute)
- void **cairo_dock_render_new_data_on_icon** (**Icon** *pIcon, **GldiContainer** *pContainer, **cairo_t** *pCairoContext, double *pNewValues)
- void **cairo_dock_remove_data_renderer_on_icon** (**Icon** *pIcon)
- void **cairo_dock_reload_data_renderer_on_icon** (**Icon** *pIcon, **GldiContainer** *pContainer)
- void **cairo_dock_resize_data_renderer_history** (**Icon** *pIcon, int iNewMemorySize)
- void **cairo_dock_refresh_data_renderer** (**Icon** *pIcon, **GldiContainer** *pContainer)

5.11.1 Detailed Description

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

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

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

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

5.11.2 Macro Definition Documentation

5.11.2.1 **cairo_dock_get_icon_data_renderer**

```
#define cairo_dock_get_icon_data_renderer(  
    pIcon )
```

Structure Access

5.11.2.2 **CAIRO_DATA_RENDERER**

```
#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 cairo_data_renderer_get_data

```
#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 CAIRO_DATA_RENDERER_ATTRIBUTE

```
#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 cairo_data_renderer_get_nb_values

```
#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 cairo_data_renderer_get_min_value

```
#define cairo_data_renderer_get_min_value(  
    pRenderer,  
    i )
```

Data Access Get 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 cairo_data_renderer_get_max_value

```
#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 cairo_data_renderer_get_value

```
#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 cairo_data_renderer_get_current_value

```
#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 cairo_data_renderer_get_previous_value

```
#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 cairo_data_renderer_get_normalized_value

```
#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 cairo_data_renderer_get_normalized_current_value

```
#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 cairo_data_renderer_get_normalized_previous_value

```
#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 cairo_data_renderer_get_normalized_current_value_with_latency

```
#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 cairo_data_renderer_format_value_full

```
#define cairo_data_renderer_format_value_full(  
    pRenderer,  
    i,  
    cBuffer )
```

Data Format Write 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 cairo_data_renderer_format_value

```
#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 cairo_dock_get_default_data_renderer_font()

```
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 cairo_dock_add_new_data_renderer_on_icon()

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

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

Parameters

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

5.11.3.3 cairo_dock_render_new_data_on_icon()

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

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

Parameters

<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 cairo_dock_remove_data_renderer_on_icon()

```
void cairo_dock_remove_data_renderer_on_icon (
    Icon * pIcon )
```

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

Parameters

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

5.11.3.5 cairo_dock_reload_data_renderer_on_icon()

```
void cairo_dock_reload_data_renderer_on_icon (
    Icon * pIcon,
    GldiContainer * pContainer )
```

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

Parameters

<i>pIcon</i>	the icon
<i>pContainer</i>	the icon's container

5.11.3.6 cairo_dock_resize_data_renderer_history()

```
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 cairo_dock_refresh_data_renderer()

```
void cairo_dock_refresh_data_renderer (
    Icon * pIcon,
    GldiContainer * pContainer )
```

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

Parameters

<i>pIcon</i>	the icon
<i>pContainer</i>	the icon's container

5.12 cairo-dock-dbus.h File Reference

Macros

- `#define cairo_dock_dbus_get_property_in_value(pDBusProxy, cInterface, cProperty, pProperties)`
deprecated...

Functions

- `DBusGConnection * cairo_dock_get_session_connection (void)`
- `gboolean cairo_dock_register_service_name (const gchar *cServiceName)`
- `gboolean cairo_dock_dbus_is_enabled (void)`
- `DBusGProxy * cairo_dock_create_new_session_proxy (const char *name, const char *path, const char *interface)`
- `DBusGProxy * cairo_dock_create_new_system_proxy (const char *name, const char *path, const char *interface)`
- `gboolean cairo_dock_dbus_detect_application (const gchar *cName)`
- `gboolean cairo_dock_dbus_detect_system_application (const gchar *cName)`
- `gboolean cairo_dock_dbus_get_boolean (DBusGProxy *pDBusProxy, const gchar *cAccessor)`
- `guint cairo_dock_dbus_get_uinteger (DBusGProxy *pDBusProxy, const gchar *cAccessor)`
- `int cairo_dock_dbus_get_integer (DBusGProxy *pDBusProxy, const gchar *cAccessor)`
- `gchar * cairo_dock_dbus_get_string (DBusGProxy *pDBusProxy, const gchar *cAccessor)`
- `gchar ** cairo_dock_dbus_get_string_list (DBusGProxy *pDBusProxy, const gchar *cAccessor)`
- `guchar * cairo_dock_dbus_get_uchar (DBusGProxy *pDBusProxy, const gchar *cAccessor)`
- `void cairo_dock_dbus_call (DBusGProxy *pDBusProxy, const gchar *cCommand)`

5.12.1 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 **cairo_dock_get_session_connection()**

```
DBusGConnection * cairo_dock_get_session_connection (
    void )
```

Get the connection to the 'session' Bus.

Returns

the connection to the bus.

5.12.2.2 **cairo_dock_register_service_name()**

```
gboolean cairo_dock_register_service_name (
    const gchar * cServiceName )
```

Register a new service on the session bus.

Parameters

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

Returns

TRUE in case of success, false otherwise.

5.12.2.3 cairo_dock_dbus_is_enabled()

```
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 cairo_dock_create_new_session_proxy()

```
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 cairo_dock_create_new_system_proxy()

```
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 cairo_dock_dbus_detect_application()

```
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 cairo_dock_dbus_detect_system_application()

```
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 cairo_dock_dbus_get_boolean()

```
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 cairo_dock_dbus_get_uinteger()

```
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 cairo_dock_dbus_get_integer()

```
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 cairo_dock_dbus_get_string()

```
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 `cairo_dock_dbus_get_string_list()`

```
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 `cairo_dock_dbus_get_uchar()`

```
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 `cairo_dock_dbus_call()`

```
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-desklet-factory.h File Reference

Data Structures

- struct [_CairoDeskletDecoration](#)
Decoration of a Desklet.
- struct [_CairoDeskletAttr](#)
Configuration attributes of a Desklet.
- struct [_CairoDeskletRenderer](#)
Definition of a Desklet's renderer.
- struct [_CairoDesklet](#)
Definition of a Desklet, which derives from a Container.

Macros

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

Enumerations

- enum [CairoDeskletVisibility](#) {
[CAIRO_DESKLET_NORMAL](#) ,
[CAIRO_DESKLET_KEEP_ABOVE](#) ,
[CAIRO_DESKLET_KEEP_BELOW](#) ,
[CAIRO_DESKLET_ON_WIDGET_LAYER](#) ,
[CAIRO_DESKLET_RESERVE_SPACE](#) }
Type of accessibility of a Desklet.

Functions

- [CairoDesklet](#) * [gldi_desklet_new](#) ([CairoDeskletAttr](#) *attr)
- void [gldi_desklet_add_interactive_widget_with_margin](#) ([CairoDesklet](#) *pDesklet, GtkWidget *pInteractiveWidget, int iRightMargin)
- void [gldi_desklet_set_margin](#) ([CairoDesklet](#) *pDesklet, int iRightMargin)
- GtkWidget * [gldi_desklet_steal_interactive_widget](#) ([CairoDesklet](#) *pDesklet)
- void [gldi_desklet_hide](#) ([CairoDesklet](#) *pDesklet)
- void [gldi_desklet_show](#) ([CairoDesklet](#) *pDesklet)
- void [gldi_desklet_set_accessibility](#) ([CairoDesklet](#) *pDesklet, [CairoDeskletVisibility](#) iVisibility, gboolean bSaveState)
- void [gldi_desklet_set_sticky](#) ([CairoDesklet](#) *pDesklet, gboolean bSticky)
- void [gldi_desklet_lock_position](#) ([CairoDesklet](#) *pDesklet, gboolean bPositionLocked)

5.13.1 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 functoins in [cairo-dock-desklet-manager.h](#).

5.13.2 Macro Definition Documentation

5.13.2.1 GLDI_OBJECT_IS_DESKLET

```
#define GLDI_OBJECT_IS_DESKLET(  
    obj )
```

Say if an object is a Desklet.

Parameters

<i>obj</i>	the object.
------------	-------------

Returns

TRUE if the object is a Desklet.

5.13.2.2 CAIRO_DESKLET

```
#define CAIRO_DESKLET(  
    pContainer )
```

Cast a Container into a Desklet.

Parameters

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

Returns

the desklet.

5.13.2.3 gldi_desklet_add_interactive_widget

```
#define gldi_desklet_add_interactive_widget (
    pDesklet,
    pInteractiveWidget )
```

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

Parameters

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

5.13.3 Enumeration Type Documentation

5.13.3.1 CairoDeskletVisibility

```
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

5.13.4 Function Documentation

5.13.4.1 gldi_desklet_new()

```
CairoDesklet * gldi_desklet_new (
    CairoDeskletAttr * attr )
```

Create a new desklet.

Parameters

<i>attr</i>	the attributes of the desklet
-------------	-------------------------------

Returns

the desklet.

5.13.4.2 gldi_desklet_add_interactive_widget_with_margin()

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

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

Parameters

<i>pInteractiveWidget</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.

5.13.4.3 gldi_desklet_set_margin()

```
void gldi_desklet_set_margin (
    CairoDesklet * pDesklet,
    int iRightMargin )
```

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

Parameters

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

5.13.4.4 gldi_desklet_steal_interactive_widget()

```
GtkWidget * gldi_desklet_steal_interactive_widget (
    CairoDesklet * pDesklet )
```

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

Parameters

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

Returns

the widget.

5.13.4.5 gldi_desklet_hide()

```
void gldi_desklet_hide (
    CairoDesklet * pDesklet )
```

Hide a desklet.

Parameters

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

5.13.4.6 gldi_desklet_show()

```
void gldi_desklet_show (
    CairoDesklet * pDesklet )
```

Show a desklet, and give it the focus.

Parameters

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

5.13.4.7 gldi_desklet_set_accessibility()

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

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

Parameters

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

5.13.4.8 gldi_desklet_set_sticky()

```
void gldi_desklet_set_sticky (
    CairoDesklet * pDesklet,
    gboolean bSticky )
```

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

Parameters

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

5.13.4.9 gldi_desklet_lock_position()

```
void gldi_desklet_lock_position (
    CairoDesklet * pDesklet,
    gboolean bPositionLocked )
```

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

Parameters

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

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

Typedefs

- typedef gboolean(* **GldiDeskletForeachFunc**) ([CairoDesklet](#) *pDesklet, gpointer data)
Definition of a function that runs through all desklets.

Enumerations

- enum [CairoDeskletNotifications](#) {
 NOTIFICATION_ENTER_DESKLET ,
 NOTIFICATION_LEAVE_DESKLET ,
 NOTIFICATION_CONFIGURE_DESKLET ,
 NB_NOTIFICATIONS_DESKLET }
signals

Functions

- [CairoDesklet](#) * [gldi_desklets_foreach](#) ([GldiDeskletForeachFunc](#) pCallback, gpointer user_data)
- void [gldi_desklets_foreach_icons](#) (GldiIconFunc pFunction, gpointer pUserData)
- void [gldi_desklets_set_visible](#) (gboolean bOnWidgetLayerToo)
- void [gldi_desklets_set_visibility_to_default](#) (void)

5.14.1 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 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.

5.14.2 Enumeration Type Documentation

5.14.2.1 CairoDeskletNotifications

enum [CairoDeskletNotifications](#)

signals

Enumerator

NOTIFICATION_ENTER_DESKLET	notification called when the mouse enters a desklet.
NOTIFICATION_LEAVE_DESKLET	notification called when the mouse leave a desklet.
NOTIFICATION_CONFIGURE_DESKLET	notification called when a desklet is resized or moved on the screen.

5.14.3 Function Documentation

5.14.3.1 gldi_desklets_foreach()

```
CairoDesklet * gldi_desklets_foreach (
    GldiDeskletForeachFunc pCallback,
    gpointer user_data )
```

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

Parameters

<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.

5.14.3.2 gldi_desklets_foreach_icons()

```
void gldi_desklets_foreach_icons (
    GldiIconFunc 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.

5.14.3.3 gldi_desklets_set_visible()

```
void gldi_desklets_set_visible (
    gboolean bOnWidgetLayerToo )
```

Make all desklets visible. Their accessibility is set to [CAIRO_DESKLET_NORMAL](#).

Parameters

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

5.14.3.4 gldi_desklets_set_visibility_to_default()

```
void gldi_desklets_set_visibility_to_default (
    void )
```

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

5.15 cairo-dock-desktop-file-db.h File Reference**Functions**

- void [gldi_desktop_file_db_init](#) (void)
- void [gldi_desktop_file_db_stop](#) (void)
- GDesktopAppInfo * [gldi_desktop_file_db_lookup](#) (const char *class, gboolean bOnlyDesktopID)

5.15.1 Detailed Description

This file is a part of the Cairo-Dock project

Copyright : (C) see the 'copyright' file. 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/>. Functions to maintain and query a database of all apps that are installed on a system.

5.15.2 Function Documentation

5.15.2.1 `gldi_desktop_file_db_init()`

```
void gldi_desktop_file_db_init (
    void )
```

Start the desktop file DB manager. This will run a background thread to populate the DB with all apps installed on the system.

5.15.2.2 `gldi_desktop_file_db_stop()`

```
void gldi_desktop_file_db_stop (
    void )
```

Stop the desktop file DB manager. This will delete all apps in the DB and free all memory.

5.15.2.3 `gldi_desktop_file_db_lookup()`

```
GDesktopAppInfo * gldi_desktop_file_db_lookup (
    const char * class,
    gboolean bOnlyDesktopID )
```

Try to look up an installed app. This function can block the first time it's called if the DB has not been fully populated yet.

Parameters

<i>class</i>	Desktop file ID, class or app-id of an app to look up (matching is based on the basename of its .desktop file, and the content of the StartupWMClass and Exec keys in it).
<i>bOnlyDesktopID</i>	if TRUE, only the .desktop file name is used for matching (can be useful if looking for a known .desktop file).

Returns

GDesktopAppInfo corresponding to the app if found. The return value is owned by the DB, the caller should call `g_object_ref ()` on it if it wants to keep it.

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

Data Structures

- struct [_GldiDesktopManagerBackend](#)
Definition of the Desktop Manager backend.
- struct [_GldiDesktopBackground](#)
Definition of a Desktop Background Buffer. It has a reference count so that it can be shared across all the lib.

Enumerations

- enum [CairoDesktopNotifications](#) {
[NOTIFICATION_DESKTOP_CHANGED](#) ,
[NOTIFICATION_DESKTOP_GEOMETRY_CHANGED](#) ,
[NOTIFICATION_DESKTOP_VISIBILITY_CHANGED](#) ,
[NOTIFICATION_KBD_STATE_CHANGED](#) ,
[NOTIFICATION_DESKTOP_NAMES_CHANGED](#) ,
[NOTIFICATION_DESKTOP_WALLPAPER_CHANGED](#) ,
[NOTIFICATION_SHORTKEY_PRESSED](#) ,
[NOTIFICATION_KEYMAP_CHANGED](#) ,
[NOTIFICATION_MENU_REQUEST](#) ,
[NOTIFICATION_DESKTOP_MONITOR_ADDED](#) ,
[NOTIFICATION_DESKTOP_MONITOR_REMOVED](#) ,
NB_NOTIFICATIONS_DESKTOP }
signals

Functions

- void [gldi_desktop_manager_register_backend](#) ([GldiDesktopManagerBackend](#) *pBackend, const gchar *name)
- gboolean [gldi_desktop_present_class](#) (const gchar *cClass, [GldiContainer](#) *pContainer)
- gboolean [gldi_desktop_present_windows](#) ([GldiContainer](#) *pContainer)
- gboolean [gldi_desktop_present_desktops](#) (void)
- gboolean [gldi_desktop_show_widget_layer](#) (void)
- gboolean [gldi_desktop_set_on_widget_layer](#) ([GldiContainer](#) *pContainer, gboolean bOnWidgetLayer)
- void [gldi_desktop_add_workspace](#) (void)
- void [gldi_desktop_remove_last_workspace](#) (void)
- void [gldi_desktop_get_current](#) (int *iCurrentDesktop, int *iCurrentViewportX, int *iCurrentViewportY)
- GdkMonitor *const * [gldi_desktop_get_monitors](#) (int *iNumMonitors)
Get the list of monitors currently managed – caller should not modify the GdkMonitor pointers stored here.*

5.16.1 Detailed Description

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

5.16.2 Enumeration Type Documentation

5.16.2.1 CairoDesktopNotifications

enum [CairoDesktopNotifications](#)

signals

Enumerator

NOTIFICATION_DESKTOP_CHANGED	notification called when the user switches to another desktop/viewport. data : NULL
NOTIFICATION_DESKTOP_GEOMETRY_↔ CHANGED	notification called when the geometry of the desktop has changed (number of viewports/desktops, dimensions). data: resolution-has-changed
NOTIFICATION_DESKTOP_VISIBILITY_CHANGED	notification called when the desktop is shown/hidden. data: NULL
NOTIFICATION_KBD_STATE_CHANGED	notification called when the state of the keyboard has changed.
NOTIFICATION_DESKTOP_NAMES_CHANGED	notification called when the names of the desktops have changed
NOTIFICATION_DESKTOP_WALLPAPER_↔ CHANGED	notification called when the wallpaper has changed
NOTIFICATION_SHORTKEY_PRESSED	notification called when a shortkey that has been registered by the dock is pressed. data: keycode, modifiers
NOTIFICATION_KEYMAP_CHANGED	notification called when the keymap changed, before and after updating it. data: updated
NOTIFICATION_MENU_REQUEST	notification when the user requests the desktop menu to be shown
NOTIFICATION_DESKTOP_MONITOR_ADDED	notification called when a new monitor was added, data : the GdkMonitor added
NOTIFICATION_DESKTOP_MONITOR_REMOVED	notification called when a monitor was removed, data : the GdkMonitor removed

5.16.3 Function Documentation

5.16.3.1 gldi_desktop_manager_register_backend()

```
void gldi_desktop_manager_register_backend (
    GldiDesktopManagerBackend * pBackend,
    const gchar * name )
```

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

Parameters

<i>pBackend</i>	a Desktop Manager backend; can be freed after.
-----------------	--

5.16.3.2 gldi_desktop_present_class()

```
gboolean gldi_desktop_present_class (
    const gchar * cClass,
    GldiContainer * pContainer )
```

Present all the windows of a given class.

Parameters

<i>cClass</i>	the class.
<i>pContainer</i>	currently active container which might need to be unfocused

Returns

TRUE on success

5.16.3.3 gldi_desktop_present_windows()

```
gboolean gldi_desktop_present_windows (  
    GldiContainer * pContainer )
```

Present all the windows of the current desktop.

Returns

TRUE on success

5.16.3.4 gldi_desktop_present_desktops()

```
gboolean gldi_desktop_present_desktops (  
    void )
```

Present all the desktops.

Returns

TRUE on success

5.16.3.5 gldi_desktop_show_widget_layer()

```
gboolean gldi_desktop_show_widget_layer (  
    void )
```

Show the Widget Layer.

Returns

TRUE on success

5.16.3.6 gldi_desktop_set_on_widget_layer()

```
gboolean gldi_desktop_set_on_widget_layer (  
    GldiContainer * pContainer,  
    gboolean bOnWidgetLayer )
```

Set a Container to be displayed on the Widget Layer.

Parameters

<i>pContainer</i>	a container.
<i>bOnWidgetLayer</i>	whether to set or unset the option.

Returns

TRUE on success

5.16.3.7 gldi_desktop_add_workspace()

```
void gldi_desktop_add_workspace (
    void )
```

Adds a new workspace, desktop or viewport in an implementation-defined manner. Typically this can mean adding one more workspace / desktop as the "last" one. On X11, this will resize the desktop geometry, and could result in adding multiple viewports. Might not succeed, depending on the capabilities of the backend (NOTIFICATION_↔_DESKTOP_GEOMETRY_CHANGED will be emitted if successful).

5.16.3.8 gldi_desktop_remove_last_workspace()

```
void gldi_desktop_remove_last_workspace (
    void )
```

Remove the "last" workspace desktop or viewport, according to the internal ordering of workspaces. The actual number of workspaces can be > 1, depending on the backend (on X11, if viewports are arranged in a square). Might not succeed, depending on the capabilities of the backend (NOTIFICATION_DESKTOP_GEOMETRY_CHANGED will be emitted if successful).

5.16.3.9 gldi_desktop_get_current()

```
void gldi_desktop_get_current (
    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>iCurrentViewportX</i>	will be filled with the current horizontal viewport number
<i>iCurrentViewportY</i>	will be filled with the current vertical viewport number

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/Menu decorator. It draws the frame of the Dialog/Menu.
- struct [_CairoDialog](#)
Definition of a Dialog.

Macros

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

Functions

- [CairoDialog](#) * [gldi_dialog_new](#) ([CairoDialogAttr](#) *pAttribute)
- [CairoDialog](#) * [gldi_dialog_show](#) (const gchar *cText, [Icon](#) *plcon, [GldiContainer](#) *pContainer, double fTimeLength, const gchar *clconPath, GtkWidget *pInteractiveWidget, [CairoDockActionOnAnswerFunc](#) pActionFunc, gpointer data, GFreeFunc pFreeDataFunc)
- [CairoDialog](#) * [gldi_dialog_show_temporary_with_icon_printf](#) (const gchar *cText, [Icon](#) *plcon, [GldiContainer](#) *pContainer, double fTimeLength, const gchar *clconPath,...) G_GNUC_PRINTF(1)
- [CairoDialog](#) [CairoDialog](#) * [gldi_dialog_show_temporary_with_icon](#) (const gchar *cText, [Icon](#) *plcon, [GldiContainer](#) *pContainer, double fTimeLength, const gchar *clconPath)
- [CairoDialog](#) * [gldi_dialog_show_temporary](#) (const gchar *cText, [Icon](#) *plcon, [GldiContainer](#) *pContainer, double fTimeLength)
- [CairoDialog](#) * [gldi_dialog_show_temporary_with_default_icon](#) (const gchar *cText, [Icon](#) *plcon, [GldiContainer](#) *pContainer, double fTimeLength)
- [CairoDialog](#) * [gldi_dialog_show_with_question](#) (const gchar *cText, [Icon](#) *plcon, [GldiContainer](#) *pContainer, const gchar *clconPath, [CairoDockActionOnAnswerFunc](#) pActionFunc, gpointer data, GFreeFunc pFreeDataFunc)
- [CairoDialog](#) * [gldi_dialog_show_with_entry](#) (const gchar *cText, [Icon](#) *plcon, [GldiContainer](#) *pContainer, const gchar *clconPath, const gchar *cTextForEntry, [CairoDockActionOnAnswerFunc](#) pActionFunc, gpointer data, GFreeFunc pFreeDataFunc)
- [CairoDialog](#) * [gldi_dialog_show_with_value](#) (const gchar *cText, [Icon](#) *plcon, [GldiContainer](#) *pContainer, const gchar *clconPath, double fValue, double fMaxValue, [CairoDockActionOnAnswerFunc](#) pActionFunc, gpointer data, GFreeFunc pFreeDataFunc)
- [CairoDialog](#) * [gldi_dialog_show_general_message](#) (const gchar *cMessage, double fTimeLength)
- int [gldi_dialog_show_and_wait](#) (const gchar *cText, [Icon](#) *plcon, [GldiContainer](#) *pContainer, const gchar *clconPath, GtkWidget *pInteractiveWidget)
- GtkWidget * [gldi_dialog_steal_interactive_widget](#) ([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 defines various helper functions to build a Dialog.

Note that Dialogs and Menus share the same rendering.

5.17.2 Macro Definition Documentation

5.17.2.1 CAIRO_DOCK_IS_DIALOG

```
#define CAIRO_DOCK_IS_DIALOG(  
    obj )
```

Say if an object is a Dialog.

Parameters

<i>obj</i>	the object.
------------	-------------

Returns

TRUE if the object is a dialog.

5.17.2.2 CAIRO_DIALOG

```
#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 `gldi_dialog_new()`

```
CairoDialog * gldi_dialog_new (
    CairoDialogAttr * pAttribute )
```

Create a new dialog.

Parameters

<i>pAttribute</i>	attributes of the dialog.
-------------------	---------------------------

Returns

the dialog.

5.17.3.2 `gldi_dialog_show()`

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

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

Parameters

<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>cIconPath</i>	path to an icon to display in the margin.
<i>pInteractiveWidget</i>	a GTK widget; It is destroyed with the dialog. Use 'cairo_dock_steal_interactive_widget_from_dialog()' before if you want to keep it alive.
<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.

5.17.3.3 gldi_dialog_show_temporary_with_icon_printf()

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

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

Parameters

<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.
...	arguments to insert in the message, in a printf way.

Returns

the newly created dialog.

5.17.3.4 gldi_dialog_show_temporary_with_icon()

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

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

Parameters

<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.

Returns

the newly created dialog.

5.17.3.5 gldi_dialog_show_temporary()

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

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

Parameters

<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 et visible, avec une reference a 1.

5.17.3.6 gldi_dialog_show_temporary_with_default_icon()

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

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

Parameters

<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 et visible, avec une reference a 1.

5.17.3.7 gldi_dialog_show_with_question()

```
CairoDialog * gldi_dialog_show_with_question (
    const gchar * cText,
```

```

Icon * pIcon,
GldiContainer * pContainer,
const gchar * cIconPath,
CairoDockActionOnAnswerFunc pActionFunc,
gpointer data,
GFreeFunc pFreeDataFunc )

```

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

Parameters

<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>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 et visible, avec une reference a 1.

5.17.3.8 gldi_dialog_show_with_entry()

```

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

```

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

Parameters

<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>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.

5.17.3.9 gldi_dialog_show_with_value()

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

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

Parameters

<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>cIconPath</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.

5.17.3.10 gldi_dialog_show_general_message()

```
CairoDialog * gldi_dialog_show_general_message (
    const gchar * cMessage,
    double fTimeLength )
```

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

Parameters

<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.

5.17.3.11 gldi_dialog_show_and_wait()

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

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

Parameters

<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>pInteractiveWidget</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.

5.17.3.12 gldi_dialog_steal_interactive_widget()

```
GtkWidget * gldi_dialog_steal_interactive_widget (
    CairoDialog * pDialog )
```

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

Parameters

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

Returns

the widget.

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

Typedefs

- typedef void(* **CairoDockActionOnAnswerFunc**) (int iClickedButton, GtkWidget *pInteractiveWidget, gpointer data, [CairoDialog](#) *pDialog)

Definition of a generic callback of a dialog, called when the user clicks on a button. Buttons are numbered from 0, -1 means 'Return' and -2 means 'Escape'.

Enumerations

- enum [CairoDialogNotifications](#)
signals

Functions

- void [gldi_dialogs_remove_on_icon](#) ([Icon](#) *icon)
- void [gldi_dialog_hide](#) ([CairoDialog](#) *pDialog)
- void [gldi_dialog_unhide](#) ([CairoDialog](#) *pDialog)
- void [gldi_dialog_toggle_visibility](#) ([CairoDialog](#) *pDialog)
- void [gldi_dialog_leave](#) ([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.

The most generic way to build a Dialog is to fill a `_CairoDialogAttr` and pass it to [gldi_dialog_new](#).

But in most of case, you can just use one of the following convenient functions, that will do the job for you.

- to show a message, you can use [gldi_dialog_show_temporary_with_icon](#)
- to ask the user a choice, a value or a text, you can use [gldi_dialog_show_with_question](#), [gldi_dialog_show_with_value](#) or [gldi_dialog_show_with_entry](#).
- if you want to pop up only 1 dialog at once on a given icon, use [gldi_dialogs_remove_on_icon](#) before you pop up your dialog.

5.18.2 Function Documentation

5.18.2.1 [gldi_dialogs_remove_on_icon\(\)](#)

```
void gldi_dialogs_remove_on_icon (
    Icon * icon )
```

Remove the dialogs attached to an icon.

Parameters

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

5.18.2.2 gldi_dialog_hide()

```
void gldi_dialog_hide (
    CairoDialog * pDialog )
```

Hide a dialog.

Parameters

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

5.18.2.3 gldi_dialog_unhide()

```
void gldi_dialog_unhide (
    CairoDialog * pDialog )
```

Show a dialog and give it focus.

Parameters

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

5.18.2.4 gldi_dialog_toggle_visibility()

```
void gldi_dialog_toggle_visibility (
    CairoDialog * pDialog )
```

Toggle the visibility of a dialog.

Parameters

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

5.18.2.5 gldi_dialog_leave()

```
void gldi_dialog_leave (
    CairoDialog * pDialog )
```

Notify the dialog's dock that the dialog is hidden or destroyed. This generates a "leave" event for the mouse and "unfreezes" the dock as well.

Parameters

<i>pDialog</i>	the dialog being hidden or destroyed.
----------------	---------------------------------------

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

Macros

- #define `cairo_dock_get_available_docks_for_icon`(*plcon*)

Functions

- void `cairo_dock_update_dock_size` (CairoDock **pDock*)
- Icon * `cairo_dock_calculate_dock_icons` (CairoDock **pDock*)
- void `cairo_dock_show_subdock` (Icon **pPointedIcon*, CairoDock **pParentDock*)
- GList * `cairo_dock_get_available_docks` (CairoDock **pParentDock*, CairoDock **pSubDock*)
- void `cairo_dock_calculate_icons_positions_at_rest_linear` (GList **plconList*, double *fFlatDockWidth*)
- Icon * `cairo_dock_apply_wave_effect_linear` (CairoDock **pDock*)
- double `cairo_dock_get_current_dock_width_linear` (CairoDock **pDock*)
- void `cairo_dock_check_if_mouse_inside_linear` (CairoDock **pDock*)
- void `cairo_dock_check_can_drop_linear` (CairoDock **pDock*)
- GList * `cairo_dock_get_first_drawn_element_linear` (GList **icons*)

5.19.1 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 Macro Definition Documentation

5.19.2.1 `cairo_dock_get_available_docks_for_icon`

```
#define cairo_dock_get_available_docks_for_icon(  
    pIcon )
```

Get a list of available docks where an user icon can be placed. Its current parent dock is excluded, as well as its sub-dock (if any) and its children.

Parameters

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

Returns

a list of CairoDock*

5.19.3 Function Documentation

5.19.3.1 `cairo_dock_update_dock_size()`

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

Compute the maximum size of a dock, and resize it if necessary. It takes into account the size limit, and moves the dock so that it stays centered. Also updates the dock's background if necessary, and re-place the appli thumbnails.

Parameters

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

5.19.3.2 `cairo_dock_calculate_dock_icons()`

```
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

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

Returns

the pointed icon, or NULL if none is pointed.

5.19.3.3 `cairo_dock_show_subdock()`

```
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.

5.19.3.4 `cairo_dock_get_available_docks()`

```
GList * cairo_dock_get_available_docks (
```

```
CairoDock * pParentDock,  
CairoDock * pSubDock )
```

Get a list of available docks.

Parameters

<i>pParentDock</i>	excluding this dock if not NULL
<i>pSubDock</i>	excluding this dock and its children if not NULL

Returns

a list of CairoDock*

5.19.3.5 cairo_dock_calculate_icons_positions_at_rest_linear()

```
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.

5.19.3.6 cairo_dock_apply_wave_effect_linear()

```
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.

5.19.3.7 cairo_dock_get_current_dock_width_linear()

```
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.

5.19.3.8 cairo_dock_check_if_mouse_inside_linear()

```
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 'i↔ MousePositionType' field.

Parameters

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

5.19.3.9 cairo_dock_check_can_drop_linear()

```
void cairo_dock_check_can_drop_linear (
    CairoDock * pDock )
```

Check if one can drop inside a linear dock. Drop is allowed between 2 icons of the launchers group, if the user is dragging something over the dock. Update the 'bCanDrop' field.

Parameters

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

5.19.3.10 cairo_dock_get_first_drawn_element_linear()

```
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 [GLDI_OBJECT_IS_DOCK\(obj\)](#)
- #define [CAIRO_DOCK\(p\)](#)

Functions

- [CairoDock *](#) [gldi_dock_new](#) (const gchar *cDockName)
- [CairoDock *](#) [gldi_subdock_new](#) (const gchar *cDockName, const gchar *cRendererName, [CairoDock *](#)p↔ParentDock, GList *plconList)
- void [cairo_dock_remove_icons_from_dock](#) ([CairoDock *](#)pDock, [CairoDock *](#)pReceivingDock)
- void [gldi_dock_leave_synthetic](#) ([CairoDock *](#)pDock)
- void [gldi_dock_enter_synthetic](#) ([CairoDock *](#)pDock)

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 GLDI_OBJECT_IS_DOCK

```
#define GLDI_OBJECT_IS_DOCK(  
    obj )
```

Say if an object is a Dock.

Parameters

<i>obj</i>	the object.
------------	-------------

Returns

TRUE if the object is a Dock.

5.20.2.2 CAIRO_DOCK

```
#define CAIRO_DOCK(  
    p )
```

Cast a Container into a Dock.

Parameters

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

Returns

the dock.

5.20.3 Function Documentation**5.20.3.1 gldi_dock_new()**

```
CairoDock * gldi_dock_new (  
    const gchar * cDockName )
```

Create a new root dock.

Parameters

<i>cDockName</i>	the name that identifies the dock
------------------	-----------------------------------

Returns

the new dock.

5.20.3.2 gldi_subdock_new()

```
CairoDock * gldi_subdock_new (  
    const gchar * cDockName,  
    const gchar * cRendererName,  
    CairoDock * pParentDock,  
    GList * pIconList )
```

Create a new dock of type "sub-dock", and load a given list of icons inside. The list then belongs to the dock, so it must not be 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>	the name that identifies the dock.
<i>cRendererName</i>	name of a renderer. If NULL, the default renderer will be applied.
<i>pParentDock</i>	the parent dock.
<i>pIconList</i>	a list of icons that will be loaded and inserted into the new dock (optional).

Returns

the new dock.

5.20.3.3 `cairo_dock_remove_icons_from_dock()`

```
void cairo_dock_remove_icons_from_dock (
    CairoDock * pDock,
    CairoDock * pReceivingDock )
```

Remove all icons from a dock (and its sub-docks). If the receiving dock is NULL, the icons are destroyed and removed from the current theme itself.

Parameters

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

5.20.3.4 `gldi_dock_leave_synthetic()`

```
void gldi_dock_leave_synthetic (
    CairoDock * pDock )
```

Notify pDock that the mouse has possibly left it, just as if it received the "leave-notify-event" signal from GTK. Use this e.g. when a subdock, dialog or menu is closed and the dock should shrink down if the mouse is not over it.

Parameters

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

5.20.3.5 `gldi_dock_enter_synthetic()`

```
void gldi_dock_enter_synthetic (
    CairoDock * pDock )
```

Notify pDock that the mouse has possibly entered it, just as if it received the "enter-notify-event" signal from GTK. Use this e.g. when a dialog is shown to prevent the dock from hiding.

Parameters

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

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

Macros

- #define [gldi_dock_get_name](#)(pDock)

Enumerations

- enum [GldilconSize](#)
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](#) ,
[NB_NOTIFICATIONS_DOCKS](#) }
signals

Functions

- gchar * [gldi_dock_get_readable_name](#) ([CairoDock](#) *pDock)
- [CairoDock](#) * [gldi_dock_get](#) (const gchar *cDockName)
- [Icon](#) * [cairo_dock_search_icon_pointing_on_dock](#) ([CairoDock](#) *pDock, [CairoDock](#) **pParentDock)
- void [gldi_dock_rename](#) ([CairoDock](#) *pDock, const gchar *cNewName)
- void [gldi_docks_foreach](#) (GFunc pFunction, gpointer pUserData)
- void [gldi_docks_foreach_root](#) (GFunc pFunction, gpointer pUserData)
- void [gldi_icons_foreach_in_docks](#) (GldilconFunc pFunction, gpointer pUserData)
- void [cairo_dock_reload_buffers_in_all_docks](#) (gboolean bUpdateIconSize)
- void [gldi_dock_add_conf_file_for_name](#) (const gchar *cDockName)
- gchar * [gldi_dock_add_conf_file](#) (void)
- void [gldi_docks_redraw_all_root](#) (void)
- void [cairo_dock_unhide_dock_delayed](#) ([CairoDock](#) *pDock, int iDelay)
Unhide the dock after the given delay, or instantly if iDelay == 0.
- void [gldi_dock_set_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 Macro Definition Documentation

5.21.2.1 gldi_dock_get_name

```
#define gldi_dock_get_name(  
    pDock )
```

Get the name of a Dock.

Parameters

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

Returns

the name of the dock, that identifies it.

5.21.3 Enumeration Type Documentation

5.21.3.1 CairoDocksNotifications

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}

5.21.4 Function Documentation

5.21.4.1 [gldi_dock_get_readable_name\(\)](#)

```
gchar * gldi_dock_get_readable_name (  
    CairoDock * pDock )
```

Get a readable name for a main Dock, suitable for display (like "Bottom dock"). Sub-Docks names are defined by the user, so you can just use [gldi_dock_get_name](#) for them.

Parameters

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

Returns

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

5.21.4.2 [gldi_dock_get\(\)](#)

```
CairoDock * gldi_dock_get (
```

```
const gchar * cDockName )
```

Get a Dock from a given name.

Parameters

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

Returns

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

5.21.4.3 cairo_dock_search_icon_pointing_on_dock()

```
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.

5.21.4.4 gl_di_dock_rename()

```
void gl_di_dock_rename (
    CairoDock * pDock,
    const gchar * cNewName )
```

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

Parameters

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

5.21.4.5 gl_di_docks_foreach()

```
void gl_di_docks_foreach (
    GHFunc pFunction,
    gpointer pUserData )
```

Execute an action on all docks.

Parameters

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

5.21.4.6 `gldi_docks_foreach_root()`

```
void gldi_docks_foreach_root (
    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.

5.21.4.7 `gldi_icons_foreach_in_docks()`

```
void gldi_icons_foreach_in_docks (
    GldiIconFunc 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.

5.21.4.8 `cairo_dock_reload_buffers_in_all_docks()`

```
void cairo_dock_reload_buffers_in_all_docks (
    gboolean bUpdateIconSize )
```

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

Parameters

<i>bUpdateIconSize</i>	TRUE to recalculate the icons and docks size.
------------------------	---

5.21.4.9 `gldi_dock_add_conf_file_for_name()`

```
void gldi_dock_add_conf_file_for_name (
    const gchar * cDockName )
```

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

Parameters

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

5.21.4.10 `gldi_dock_add_conf_file()`

```
gchar * gldi_dock_add_conf_file (
    void )
```

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

Returns

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

5.21.4.11 `gldi_docks_redraw_all_root()`

```
void gldi_docks_redraw_all_root (
    void )
```

Redraw every root docks.

5.21.4.12 `gldi_dock_set_visibility()`

```
void gldi_dock_set_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-dock-visibility.h File Reference

Functions

- void [gldi_dock_visibility_refresh](#) ([CairoDock](#) *pDock)
- gboolean [gldi_dock_has_overlapping_window](#) ([CairoDock](#) *pDock)

5.22.1 Detailed Description

This class manages the visibility of Docks.

5.22.2 Function Documentation

5.22.2.1 [gldi_dock_visibility_refresh\(\)](#)

```
void gldi_dock_visibility_refresh (  
    CairoDock * pDock )
```

Re-check if the given dock should be shown given its visibility settings

5.22.2.2 [gldi_dock_has_overlapping_window\(\)](#)

```
gboolean gldi_dock_has_overlapping_window (  
    CairoDock * pDock )
```

Get the whether any application window overlaps the given dock.

Parameters

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

Returns

whether an overlapping window has been found.

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

Macros

- #define [cairo_dock_create_texture_from_image](#)(cImagePath)
- #define [_cairo_dock_delete_texture](#)(iTexture)
- #define [_cairo_dock_enable_texture](#)(...)
- #define [_cairo_dock_disable_texture](#)(...)
- #define [_cairo_dock_set_alpha](#)(fAlpha)
- #define [_cairo_dock_set_blend_source](#)(...)
- #define [_cairo_dock_set_blend_alpha](#)(...)

- `#define _cairo_dock_set_blend_over(...)`
- `#define _cairo_dock_set_blend_pbuffer(...)`
- `#define _cairo_dock_apply_texture_at_size(iTexture, w, h)`
- `#define _cairo_dock_apply_texture(iTexture)`
- `#define _cairo_dock_apply_texture_at_size_with_alpha(iTexture, w, h, fAlpha)`

Functions

- void `cairo_dock_render_one_icon_opengl` (`Icon` *icon, `CairoDock` *pDock, double fDockMagnitude, gboolean bUseText)
- GLuint `cairo_dock_create_texture_from_surface_full` (`cairo_surface_t` *pImageSurface, int *pWidth, int *pHeight)
- GLuint `cairo_dock_create_texture_from_surface` (`cairo_surface_t` *pImageSurface)
- GLuint `cairo_dock_create_texture_from_raw_data` (const guchar *pTextureRaw, int iWidth, int iHeight)
- GLuint `cairo_dock_create_texture_from_image_full` (const gchar *cImagePath, double *fImageWidth, double *fImageHeight)
- void `cairo_dock_update_icon_texture` (`Icon` *pIcon)

5.23.1 Detailed Description

This class provides some useful functions to draw with OpenGL.

5.23.2 Macro Definition Documentation

5.23.2.1 `cairo_dock_create_texture_from_image`

```
#define cairo_dock_create_texture_from_image(  
    cImagePath )
```

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

Parameters

<code>cImagePath</code>	path to an image.
-------------------------	-------------------

Returns

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

5.23.2.2 `_cairo_dock_delete_texture`

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

5.23.2.3 _cairo_dock_enable_texture

```
#define _cairo_dock_enable_texture(  
    ... )
```

Enable texture drawing.

5.23.2.4 _cairo_dock_disable_texture

```
#define _cairo_dock_disable_texture(  
    ... )
```

Disable texture drawing.

5.23.2.5 _cairo_dock_set_alpha

```
#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
---------------	-------

5.23.2.6 _cairo_dock_set_blend_source

```
#define _cairo_dock_set_blend_source(  
    ... )
```

Set the color blending to overwrite.

5.23.2.7 _cairo_dock_set_blend_alpha

```
#define _cairo_dock_set_blend_alpha(  
    ... )
```

Set the color blending to mix, for premultiplied texture.

5.23.2.8 `_cairo_dock_set_blend_over`

```
#define _cairo_dock_set_blend_over(  
    ... )
```

Set the color blending to mix.

5.23.2.9 `_cairo_dock_set_blend_pbuffer`

```
#define _cairo_dock_set_blend_pbuffer(  
    ... )
```

Set the color blending to mix on a pbuffer.

5.23.2.10 `_cairo_dock_apply_texture_at_size`

```
#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

5.23.2.11 `_cairo_dock_apply_texture`

```
#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
-----------------	-------------

5.23.2.12 `_cairo_dock_apply_texture_at_size_with_alpha`

```
#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.

5.23.3 Function Documentation

5.23.3.1 cairo_dock_render_one_icon_opengl()

```
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.

5.23.3.2 cairo_dock_create_texture_from_surface_full()

```
GLuint cairo_dock_create_texture_from_surface_full (
    cairo_surface_t * pImageSurface,
    int * pWidth,
    int * pHeight )
```

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 (physical) size as the surface, but potentially rounded up to the nearest power of 2 if needed.

Parameters

<i>pImageSurface</i>	the surface, created with one of the 'cairo_dock_create_surface_xxx' functions.
<i>pWidth</i>	if not NULL, return the actual width of the newly allocated texture here
<i>pHeight</i>	if not NULL, return the actual width of the newly allocated texture here

Returns

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

5.23.3.3 cairo_dock_create_texture_from_surface()

```
GLuint cairo_dock_create_texture_from_surface (
    cairo_surface_t * pImageSurface )
```

Load a cairo surface into an OpenGL texture. The surface can be destroyed after that if you don't need it. The texture will have the same (physical) size as the surface, but potentially rounded up to the nearest power of 2 if needed. This function is the same as `cairo_dock_create_texture_from_surface_full ()` but does not return the size of the new texture.

Parameters

<i>pImageSurface</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`.

5.23.3.4 cairo_dock_create_texture_from_raw_data()

```
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`.

5.23.3.5 cairo_dock_create_texture_from_image_full()

```
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`.

5.23.3.6 `cairo_dock_update_icon_texture()`

```
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

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

5.24 `cairo-dock-draw.h` File Reference**Macros**

- #define `cairo_dock_erase_cairo_context`(pCairoContext)

Functions

- `cairo_t * cairo_dock_create_drawing_context_generic` (GldiContainer *pContainer)
CONTEXT ///.
- `cairo_t * cairo_dock_create_drawing_context_on_container` (GldiContainer *pContainer)
- `cairo_t * cairo_dock_create_drawing_context_on_area` (GldiContainer *pContainer, GdkRectangle *pArea, double *fBgColor)
- void `cairo_dock_draw_rounded_rectangle` (cairo_t *pCairoContext, double fRadius, double fLineWidth, double fFrameWidth, double fFrameHeight)
- void `cairo_dock_draw_icon_cairo` (Icon *icon, CairoDock *pDock, cairo_t *pCairoContext)
- void `cairo_dock_render_one_icon` (Icon *icon, CairoDock *pDock, cairo_t *pCairoContext, double fDock↔Magnitude, gboolean bUseText)
- void `cairo_dock_draw_string` (cairo_t *pCairoContext, CairoDock *pDock, double fStringLineWidth, gboolean blsLoop, gboolean bForceConstantSeparator)

5.24.1 Detailed Description

This class provides some useful functions to draw with libcairo.

5.24.2 Macro Definition Documentation**5.24.2.1 `cairo_dock_erase_cairo_context`**

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

5.24.3 Function Documentation

5.24.3.1 cairo_dock_create_drawing_context_generic()

```
cairo_t * cairo_dock_create_drawing_context_generic (
    GldiContainer * pContainer )
```

CONTEXT ///.

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

Parameters

<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.

5.24.3.2 cairo_dock_create_drawing_context_on_container()

```
cairo_t * cairo_dock_create_drawing_context_on_container (
    GldiContainer * pContainer )
```

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

Parameters

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

Returns

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

5.24.3.3 cairo_dock_create_drawing_context_on_area()

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

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

Parameters

<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'.

5.24.3.4 cairo_dock_draw_rounded_rectangle()

```
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.

5.24.3.5 cairo_dock_draw_icon_cairo()

```
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.

5.24.3.6 cairo_dock_render_one_icon()

```
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

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

5.24.3.7 cairo_dock_draw_string()

```

void cairo_dock_draw_string (
    cairo_t * pCairoContext,
    CairoDock * pDock,
    double fStringLineWidth,
    gboolean bIsLoop,
    gboolean bForceConstantSeparator )

```

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

Parameters

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

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 iNew↳
↳IconType, gboolean bListHiddenFiles, int iNbMaxFiles, gchar **cFullURI)
- gsize [cairo_dock_fm_measure_diretory](#) (const gchar *cBaseURI, gint iCountType, gboolean bRecursive, gint
↳*pCancel)
- gboolean [cairo_dock_fm_get_file_info](#) (const gchar *cBaseURI, gchar **cName, gchar **cURI, gchar **c↳
↳IconName, gboolean *bIsDirectory, int *iVolumeID, double *fOrder, CairoDockFMSortType iSortType)
- gboolean [cairo_dock_fm_get_file_properties](#) (const gchar *cURI, guint64 *iSize, time_t *iLastModification↳
↳Time, gchar **cMimeType, int *iUID, int *iGID, int *iPermissionsMask)
- gboolean [cairo_dock_fm_launch_uri](#) (const gchar *cURI)
- gboolean [cairo_dock_fm_add_monitor_full](#) (const gchar *cURI, gboolean bDirectory, const gchar *c↳
↳MountedURI, CairoDockFMMonitorCallback pCallback, gpointer data)
- gboolean [cairo_dock_fm_remove_monitor_full](#) (const gchar *cURI, gboolean bDirectory, const gchar *c↳
↳MountedURI)
- gboolean [cairo_dock_fm_mount_full](#) (const gchar *cURI, int iVolumeID, CairoDockFMMountCallback p↳
↳Callback, gpointer user_data)
- gboolean [cairo_dock_fm_unmount_full](#) (const gchar *cURI, int iVolumeID, CairoDockFMMountCallback p↳
↳Callback, gpointer user_data)
- gchar * [cairo_dock_fm_is_mounted](#) (const gchar *cURI, gboolean *bIsMounted)
- gboolean [cairo_dock_fm_can_eject](#) (const gchar *cURI)
- gboolean [cairo_dock_fm_eject_drive](#) (const gchar *cURI)
- gboolean [cairo_dock_fm_delete_file](#) (const gchar *cURI, gboolean bNoTrash)
- gboolean [cairo_dock_fm_rename_file](#) (const gchar *cOldURI, const gchar *cNewName)
- gboolean [cairo_dock_fm_move_file](#) (const gchar *cURI, const gchar *cDirectoryURI)
- gboolean [cairo_dock_fm_create_file](#) (const gchar *cURI, gboolean bDirectory)
- GList * [cairo_dock_fm_list_apps_for_file](#) (const gchar *cURI)
- gboolean [cairo_dock_fm_empty_trash](#) (void)
- gchar * [cairo_dock_fm_get_trash_path](#) (const gchar *cNearURI, gchar **cFileInfoPath)
- gchar * [cairo_dock_fm_get_desktop_path](#) (void)
- gboolean [cairo_dock_fm_logout](#) (void)
- gboolean [cairo_dock_fm_shutdown](#) (void)
- gboolean [cairo_dock_fm_reboot](#) (void)
- gboolean [cairo_dock_fm_lock_screen](#) (void)
- gboolean [cairo_dock_fm_setup_time](#) (void)
- gboolean [cairo_dock_fm_show_system_monitor](#) (void)
- Icon * [cairo_dock_fm_create_icon_from_URI](#) (const gchar *cURI, GldiContainer *pContainer, CairoDockFMSortType
↳iFileSortType)
- int [cairo_dock_get_file_size](#) (const gchar *cFilePath)
- int [cairo_dock_fm_get_pid](#) (const gchar *cProcessName)
- gboolean [cairo_dock_fm_monitor_pid](#) (const gchar *cProcessName, gboolean bCheckSameProcess,
↳GSourceFunc pCallback, gboolean bAlwaysLaunch, gpointer pUserData)
- gboolean [cairo_dock_fm_add_open_with_submenu](#) (GList *pAppList, const gchar *cPath, GtkWidget *p↳
↳Menu, const gchar *cLabel, const gchar *cImage, CairoDockFMOpenedWithCallback pCallback, gpointer
↳user_data)

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 `cairo_dock_fm_register_vfs_backend()`

```
void cairo_dock_fm_register_vfs_backend (
    CairoDockDesktopEnvBackend * pVFSBackend )
```

Register a environment backend, overwriting any previous backend.

5.25.2.2 `cairo_dock_fm_list_directory()`

```
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 `cairo_dock_fm_measure_diretory()`

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

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

5.25.2.4 `cairo_dock_fm_get_file_info()`

```
gboolean cairo_dock_fm_get_file_info (
    const gchar * cBaseURI,
    gchar ** cName,
    gchar ** cURI,
    gchar ** cIconName,
    gboolean * bIsDirectory,
    int * iVolumeID,
    double * fOrder,
    CairoDockFMSortType iSortType )
```

Get the main info to represent a file.

5.25.2.5 `cairo_dock_fm_get_file_properties()`

```
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 `cairo_dock_fm_launch_uri()`

```
gboolean cairo_dock_fm_launch_uri (
    const gchar * cURI )
```

Open a file with the default application.

5.25.2.7 `cairo_dock_fm_add_monitor_full()`

```
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 `cairo_dock_fm_remove_monitor_full()`

```
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 `cairo_dock_fm_mount_full()`

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

Mount a point.

5.25.2.10 `cairo_dock_fm_unmount_full()`

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

Unmount a point.

5.25.2.11 `cairo_dock_fm_is_mounted()`

```
gchar * cairo_dock_fm_is_mounted (
    const gchar * cURI,
    gboolean * bIsMounted )
```

Say if a point is currently mounted.

5.25.2.12 `cairo_dock_fm_can_eject()`

```
gboolean cairo_dock_fm_can_eject (
    const gchar * cURI )
```

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

5.25.2.13 `cairo_dock_fm_eject_drive()`

```
gboolean cairo_dock_fm_eject_drive (
    const gchar * cURI )
```

Eject a drive, like a CD player.

5.25.2.14 `cairo_dock_fm_delete_file()`

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

Delete a file.

5.25.2.15 `cairo_dock_fm_rename_file()`

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

Rename a file.

5.25.2.16 cairo_dock_fm_move_file()

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

Move a file.

5.25.2.17 cairo_dock_fm_create_file()

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

Create a new file.

5.25.2.18 cairo_dock_fm_list_apps_for_file()

```
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 GAppInfo

5.25.2.19 cairo_dock_fm_empty_trash()

```
gboolean cairo_dock_fm_empty_trash (
    void )
```

Empty the Trash.

5.25.2.20 cairo_dock_fm_get_trash_path()

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

Get the path to the Trash.

5.25.2.21 cairo_dock_fm_get_desktop_path()

```
gchar * cairo_dock_fm_get_desktop_path (
    void )
```

Get the path to the Desktop.

5.25.2.22 cairo_dock_fm_logout()

```
gboolean cairo_dock_fm_logout (
    void )
```

Raise the logout panel.

5.25.2.23 cairo_dock_fm_shutdown()

```
gboolean cairo_dock_fm_shutdown (
    void )
```

Raise the shutdown panel.

5.25.2.24 cairo_dock_fm_reboot()

```
gboolean cairo_dock_fm_reboot (
    void )
```

Raise the reboot panel.

5.25.2.25 cairo_dock_fm_lock_screen()

```
gboolean cairo_dock_fm_lock_screen (
    void )
```

Lock the screen.

5.25.2.26 cairo_dock_fm_setup_time()

```
gboolean cairo_dock_fm_setup_time (
    void )
```

Raise the panel to configure the time.

5.25.2.27 cairo_dock_fm_show_system_monitor()

```
gboolean cairo_dock_fm_show_system_monitor (
    void )
```

Raise the default system monitor.

5.25.2.28 cairo_dock_fm_create_icon_from_URI()

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

Create an Icon representing a given URI.

5.25.2.29 cairo_dock_get_file_size()

```
int cairo_dock_get_file_size (
    const gchar * cFilePath )
```

Get the size of a local file.

Parameters

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

Returns

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

5.25.2.30 cairo_dock_fm_get_pid()

```
int cairo_dock_fm_get_pid (
    const gchar * cProcessName )
```

Get process ID given its name

Parameters

<i>cProcessName</i>	name of the process
---------------------	---------------------

Returns

the PID if it exists or -1

5.25.2.31 cairo_dock_fm_monitor_pid()

```
gboolean cairo_dock_fm_monitor_pid (
    const gchar * cProcessName,
    gboolean bCheckSameProcess,
    GSourceFunc pCallback,
    gboolean bAlwaysLaunch,
    gpointer pUserData )
```

Monitor a process. Call a function when the process is no longer running

Parameters

<i>cProcessName</i>	name(es) of the process(es)
<i>bCheckSameProcess</i>	TRUE to check if first match is running. FALSE to check every time if this process name is running even if it's not the same PID.
<i>pCallback</i>	function to call when the process is no longer running
<i>bAlwaysLaunch</i>	TRUE to launch the callback function even if the process is not running or if there is an error
<i>pUserData</i>	data to pass to pCallback

Returns

FALSE if the process is not running or if there is an error

5.25.2.32 cairo_dock_fm_add_open_with_submenu()

```
gboolean cairo_dock_fm_add_open_with_submenu (
    GList * pAppList,
    const gchar * cPath,
    GtkWidget * pMenu,
    const gchar * cLabel,
    const gchar * cImage,
    CairoDockFMOpenedWithCallback pCallback,
    gpointer user_data )
```

Create a submenu for presenting options to open a file and add it to the given menu.

Parameters

<i>pAppList</i>	a list with GAppInfo elements to add to the submenu (e.g. the return value of <code>cairo_dock_fm_list_apps_for_file()</code>). Owned by the caller, but apps added to the menu will be refed, so can be freed.
<i>cPath</i>	path of the file to open
<i>pMenu</i>	menu to add a submenu to
<i>cLabel</i>	label of the submenu item
<i>cImage</i>	stock image to use with the label
<i>pCallback</i>	an optional callback function to call when the app was launched (can be NULL)
<i>user_data</i>	data to pass to pCallback

Returns

TRUE if the submenu was successfully created and added to pMenu

Note: the created submenu is managed internally and will be freed when the menu is destroyed. If given, `user_data` should remain valid while the menu is open. The callback function is only called if the app is launched, so it is possible that it is never called.

5.26 cairo-dock-gui-factory.h File Reference

Data Structures

- struct [_CairoDockGroupKeyWidget](#)
Definition of a widget corresponding to a given (group;key) pair.

Enumerations

- enum [CairoDockGUIWidgetType](#) {
[CAIRO_DOCK_WIDGET_CHECK_BUTTON](#) ,
[CAIRO_DOCK_WIDGET_CHECK_CONTROL_BUTTON](#) ,
[CAIRO_DOCK_WIDGET_SPIN_INTEGER](#) ,
[CAIRO_DOCK_WIDGET_HSCALE_INTEGER](#) ,
[CAIRO_DOCK_WIDGET_SIZE_INTEGER](#) ,
[CAIRO_DOCK_WIDGET_SPIN_DOUBLE](#) ,
[CAIRO_DOCK_WIDGET_COLOR_SELECTOR_RGB](#) ,


```

CAIRO_DOCK_WIDGET_COLOR_SELECTOR_RGBA ,
CAIRO_DOCK_WIDGET_HSCALE_DOUBLE ,
CAIRO_DOCK_WIDGET_VIEW_LIST ,
CAIRO_DOCK_WIDGET_THEME_LIST ,
CAIRO_DOCK_WIDGET_ANIMATION_LIST ,
CAIRO_DOCK_WIDGET_DIALOG_DECORATOR_LIST ,
CAIRO_DOCK_WIDGET_DESKLET_DECORATION_LIST ,
CAIRO_DOCK_WIDGET_DESKLET_DECORATION_LIST_WITH_DEFAULT ,
CAIRO_DOCK_WIDGET_DOCK_LIST ,
CAIRO_DOCK_WIDGET_ICONS_LIST ,
CAIRO_DOCK_WIDGET_ICON_THEME_LIST ,
CAIRO_DOCK_WIDGET_SCREEN_LIST ,
CAIRO_DOCK_WIDGET_JUMP_TO_MODULE ,
CAIRO_DOCK_WIDGET_JUMP_TO_MODULE_IF_EXISTS ,
CAIRO_DOCK_WIDGET_LAUNCH_COMMAND ,
CAIRO_DOCK_WIDGET_LAUNCH_COMMAND_IF_CONDITION ,
CAIRO_DOCK_WIDGET_STRING_ENTRY ,
CAIRO_DOCK_WIDGET_FILE_SELECTOR ,
CAIRO_DOCK_WIDGET_IMAGE_SELECTOR ,
CAIRO_DOCK_WIDGET_FOLDER_SELECTOR ,
CAIRO_DOCK_WIDGET_SOUND_SELECTOR ,
CAIRO_DOCK_WIDGET_SHORTKEY_SELECTOR ,
CAIRO_DOCK_WIDGET_CLASS_SELECTOR ,
CAIRO_DOCK_WIDGET_PASSWORD_ENTRY ,
CAIRO_DOCK_WIDGET_FONT_SELECTOR ,
CAIRO_DOCK_WIDGET_LIST ,
CAIRO_DOCK_WIDGET_LIST_WITH_ENTRY ,
CAIRO_DOCK_WIDGET_NUMBERED_LIST ,
CAIRO_DOCK_WIDGET_NUMBERED_CONTROL_LIST ,
CAIRO_DOCK_WIDGET_NUMBERED_CONTROL_LIST_SELECTIVE ,
CAIRO_DOCK_WIDGET_TREE_VIEW_SORT ,
CAIRO_DOCK_WIDGET_TREE_VIEW_SORT_AND_MODIFY ,
CAIRO_DOCK_WIDGET_TREE_VIEW_MULTI_CHOICE ,
CAIRO_DOCK_WIDGET_EMPTY_WIDGET ,
CAIRO_DOCK_WIDGET_EMPTY_FULL ,
CAIRO_DOCK_WIDGET_TEXT_LABEL ,
CAIRO_DOCK_WIDGET_LINK ,
CAIRO_DOCK_WIDGET_HANDBOOK ,
CAIRO_DOCK_WIDGET_SEPARATOR ,
CAIRO_DOCK_WIDGET_FRAME ,
CAIRO_DOCK_WIDGET_EXPANDER ,
CAIRO_DOCK_NB_GUI_WIDGETS }

```

Types of widgets that Cairo-Dock can automatically build.

- enum [CairoDockGUIModelColumns](#)

Model used for combo-box and tree-view. CAIRO_DOCK_MODEL_NAME is the name as displayed in the widget, and CAIRO_DOCK_MODEL_RESULT is the resulting string effectively written in the config file.

Functions

- [CairoDockGroupKeyWidget](#) * [cairo_dock_gui_find_group_key_widget_in_list](#) (GSLList *pWidgetList, const gchar *cGroupName, const gchar *cKeyName)
- GtkWidget * [cairo_dock_gui_menu_item_add](#) (GtkWidget *pMenu, const gchar *cLabel, const gchar *cImage, GCallback pFunction, gpointer pData)
- GtkWidget * [cairo_dock_gui_image_from_file](#) (const gchar *clcon, int iSize)

5.26.1 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 apppearance one, suffix the widget character with a '+'. Thus, keys not marked with a '+' won't be loaded when the user loads a theme, except if he forces it.

After the widget character and its suffix, some widget accept a list of values. For instance, a spinbutton can have a min and a max limits, a list can have pre-defined elements, etc. Such values are set between '[' and ']' brackets, and separated by ';' inside.

After that, let a blank to start the widget description. It will appear on the left of the widget; description must be short enough to fit the config panel width.

You can complete this description with a tooltip. To do that, on a new comment line, add some text between '{' and '}' brackets. Tooltips appear above the widget when you let the mouse over it for ~1 second. They can be as long as you want. Use '

' to insert new lines inside the tooltip.

5.26.2 Enumeration Type Documentation

5.26.2.1 CairoDockGUIWidgetType

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.

Enumerator

CAIRO_DOCK_WIDGET_THEME_LIST	list of themes in a combo, with preview and readme.
CAIRO_DOCK_WIDGET_ANIMATION_LIST	list of available animations.
CAIRO_DOCK_WIDGET_DIALOG_DECORATOR_↔ _LIST	list of available dialog decorators.
CAIRO_DOCK_WIDGET_DESKLET_↔ DECORATION_LIST	list of available desklet decorations.
CAIRO_DOCK_WIDGET_DESKLET_↔ DECORATION_LIST_WITH_DEFAULT	same but with the 'default' choice too.
CAIRO_DOCK_WIDGET_DOCK_LIST	list of existing docks.
CAIRO_DOCK_WIDGET_ICONS_LIST	list of icons of a dock.
CAIRO_DOCK_WIDGET_ICON_THEME_LIST	list of installed icon themes.
CAIRO_DOCK_WIDGET_SCREEN_LIST	list of screens
CAIRO_DOCK_WIDGET_JUMP_TO_MODULE	a button to jump to another module inside the config panel.
CAIRO_DOCK_WIDGET_JUMP_TO_MODULE_IF_↔ _EXISTS	same but only if the module exists.
CAIRO_DOCK_WIDGET_LAUNCH_COMMAND	a button to launch a specific command.
CAIRO_DOCK_WIDGET_LAUNCH_COMMAND_↔ IF_CONDITION	a button to launch a specific command with a condition.
CAIRO_DOCK_WIDGET_STRING_ENTRY	a text entry.
CAIRO_DOCK_WIDGET_FILE_SELECTOR	a text entry with a file selector.
CAIRO_DOCK_WIDGET_IMAGE_SELECTOR	a text entry with a file selector, files are filtered to only display images.
CAIRO_DOCK_WIDGET_FOLDER_SELECTOR	a text entry with a folder selector.
CAIRO_DOCK_WIDGET_SOUND_SELECTOR	a text entry with a file selector and a 'play' button, for sound files.
CAIRO_DOCK_WIDGET_SHORTKEY_SELECTOR	a text entry with a shortcut 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.

Enumerator

CAIRO_DOCK_WIDGET_EMPTY_FULL	an empty GtkContainer, the same but using full available space.
CAIRO_DOCK_WIDGET_TEXT_LABEL	a simple text label.
CAIRO_DOCK_WIDGET_LINK	a simple text label.
CAIRO_DOCK_WIDGET_HANDBOOK	a label containing the handbook of the applet.
CAIRO_DOCK_WIDGET_SEPARATOR	an horizontal separator.
CAIRO_DOCK_WIDGET_FRAME	a frame. The previous frame will be closed.
CAIRO_DOCK_WIDGET_EXPANDER	a frame inside an expander. The previous frame will be closed.

5.26.3 Function Documentation

5.26.3.1 cairo_dock_gui_find_group_key_widget_in_list()

```
CairoDockGroupKeyWidget * cairo_dock_gui_find_group_key_widget_in_list (
    GSList * pWidgetList,
    const gchar * cGroupName,
    const gchar * cKeyName )
```

Get a widget from a list of widgets representing a configuration window. The widgets represent a pair (group,key) as defined in the config file.

Parameters

<i>pWidgetList</i>	list of widgets built from the config file
<i>cGroupName</i>	name of the group the widget belongs to
<i>cKeyName</i>	name of the key the widget represents

Returns

the widget associated with the (group,key) , or NULL if none is found

5.26.3.2 cairo_dock_gui_menu_item_add()

```
GtkWidget * cairo_dock_gui_menu_item_add (
    GtkWidget * pMenu,
    const gchar * cLabel,
    const gchar * cImage,
    GCallback pFunction,
    gpointer pData )
```

Add a new item in a menu. Adapted from [cairo-dock-menu.h](#) and does the same as `gldi_menu_add_item ()` / `cairo_dock_add_in_menu_with_stock_and_data ()`, but does not add the custom styling used for menus belonging to docks.

Parameters

<i>pMenu</i>	the menu
<i>cLabel</i>	the label, or NULL
<i>cImage</i>	the image path or name, or NULL
<i>pFunction</i>	the callback
<i>pData</i>	the data passed to the callback

Returns

the new menu-entry that has been added.

5.26.3.3 `cairo_dock_gui_image_from_file()`

```
GtkWidget * cairo_dock_gui_image_from_file (
    const gchar * cIcon,
    int iSize )
```

Find and load an icon image at the given GtkIconSize and the default scale factor if possible.

Parameters

<i>cIcon</i>	name or full path of the image file to load
<i>iSize</i>	the GtkIconSize to use

Returns

a GtkImage that contains the desired icon if found or an empty GtkImage otherwise or NULL if cIcon was NULL.

5.27 cairo-dock-gui-manager.h File Reference

Data Structures

- struct [_CairoDockGuiBackend](#)
Definition of the GUI interface for modules.

Macros

- #define [cairo_dock_reload_current_module_widget](#)(pModuleInstance)

Typedefs

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

Functions

- void [cairo_dock_set_status_message](#) (GtkWidget *pWindow, const gchar *cMessage)
- void [cairo_dock_set_status_message_printf](#) (GtkWidget *pWindow, const gchar *cFormat,...) G_GNUC_↵
PRINTF(2)

5.27.1 Detailed Description

This class provides functions to act on configuration windows.

It also defines the interface that a GUI backend should implement.

Note: GUIs are built from a .conf file; .conf files are normal group/key files, but with some special indications in the comments. Each key will be represented by a pre-defined widget, that is defined by the first 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.

5.27.2 Macro Definition Documentation

5.27.2.1 cairo_dock_reload_current_module_widget

```
#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>pModuleInstance</i>	an instance of a module.
------------------------	--------------------------

5.27.3 Function Documentation

5.27.3.1 cairo_dock_set_status_message()

```
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.

5.27.3.2 cairo_dock_set_status_message_printf()

```
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.

5.28 cairo-dock-icon-facility.h File Reference

Macros

- `#define cairo_dock_icon_is_being_inserted(icon)`
- `#define cairo_dock_icon_is_being_removed(icon)`
- `#define cairo_dock_get_icon_order(icon)`
- `#define cairo_dock_get_next_element(ic, list)`
- `#define cairo_dock_get_previous_element(ic, list)`
- `#define cairo_dock_set_icon_static(icon, _bStatic)`
- `#define cairo_dock_set_icon_always_visible(icon, _bAlwaysVisible)`
- `#define gldi_icon_mark_as_launching(plcon)`
- `#define gldi_icon_is_launching(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`)
- `void cairo_dock_get_icon_extnt` (`Icon *plcon`, `int *iWidth`, `int *iHeight`)
- `void cairo_dock_get_current_icon_size` (`Icon *plcon`, `GldiContainer *pContainer`, `double *fSizeX`, `double *fSizeY`)
- `void cairo_dock_compute_icon_area` (`Icon *icon`, `GldiContainer *pContainer`, `GdkRectangle *pArea`)
- `void gldi_icon_set_name` (`Icon *plcon`, `const gchar *clconName`)
- `void gldi_icon_set_name_printf` (`Icon *plcon`, `const gchar *clconNameFormat`,...) `G_GNUC_PRINTF(2`
- `void void gldi_icon_set_quick_info` (`Icon *plcon`, `const gchar *cQuickInfo`)
- `void gldi_icon_set_quick_info_printf` (`Icon *plcon`, `const gchar *cQuickInfoFormat`,...) `G_GNUC_PRINTF(2`
- `cairo_surface_t * cairo_dock_icon_buffer_to_cairo` (`Icon *icon`, `int iWidth`, `int iHeight`)
- `gboolean cairo_dock_begin_draw_icon` (`Icon *plcon`, `gint iRenderingMode`)
- `void cairo_dock_end_draw_icon` (`Icon *plcon`)

5.28.1 Detailed Description

This class provides utility functions on Icons.

5.28.2 Macro Definition Documentation

5.28.2.1 `cairo_dock_icon_is_being_inserted`

```
#define cairo_dock_icon_is_being_inserted(  
    icon )
```

Say whether an icon is currently being inserted.

5.28.2.2 `cairo_dock_icon_is_being_removed`

```
#define cairo_dock_icon_is_being_removed(  
    icon )
```

Say whether an icon is currently being removed.

5.28.2.3 `cairo_dock_get_icon_order`

```
#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.

5.28.2.4 `cairo_dock_get_next_element`

```
#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.

5.28.2.5 cairo_dock_get_previous_element

```
#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.

5.28.2.6 cairo_dock_set_icon_static

```
#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.

5.28.2.7 cairo_dock_set_icon_always_visible

```
#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.

5.28.2.8 gldi_icon_mark_as_launching

```
#define gldi_icon_mark_as_launching(
    pIcon )
```

Mark an Icon as 'launching'. This states lasts until the corresponding window appears (with a timeout of 15 seconds). Typically used to prevent the program from being started 2 times in a row, or to keep the animation running until the program is started.

5.28.2.9 gldi_icon_is_launching

```
#define gldi_icon_is_launching(
    pIcon )
```

Tell if an Icon is being launched.

5.28.3 Function Documentation**5.28.3.1 cairo_dock_get_icon_type()**

```
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.

5.28.3.2 cairo_dock_compare_icons_order()

```
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 $icon1 = icon2$.

5.28.3.3 cairo_dock_compare_icons_name()

```
int cairo_dock_compare_icons_name (  
    Icon * icon1,  
    Icon * icon2 )
```

Compare 2 icons with the order relation on the name (case unsensitive alphabetical order).

Parameters

<i>icon1</i>	an icon.
<i>icon2</i>	another icon.

Returns

-1 if $icon1 < icon2$, 1 if $icon1 > icon2$, 0 if $icon1 = icon2$.

5.28.3.4 cairo_dock_compare_icons_extension()

```
int cairo_dock_compare_icons_extension (  
    Icon * icon1,  
    Icon * icon2 )
```

Compare 2 icons with the order relation on the extension of their URIs (case unsensitive alphabetical order).

Parameters

<i>icon1</i>	an icon.
<i>icon2</i>	another icon.

Returns

-1 if $icon1 < icon2$, 1 if $icon1 > icon2$, 0 if $icon1 = icon2$.

5.28.3.5 cairo_dock_sort_icons_by_order()

```
GList * cairo_dock_sort_icons_by_order (  
    GList * pIconList )
```

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.

5.28.3.6 cairo_dock_sort_icons_by_name()

```
GList * cairo_dock_sort_icons_by_name (
    GList * pIconList )
```

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.

5.28.3.7 cairo_dock_get_first_icon()

```
Icon * cairo_dock_get_first_icon (
    GList * pIconList )
```

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.

5.28.3.8 cairo_dock_get_last_icon()

```
Icon * cairo_dock_get_last_icon (
    GList * pIconList )
```

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.

5.28.3.9 cairo_dock_get_first_icon_of_group()

```
Icon * cairo_dock_get_first_icon_of_group (
    GList * pIconList,
    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.

5.28.3.10 cairo_dock_get_last_icon_of_group()

```
Icon * cairo_dock_get_last_icon_of_group (
    GList * pIconList,
    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.

5.28.3.11 cairo_dock_get_first_icon_of_order()

```
Icon * cairo_dock_get_first_icon_of_order (
    GList * pIconList,
    CairoDockIconGroup iGroup )
```

Get the first icon whose group has the same order as a given one.

Parameters

<i>plconList</i>	a list of icons.
<i>iGroup</i>	a group of icon.

Returns

the first found icon, or NULL if none matches.

5.28.3.12 cairo_dock_get_last_icon_of_order()

```
Icon * cairo_dock_get_last_icon_of_order (
    GList * pIconList,
    CairoDockIconGroup iGroup )
```

Get the last icon whose group has the same order as a given one.

Parameters

<i>plconList</i>	a list of icons.
<i>iGroup</i>	a group of icon.

Returns

the last found icon, or NULL if none matches.

5.28.3.13 cairo_dock_get_pointed_icon()

```
Icon * cairo_dock_get_pointed_icon (
    GList * pIconList )
```

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.

5.28.3.14 cairo_dock_get_next_icon()

```
Icon * cairo_dock_get_next_icon (
    GList * pIconList,
    Icon * pIcon )
```

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.

5.28.3.15 cairo_dock_get_previous_icon()

```
Icon * cairo_dock_get_previous_icon (
    GList * pIconList,
    Icon * pIcon )
```

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.

5.28.3.16 cairo_dock_get_icon_with_command()

```
Icon * cairo_dock_get_icon_with_command (
    GList * pIconList,
    const gchar * cCommand )
```

Search an icon with a given command in a list of icons.

Parameters

<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.

5.28.3.17 cairo_dock_get_icon_with_base_uri()

```
Icon * cairo_dock_get_icon_with_base_uri (
    GList * pIconList,
    const gchar * cBaseURI )
```


Search an icon with a given URI in a list of icons.

Parameters

<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.

5.28.3.18 cairo_dock_get_icon_with_name()

```
Icon * cairo_dock_get_icon_with_name (
    GList * pIconList,
    const gchar * cName )
```

Search an icon with a given name in a list of icons.

Parameters

<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.

5.28.3.19 cairo_dock_get_icon_with_subdock()

```
Icon * cairo_dock_get_icon_with_subdock (
    GList * pIconList,
    CairoDock * pSubDock )
```

Search the icon pointing on a given sub-dock in a list of icons.

Parameters

<i>plconList</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.

5.28.3.20 cairo_dock_get_icon_extent()

```
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>plcon</i>	the icon.
<i>iWidth</i>	pointer to the width.
<i>iHeight</i>	pointer to the height.

5.28.3.21 cairo_dock_get_current_icon_size()

```

void cairo_dock_get_current_icon_size (
    Icon * pIcon,
    GldiContainer * pContainer,
    double * fSizeX,
    double * fSizeY )

```

Get the current size of an icon as it is seen on the screen (taking into account the zoom and the ratio).

Parameters

<i>plcon</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)

5.28.3.22 cairo_dock_compute_icon_area()

```

void cairo_dock_compute_icon_area (
    Icon * icon,
    GldiContainer * pContainer,
    GdkRectangle * pArea )

```

Get the total zone used by an icon on its container (taking into account reflect, gap to reflect, zoom and stretching).

Parameters

<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.

5.28.3.23 gldi_icon_set_name()

```

void gldi_icon_set_name (

```

```
Icon * pIcon,  
const gchar * cIconName )
```

Set the label of an icon. If it has a sub-dock, it is renamed (the name is possibly altered to stay unique). The label buffer is updated too.

Parameters

<i>plcon</i>	the icon.
<i>clconName</i>	the new label of the icon. You can even pass <i>plcon->cName</i> .

5.28.3.24 gldi_icon_set_name_printf()

```
void gldi_icon_set_name_printf (  
    Icon * pIcon,  
    const gchar * cIconNameFormat,  
    ... )
```

Same as above, but takes a printf-like format string.

Parameters

<i>plcon</i>	the icon.
<i>clconNameFormat</i>	the new label of the icon, in a 'printf' way.
...	data to be inserted into the string.

5.28.3.25 gldi_icon_set_quick_info()

```
void void gldi_icon_set_quick_info (  
    Icon * pIcon,  
    const gchar * cQuickInfo )
```

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

Parameters

<i>plcon</i>	the icon.
<i>cQuickInfo</i>	the text of the quick-info. If NULL, will just remove the current the quick-info.

5.28.3.26 gldi_icon_set_quick_info_printf()

```
void gldi_icon_set_quick_info_printf (  
    Icon * pIcon,  
    const gchar * cQuickInfoFormat,  
    ... )
```

Same as above, but takes a printf-like format string.

Parameters

<i>plcon</i>	the icon.
<i>cQuickInfoFormat</i>	the text of the quick-info, in a 'printf' way.
...	data to be inserted into the string.

5.28.3.27 cairo_dock_icon_buffer_to_cairo()

```
cairo_surface_t * cairo_dock_icon_buffer_to_cairo (
    Icon * icon,
    int iWidth,
    int iHeight )
```

Create a copy of the icon's image, scaled to the desired size, preserving the correct device scale.

5.28.3.28 cairo_dock_begin_draw_icon()

```
gboolean cairo_dock_begin_draw_icon (
    Icon * pIcon,
    gint iRenderingMode )
```

Initiate an OpenGL drawing session on an icon's texture.

Parameters

<i>plcon</i>	the icon on which to draw.
<i>iRenderingMode</i>	rendering mode. 0:normal, 1:don't clear the current texture, so that the drawing will be superimposed on it, 2:keep the current icon texture unchanged for all the drawing (the drawing is made on another texture).

Returns

TRUE if you can proceed to the drawing, FALSE if an error occurred.

5.28.3.29 cairo_dock_end_draw_icon()

```
void cairo_dock_end_draw_icon (
    Icon * pIcon )
```

Finish an OpenGL drawing session on an icon.

Parameters

<i>plcon</i>	the icon on which to draw.
--------------	----------------------------

Returns

TRUE if you can proceed to the drawing, FALSE if an error occurred.

5.29 cairo-dock-icon-factory.h File Reference

Data Structures

- struct [_IconInterface](#)
Icon's interface.
- struct [_Icon](#)
Definition of an Icon.
- struct [_CairoIconContainerRenderer](#)
Definition of an Icon container (= an icon holding a sub-dock) renderer.

Macros

- #define [CAIRO_DOCK_IS_ICON](#)(obj)
- #define [CAIRO_DOCK_IS_APPLI](#)(icon)
- #define [CAIRO_DOCK_IS_APPLET](#)(icon)
- #define [CAIRO_DOCK_IS_MULTI_APPLI](#)(icon)
- #define [CAIRO_DOCK_IS_AUTOMATIC_SEPARATOR](#)(icon)
- #define [CAIRO_DOCK_IS_USER_SEPARATOR](#)(icon)
- #define [CAIRO_DOCK_IS_NORMAL_APPLI](#)(icon)
- #define [CAIRO_DOCK_IS_DETACHABLE_APPLET](#)(icon)

Enumerations

- enum [CairoDockIconGroup](#)
Available groups of icons.
- enum [CairoDockAnimationState](#)
Animation state of an icon, sorted by priority.

Functions

- [Icon](#) * [gldi_icon_new](#) (void)
- [Icon](#) * [cairo_dock_create_dummy_launcher](#) (gchar *cName, gchar *cFileName, gchar *cCommand, gchar *cQuickInfo, double fOrder)
- void [cairo_dock_load_icon_image](#) ([Icon](#) *icon, [GldiContainer](#) *pContainer)
- void [cairo_dock_load_icon_text](#) ([Icon](#) *icon)
- void [cairo_dock_load_icon_quickinfo](#) ([Icon](#) *icon)
- void [cairo_dock_load_icon_buffers](#) ([Icon](#) *pIcon, [GldiContainer](#) *pContainer)

5.29.1 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.

5.29.2 Macro Definition Documentation

5.29.2.1 CAIRO_DOCK_IS_ICON

```
#define CAIRO_DOCK_IS_ICON(  
    obj )
```

Say if an object is an Icon.

Parameters

<i>obj</i>	the object.
------------	-------------

Returns

TRUE if the object is an icon.

5.29.2.2 CAIRO_DOCK_IS_APPLI

```
#define CAIRO_DOCK_IS_APPLI(  
    icon )
```

TRUE if the icon holds a window.

Parameters

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

5.29.2.3 CAIRO_DOCK_IS_APPLET

```
#define CAIRO_DOCK_IS_APPLET(  
    icon )
```

TRUE if the icon holds an instance of a module.

Parameters

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

5.29.2.4 CAIRO_DOCK_IS_MULTI_APPLI

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

5.29.2.5 CAIRO_DOCK_IS_AUTOMATIC_SEPARATOR

```
#define CAIRO_DOCK_IS_AUTOMATIC_SEPARATOR(  
    icon )
```

TRUE if the icon is an automatic separator.

Parameters

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

5.29.2.6 CAIRO_DOCK_IS_USER_SEPARATOR

```
#define CAIRO_DOCK_IS_USER_SEPARATOR(  
    icon )
```

TRUE if the icon is a separator added by the user.

Parameters

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

5.29.2.7 CAIRO_DOCK_IS_NORMAL_APPLI

```
#define CAIRO_DOCK_IS_NORMAL_APPLI(  
    icon )
```

TRUE if the icon is an icon d'appli only.

Parameters

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

5.29.2.8 CAIRO_DOCK_IS_DETACHABLE_APPLET

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

5.29.3 Function Documentation

5.29.3.1 gldi_icon_new()

```
Icon * gldi_icon_new (  
    void )
```

Create an empty icon.

Returns

the newly allocated icon object.

5.29.3.2 cairo_dock_create_dummy_launcher()

```
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.

5.29.3.3 cairo_dock_load_icon_image()

```
void cairo_dock_load_icon_image (
    Icon * icon,
    GldiContainer * pContainer )
```

Fill the image buffer (surface & texture) of a given icon, according to its type. Set its size if necessary, and fills the reflection buffer for cairo.

Parameters

<i>icon</i>	the icon.
<i>pContainer</i>	its container.

5.29.3.4 cairo_dock_load_icon_text()

```
void cairo_dock_load_icon_text (
    Icon * icon )
```

Fill the label buffer (surface & texture) of a given icon, according to a text description.

Parameters

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

5.29.3.5 cairo_dock_load_icon_quickinfo()

```
void cairo_dock_load_icon_quickinfo (
    Icon * icon )
```

Fill the quick-info buffer (surface & texture) of a given icon, according to a text description.

Parameters

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

5.29.3.6 cairo_dock_load_icon_buffers()

```
void cairo_dock_load_icon_buffers (
    Icon * pIcon,
    GldiContainer * pContainer )
```

Fill all the buffers (surfaces & textures) of a given icon, according to its type. Set its size accordingly, and fills the reflection buffer for cairo. Label and quick-info are loaded with the current global text description.

Parameters

<i>pIcon</i>	the icon.
<i>pContainer</i>	its container.

5.30 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](#) ,
[NB_NOTIFICATIONS_ICON](#) }
signals

Functions

- void [gldi_icons_foreach](#) (GldiIconFunc pFunction, gpointer pUserData)
- gint [cairo_dock_search_icon_size](#) (GtkIconSize iIconSize)
- gchar * [cairo_dock_search_icon_s_path](#) (const gchar *cFileName, gint iDesiredIconSize)

5.30.1 Detailed Description

This class manages the icons parameters and their associated ressources.

Specialized Icons are handled by the corresponding manager.

5.30.2 Enumeration Type Documentation

5.30.2.1 CairoIconNotifications

```
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.

5.30.3 Function Documentation

5.30.3.1 gldi_icons_foreach()

```
void gldi_icons_foreach (
    GldiIconFunc pFunction,
    gpointer pUserData )
```

Execute an action on all icons.

Parameters

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

5.30.3.2 cairo_dock_search_icon_size()

```
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)

5.30.3.3 cairo_dock_search_icon_s_path()

```
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.

5.31 cairo-dock-image-buffer.h File Reference

Data Structures

- struct [_CairoDockImageBuffer](#)

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

Macros

- #define [cairo_dock_load_image_buffer](#)(pImage, cImageFile, iWidth, iHeight, iLoadModifier)
- #define [cairo_dock_apply_image_buffer_surface](#)(pImage, pCairoContext)
- #define [cairo_dock_apply_image_buffer_texture](#)(pImage)

Functions

- gchar * [cairo_dock_search_image_s_path](#) (const gchar *cImageFile)
- void [cairo_dock_load_image_buffer_full](#) ([CairoDockImageBuffer](#) *pImage, const gchar *cImageFile, int 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)
- void [cairo_dock_apply_image_buffer_surface_with_offset](#) (const [CairoDockImageBuffer](#) *pImage, cairo_t *pCairoContext, double x, double y, double fAlpha)
- void [cairo_dock_apply_image_buffer_texture_with_offset](#) (const [CairoDockImageBuffer](#) *pImage, double x, double y)
- void [cairo_dock_apply_image_buffer_surface_at_size](#) (const [CairoDockImageBuffer](#) *pImage, cairo_t *pCairoContext, int w, int h, double x, double y, double fAlpha)
- void [cairo_dock_apply_image_buffer_texture_at_size](#) (const [CairoDockImageBuffer](#) *pImage, int w, int h, double x, double y)
- void [cairo_dock_create_icon_fbo](#) (void)
- void [cairo_dock_destroy_icon_fbo](#) (void)
- cairo_surface_t * [cairo_dock_image_buffer_copy_scale](#) ([CairoDockImageBuffer](#) *pImage, int iWidth, int iHeight)

5.31.1 Detailed Description

This class defines a generic image API that works for both Cairo and OpenGL. It allows to easily load and display images, without having to care the rendering mode. It supports animated images (an animated image is made of several frames, ordered side by side from left to right).

Use [cairo_dock_create_image_buffer](#) to create an image buffer from a file, or [cairo_dock_load_image_buffer](#) to load an image into an existing image buffer. Use [cairo_dock_free_image_buffer](#) to destroy it or [cairo_dock_unload_image_buffer](#) to unload and reset it to 0.

Use [cairo_dock_apply_image_buffer_surface](#) or [cairo_dock_apply_image_buffer_texture](#) to display the image.

5.31.2 Macro Definition Documentation

5.31.2.1 [cairo_dock_load_image_buffer](#)

```
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

5.31.2.2 [cairo_dock_apply_image_buffer_surface](#)

```
#define cairo_dock_apply_image_buffer_surface(  
    pImage,  
    pCairoContext )
```

Draw an ImageBuffer on a cairo context.

Parameters

<i>pImage</i>	an ImageBuffer.
<i>pCairoContext</i>	the current cairo context.

5.31.2.3 cairo_dock_apply_image_buffer_texture

```
#define cairo_dock_apply_image_buffer_texture(  
    pImage )
```

Draw an ImageBuffer on the current OpenGL context.

Parameters

<i>pImage</i>	an ImageBuffer.
---------------	-----------------

5.31.3 Function Documentation

5.31.3.1 cairo_dock_search_image_s_path()

```
gchar * cairo_dock_search_image_s_path (  
    const gchar * cImageFile )
```

Find the path of an image. '~' is handled, as well as the 'images' folder of the current theme. Use [cairo_dock_search_icon_s_path](#) to search theme icons.

Parameters

<i>cImageFile</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.

5.31.3.2 cairo_dock_load_image_buffer_full()

```
void cairo_dock_load_image_buffer_full (  
    CairoDockImageBuffer * pImage,  
    const gchar * cImageFile,  
    int iWidth,  
    int iHeight,  
    CairoDockLoadImageModifier iLoadModifier,  
    double fAlpha )
```

Load an image into an ImageBuffer with a given transparency. If the image is given by its sole name, it is taken in the root folder of the current theme.

Parameters

<i>pImage</i>	an ImageBuffer.
<i>cImageFile</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)

5.31.3.3 cairo_dock_load_image_buffer_from_surface()

```
void cairo_dock_load_image_buffer_from_surface (
    CairoDockImageBuffer * pImage,
    cairo_surface_t * pSurface,
    int iWidth,
    int iHeight )
```

Load a surface into an ImageBuffer.

Parameters

<i>pImage</i>	an ImageBuffer.
<i>pSurface</i>	a cairo surface
<i>iWidth</i>	width of the surface
<i>iHeight</i>	height of the surface

5.31.3.4 cairo_dock_create_image_buffer()

```
CairoDockImageBuffer * cairo_dock_create_image_buffer (
    const gchar * cImageFile,
    int iWidth,
    int iHeight,
    CairoDockLoadImageModifier iLoadModifier )
```

Create and load an image into an ImageBuffer. If the image is given by its sole name, it is taken in the root folder of the current theme.

Parameters

<i>cImageFile</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.

5.31.3.5 cairo_dock_unload_image_buffer()

```
void cairo_dock_unload_image_buffer (
    CairoDockImageBuffer * pImage )
```

Reset an ImageBuffer's ressources. It can be used to load another image then.

Parameters

<i>pImage</i>	an ImageBuffer.
---------------	-----------------

5.31.3.6 cairo_dock_free_image_buffer()

```
void cairo_dock_free_image_buffer (
    CairoDockImageBuffer * pImage )
```

Reset and free an ImageBuffer.

Parameters

<i>pImage</i>	an ImageBuffer.
---------------	-----------------

5.31.3.7 cairo_dock_apply_image_buffer_surface_with_offset()

```
void cairo_dock_apply_image_buffer_surface_with_offset (
    const CairoDockImageBuffer * pImage,
    cairo_t * pCairoContext,
    double x,
    double y,
    double fAlpha )
```

Draw an ImageBuffer with an offset on a Cairo context, at the size it was loaded.

Parameters

<i>pImage</i>	an ImageBuffer.
<i>pCairoContext</i>	the current cairo context.
<i>x</i>	horizontal offset.
<i>y</i>	vertical offset.
<i>fAlpha</i>	transparency (in [0;1])

5.31.3.8 cairo_dock_apply_image_buffer_texture_with_offset()

```
void cairo_dock_apply_image_buffer_texture_with_offset (
    const CairoDockImageBuffer * pImage,
    double x,
    double y )
```

Draw an ImageBuffer with an offset on the current OpenGL context, at the size it was loaded.

Parameters

<i>pImage</i>	an ImageBuffer.
<i>x</i>	horizontal offset.
<i>y</i>	vertical offset.

5.31.3.9 cairo_dock_apply_image_buffer_surface_at_size()

```
void cairo_dock_apply_image_buffer_surface_at_size (
```



```

    const CairoDockImageBuffer * pImage,
    cairo_t * pCairoContext,
    int w,
    int h,
    double x,
    double y,
    double fAlpha )

```

Draw an ImageBuffer with an offset on a Cairo context, at a given size.

Parameters

<i>pImage</i>	an ImageBuffer.
<i>pCairoContext</i>	the current cairo context.
<i>w</i>	requested width
<i>h</i>	requested height
<i>x</i>	horizontal offset.
<i>y</i>	vertical offset.
<i>fAlpha</i>	transparency (in [0;1])

5.31.3.10 cairo_dock_apply_image_buffer_texture_at_size()

```

void cairo_dock_apply_image_buffer_texture_at_size (
    const CairoDockImageBuffer * pImage,
    int w,
    int h,
    double x,
    double y )

```

Draw an ImageBuffer on the current OpenGL context at a given size.

Parameters

<i>pImage</i>	an ImageBuffer.
<i>w</i>	requested width
<i>h</i>	requested height
<i>x</i>	horizontal offset.
<i>y</i>	vertical offset.

5.31.3.11 cairo_dock_create_icon_fbo()

```

void cairo_dock_create_icon_fbo (
    void )

```

Create an FBO to render the icons inside a dock.

5.31.3.12 cairo_dock_destroy_icon_fbo()

```

void cairo_dock_destroy_icon_fbo (
    void )

```

Destroy the icons FBO.

5.31.3.13 `cairo_dock_image_buffer_copy_scale()`

```
cairo_surface_t * cairo_dock_image_buffer_copy_scale (
    CairoDockImageBuffer * pImage,
    int iWidth,
    int iHeight )
```

Create a scaled copy of an image as Cairo surface suitable for e.g. using in menus.

5.32 `cairo-dock-indicator-manager.h` File Reference

5.32.1 Detailed Description

This class manages the indicators.

5.33 `cairo-dock-keybinder.h` File Reference

Macros

- `#define gldi_shortkey_could_grab(binding)`

Typedefs

- `typedef void(* CDBindkeyHandler) (const gchar *keystring, gpointer user_data)`
Definition of a callback, called when a shortcut is pressed by the user.

Functions

- `GldiShortcut * gldi_shortkey_new (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)`
- `gboolean gldi_shortkey_rebind (GldiShortcut *binding, const gchar *cNewKeyString, const gchar *cNewDescription)`
- `gboolean cairo_dock_trigger_shortkey (const gchar *cKeyString)`

5.33.1 Detailed Description

This class defines the Shortkeys, which are objects that bind a keyboard shortcut to an action. The keyboard shortcut is defined globally on the desktop, that is to say they will be effective whatever window has the focus. Keyboard shortcuts are of the form <alt>F1 or <ctrl><shift>s.

Use [gldi_shortkey_new](#) to create a new shortcut, and simply unref it with [gldi_object_unref](#) to unbind the keyboard shortcut. To update a binding (whenever the shortcut or the description change, or just to re-grab it), use [gldi_shortkey_rebind](#).

5.33.2 Macro Definition Documentation

5.33.2.1 `gldi_shortkey_could_grab`

```
#define gldi\_shortkey\_could\_grab(
    binding )
```

Says if the shortcut of a key binding could be grabbed.

Parameters

<i>binding</i>	a key binding.
----------------	----------------

Returns

TRUE iif the shortcut has been successfully grabbed by the key binding.

5.33.3 Function Documentation

5.33.3.1 gldi_shortkey_new()

```
GldiShortkey * gldi_shortkey_new (
    const gchar * keystring,
    const gchar * cDemander,
    const gchar * cDescription,
    const gchar * cIconFilePath,
    const gchar * cConfFilePath,
    const gchar * cGroupName,
    const gchar * cKeyName,
    CDBindkeyHandler handler,
    gpointer user_data )
```

Create a new shortcut, that binds an action to a shortcut. Unref 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>cIconFilePath</i>	an icon that represents the demander
<i>cConfFilePath</i>	conf file where the shortcut is 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 shortcut, already bound.

5.33.3.2 gldi_shortkey_rebind()

```
gboolean gldi_shortkey_rebind (
    GldiShortkey * binding,
    const gchar * cNewKeyString,
    const gchar * cNewDescription )
```

Rebind a shortcut to a new one. If the shortcut is the same, don't re-bind it.

Parameters

<i>binding</i>	a key binding.
<i>cNewKeyString</i>	the new shortcutkey
<i>cNewDescription</i>	the new description, or NULL to keep the current one.

Returns

TRUE on success

5.33.3.3 cairo_dock_trigger_shortkey()

```
gboolean cairo_dock_trigger_shortkey (
    const gchar * cKeyString )
```

Trigger a 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.

5.34 cairo-dock-keyfile-utilities.h File Reference

Functions

- GKeyFile * [cairo_dock_open_key_file](#) (const gchar *cConfFilePath)
- void [cairo_dock_write_keys_to_file_full](#) (GKeyFile *pKeyFile, const gchar *cConfFilePath, gboolean bAllow↵ Empty)
- gchar * [cairo_dock_write_keys_to_new_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 *cIniValue, [CairoDockGUIWidgetType](#) iWidgetType, const gchar *cAuthorized↵ Values, const gchar *cDescription, const gchar *cTooltip)
- void [cairo_dock_remove_group_key_from_conf_file](#) (GKeyFile *pKeyFile, const gchar *cGroupName, const gchar *cKeyName)
- void [cairo_dock_update_keyfile](#) (const gchar *cConfFilePath, GType iFirstDataType,...)

5.34.1 Detailed Description

This class provides useful functions to manipulate the conf files of Cairo-Dock, which are classic group/key pair files.

5.34.2 Function Documentation

5.34.2.1 cairo_dock_open_key_file()

```
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.

5.34.2.2 cairo_dock_write_keys_to_file_full()

```
void cairo_dock_write_keys_to_file_full (
    GKeyFile * pKeyFile,
    const gchar * cConfFilePath,
    gboolean bAllowEmpty )
```

Write a key file on the disk.

5.34.2.3 cairo_dock_write_keys_to_new_file()

```
gchar * cairo_dock_write_keys_to_new_file (
    GKeyFile * pKeyFile,
    const gchar * cConfFilePath )
```

Write a key file on the disk.

Parameters

<i>pKeyFile</i>	the key-file
<i>cConfFilePath</i>	its path on the disk

Returns

The name of the new config file in a newly allocated string, or NULL on failure. If *cConfFilePath* does not exist, a new file is created using this path. If *cConfFilePath* already exists, a new filename is generated by using it as a template and adding a unique suffix, and the keyfile is written there instead.

5.34.2.4 cairo_dock_merge_conf_files()

```
void cairo_dock_merge_conf_files (
    const gchar * cConfFilePath,
```

```
gchar * cReplacementConfFilePath,  
gchar iIdentifier )
```

Merge the values of a conf-file into another one. Keys are filtered by an identifier on the original conf-file.

Parameters

<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>iIdentifier</i>	a character to filter the keys, or 0.

5.34.2.5 cairo_dock_upgrade_conf_file_full()

```
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.

5.34.2.6 cairo_dock_get_conf_file_version()

```
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.

5.34.2.7 cairo_dock_conf_file_needs_update()

```
gboolean cairo_dock_conf_file_needs_update (
    GKeyFile * pKeyFile,
    const gchar * cVersion )
```

Say if a conf file's version mismatches a given version.

5.34.2.8 cairo_dock_add_remove_element_to_key()

```
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.

5.34.2.9 cairo_dock_add_group_key_to_conf_file()

```
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.

5.34.2.10 cairo_dock_remove_group_key_from_conf_file()

```
void cairo_dock_remove_group_key_from_conf_file (
    GKeyFile * pKeyFile,
    const gchar * cGroupName,
    const gchar * ckeyName )
```

Remove a key from a conf file.

5.34.2.11 cairo_dock_update_keyfile()

```
void cairo_dock_update_keyfile (
    const gchar * cConfFilePath,
    GType iFirstDataType,
    ... )
```

Update a conf file with a list of values of the form : {type, name of the groupe, name of the key, value}. Must end with G_TYPE_INVALID.

Parameters

<i>cConfFilePath</i>	path to the conf file.
<i>iFirstDataType</i>	type of the first value.

5.35 cairo-dock-launcher-manager.h File Reference

Macros

- #define [GLDI_OBJECT_IS_LAUNCHER_ICON](#)(obj)

Functions

- [Icon](#) * [gldi_launcher_add_new_full](#) (const gchar *cURI, [CairoDock](#) *pDock, double fOrder, gboolean b↵ Validate)
- [Icon](#) * [gldi_launcher_add_new](#) (const gchar *cURI, [CairoDock](#) *pDock, double fOrder)

5.35.1 Detailed Description

This class handles the Launcher Icons, which are user icons used to launch a program.

5.35.2 Macro Definition Documentation

5.35.2.1 GLDI_OBJECT_IS_LAUNCHER_ICON

```
#define GLDI_OBJECT_IS_LAUNCHER_ICON(  
    obj )
```

Say if an object is a LauncherIcon.

Parameters

<i>obj</i>	the object.
------------	-------------

Returns

TRUE if the object is a LauncherIcon.

5.35.3 Function Documentation

5.35.3.1 gldi_launcher_add_new_full()

```
Icon * gldi_launcher_add_new_full (  
    const gchar * cURI,  
    CairoDock * pDock,  
    double fOrder,  
    gboolean bValidate )
```

Add a new launcher from a .desktop file or a blank one.

Parameters

<i>cURI</i>	absolute path of the .desktop file to add, a "application://" URI or NULL to create a blank launcher
<i>pDock</i>	the dock to add the launcher to
<i>fOrder</i>	where to add the launcher
<i>bValidate</i>	whether to check if the .desktop file supplied as cURI actually exists. If this is TRUE and the .desktop file cannot be found, this function will fail.

Returns

the icon corresponding to the new launcher

5.35.3.2 gldi_launcher_add_new()

```
Icon * gldi_launcher_add_new (  
    const gchar * cURI,
```

```
CairoDock * pDock,
double fOrder )
```

Add a new launcher from a .desktop file or a blank one. This is the same as `gldi_launcher_add_new_full ()` with `bValidate == FALSE`.

5.36 cairo-dock-manager.h File Reference

Data Structures

- struct [_GldiManager](#)
Definition of a Manager.

Macros

- #define [GLDI_OBJECT_IS_MANAGER\(obj\)](#)

5.36.1 Detailed Description

This class defines the Managers. A Manager is like an internal module: it has a classic module interface, manages a set of resources, and has its own configuration.

Each manager is initialized at the beginning. When loading the current theme, `get_config` and `load` are called. When unloading the current theme, `unload` and `reset_config` are called. When reloading a part of the current theme, `reset_config`, `get_config` and `load` are called.

5.36.2 Macro Definition Documentation

5.36.2.1 GLDI_OBJECT_IS_MANAGER

```
#define GLDI_OBJECT_IS_MANAGER(  
    obj )
```

Say if an object is a Manager.

Parameters

<i>obj</i>	the object.
------------	-------------

Returns

TRUE if the object is a Manager.

5.37 cairo-dock-menu.h File Reference

Macros

- #define [gldi_submenu_new\(...\)](#)

- #define [gldi_menu_item_new](#)(cLabel, cImage)
- #define [gldi_menu_add_sub_menu](#)(pMenu, cLabel, cImage)

Functions

- GtkWidget * [gldi_menu_new](#) (Icon *plcon)
- void [gldi_menu_init](#) (GtkWidget *pMenu, Icon *plcon)
- void [gldi_menu_popup_full](#) (GtkWidget *menu, const GdkEvent *event)
- GtkWidget * [gldi_menu_item_new_full2](#) (const gchar *cLabel, const gchar *cImage, gboolean bUseMnemonic, GtkIconSize iSize, gboolean bUseStyle)
- GtkWidget * [gldi_menu_item_new_with_action](#) (const gchar *cLabel, const gchar *cImage, GCallback pFunction, gpointer pData)
- GtkWidget * [gldi_menu_item_new_with_submenu](#) (const gchar *cLabel, const gchar *cImage, GtkWidget **pSubMenuPtr)
- void [gldi_menu_item_set_image](#) (GtkWidget *pMenuItem, GtkWidget *image)
- GtkWidget * [gldi_menu_item_get_image](#) (GtkWidget *pMenuItem)
- GtkWidget * [gldi_menu_add_item](#) (GtkWidget *pMenu, const gchar *cLabel, const gchar *cImage, GCallback pFunction, gpointer pData)
- GtkWidget * [gldi_menu_add_item_with_tooltip](#) (GtkWidget *pMenu, const gchar *cLabel, const gchar *cImage, const gchar *cToolTip, void(*pFunction)(GtkMenuItem *, gpointer), gpointer pData)
- GtkWidget * [gldi_menu_add_sub_menu_full](#) (GtkWidget *pMenu, const gchar *cLabel, const gchar *cImage, GtkWidget **pMenuItemPtr)
- void [gldi_menu_add_separator](#) (GtkWidget *pMenu)

5.37.1 Detailed Description

This class defines the Menu. They are classical menus, but with a custom looking.

5.37.2 Macro Definition Documentation

5.37.2.1 gldi_submenu_new

```
#define gldi_submenu_new(
    ... )
```

Creates a new sub-menu. It's just a menu that doesn't point on an Icon/Container.

5.37.2.2 gldi_menu_item_new

```
#define gldi_menu_item_new(
    cLabel,
    cImage )
```

A convenient function to create a menu-item with a label and an image.

Parameters

<i>cLabel</i>	the label, or NULL
<i>cImage</i>	the image path or name, or NULL

Returns

the new menu-item.

5.37.2.3 gldi_menu_add_sub_menu

```
#define gldi_menu_add_sub_menu(  
    pMenu,  
    cLabel,  
    cImage )
```

A convenient function to add a sub-menu to a given menu.

Parameters

<i>pMenu</i>	the menu
<i>cLabel</i>	the label, or NULL
<i>cImage</i>	the image path or name, or NULL

Returns

the new sub-menu that has been added.

5.37.3 Function Documentation**5.37.3.1 gldi_menu_new()**

```
GtkWidget * gldi_menu_new (  
    Icon * pIcon )
```

Creates a new menu that will point on a given icon. If the icon is NULL, it will be placed under the mouse.

Parameters

<i>pIcon</i>	the icon, or NULL
--------------	-------------------

Returns

the new menu.

5.37.3.2 gldi_menu_init()

```
void gldi_menu_init (  
    GtkWidget * pMenu,  
    Icon * pIcon )
```

Initialize a menu, so that it can be drawn and placed correctly. It's useful if the menu was created beforehand (like a `DBusMenu`).

Parameters

<i>plcon</i>	the icon, or NULL
--------------	-------------------

5.37.3.3 gldi_menu_popup_full()

```
void gldi_menu_popup_full (
    GtkWidget * menu,
    const GdkEvent * event )
```

Pop-up a menu. The menu is placed above the icon, or above the container, or above the mouse, depending on how it has been initialized.

Parameters

<i>menu</i>	the menu.
<i>event</i>	an event to which the menu is popped up in response (NULL to use the current GTK event)

5.37.3.4 gldi_menu_item_new_full2()

```
GtkWidget * gldi_menu_item_new_full2 (
    const gchar * cLabel,
    const gchar * cImage,
    gboolean bUseMnemonic,
    GtkIconSize iSize,
    gboolean bUseStyle )
```

Creates a menu-item, with a label and an image. The child widget of the menu-item is a gtk-label. If the label is NULL, the child widget will be NULL too (this is useful if the menu-item will hold a custom widget).

Parameters

<i>cLabel</i>	the label, or NULL
<i>cImage</i>	the image path or name, or NULL
<i>bUseMnemonic</i>	whether to use the mnemonic inside the label or not
<i>iSize</i>	size of the image, or 0 to use the default size
<i>bUseStyle</i>	whether to use our custom style to draw this menu item

Returns

the new menu-item.

5.37.3.5 gldi_menu_item_new_with_action()

```
GtkWidget * gldi_menu_item_new_with_action (
    const gchar * cLabel,
    const gchar * cImage,
```

```
GCallback pFunction,  
gpointer pData )
```

A convenient function to create a menu-item with a label, an image, and an associated action.

Parameters

<i>cLabel</i>	the label, or NULL
<i>cImage</i>	the image path or name, or NULL
<i>pFunction</i>	the callback
<i>pData</i>	the data passed to the callback

Returns

the new menu-item.

5.37.3.6 `gldi_menu_item_new_with_submenu()`

```
GtkWidget * gldi_menu_item_new_with_submenu (   
    const gchar * cLabel,  
    const gchar * cImage,  
    GtkWidget ** pSubMenuPtr )
```

A convenient function to create a menu-item with a label, an image, and an associated sub-menu.

Parameters

<i>cLabel</i>	the label
<i>cImage</i>	the image path or name, or NULL
<i>pSubMenuPtr</i>	pointer that will contain the new sub-menu, or NULL

Returns

the new menu-item.

5.37.3.7 `gldi_menu_item_set_image()`

```
void gldi_menu_item_set_image (   
    GtkWidget * pMenuItem,  
    GtkWidget * image )
```

Sets a gtk-image on a menu-item. This is useful if the image can't be given by a name or path (for instance, loaded from a cairo surface).

Parameters

<i>pMenuItem</i>	the menu-item
<i>image</i>	the image

5.37.3.8 gldi_menu_item_get_image()

```
GtkWidget * gldi_menu_item_get_image (
    GtkWidget * pMenuItem )
```

Gets the image of a menu-item.

Parameters

<i>pMenuItem</i>	the menu-item
------------------	---------------

Returns

the gtk-image

5.37.3.9 gldi_menu_add_item()

```
GtkWidget * gldi_menu_add_item (
    GtkWidget * pMenu,
    const gchar * cLabel,
    const gchar * cImage,
    GCallback pFunction,
    gpointer pData )
```

A convenient function to add an item to a given menu.

Parameters

<i>pMenu</i>	the menu
<i>cLabel</i>	the label, or NULL
<i>cImage</i>	the image path or name, or NULL
<i>pFunction</i>	the callback
<i>pData</i>	the data passed to the callback

Returns

the new menu-entry that has been added.

5.37.3.10 gldi_menu_add_item_with_tooltip()

```
GtkWidget * gldi_menu_add_item_with_tooltip (
    GtkWidget * pMenu,
    const gchar * cLabel,
    const gchar * cImage,
    const gchar * cToolTip,
    void(*) (GtkMenuItem *, gpointer) pFunction,
    gpointer pData )
```

A convenient function to add an item to a given menu with an optional tooltip to display. It is recommended to use this function to add a tooltip instead of `gtk_widget_set_tooltip_text ()` as on Wayland and `gtk-layer-shell` there seems to be a race condition with GTK internals that can result in an attempt to re-show the tooltip after the menu has been closed, that leads to a protocol error and crash; see <https://github.com/wmww/gtk-layer-shell/issues/207>. This function takes care to keep the tooltip hidden when the menu has been closed.

Parameters

<i>pMenu</i>	the menu
<i>cLabel</i>	the label, or NULL
<i>cImage</i>	the image path or name, or NULL
<i>cToolTip</i>	the tooltip to display when the user hovers on the menu item or NULL
<i>pFunction</i>	the callback
<i>pData</i>	the data passed to the callback

Returns

the new menu-entry that has been added.

5.37.3.11 `gldi_menu_add_sub_menu_full()`

```
GtkWidget * gldi_menu_add_sub_menu_full (
    GtkWidget * pMenu,
    const gchar * cLabel,
    const gchar * cImage,
    GtkWidget ** pMenuItemPtr )
```

A convenient function to add a sub-menu to a given menu.

Parameters

<i>pMenu</i>	the menu
<i>cLabel</i>	the label, or NULL
<i>cImage</i>	the image path or name, or NULL
<i>pMenuItemPtr</i>	pointer that will contain the new menu-item, or NULL

Returns

the new sub-menu that has been added.

5.37.3.12 `gldi_menu_add_separator()`

```
void gldi_menu_add_separator (
    GtkWidget * pMenu )
```

A convenient function to add a separator to a given menu.

Parameters

<i>pMenu</i>	the menu
--------------	----------

5.38 cairo-dock-module-instance-manager.h File Reference

Data Structures

- struct [_GldiModuleInstance](#)

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

Macros

- #define [GLDI_OBJECT_IS_MODULE_INSTANCE](#)(obj)

Functions

- [GldiModuleInstance](#) * [gldi_module_instance_new](#) ([GldiModule](#) *pModule, gchar *cConfFilePath)

5.38.1 Detailed Description

This class defines the instances of modules.

A module-instance represents one instance of a module; it holds a set of data: the icon and its container, the config structure and its conf file, the data structure and a slot to plug datas into containers and icons. All these parameters are optional; a module-instance that has an icon is also called an applet.

5.38.2 Macro Definition Documentation

5.38.2.1 GLDI_OBJECT_IS_MODULE_INSTANCE

```
#define GLDI_OBJECT_IS_MODULE_INSTANCE(  
    obj )
```

Say if an object is a Module-instance.

Parameters

<i>obj</i>	the object.
------------	-------------

Returns

TRUE if the object is a Module-instance.

5.38.3 Function Documentation

5.38.3.1 gldi_module_instance_new()

```
GldiModuleInstance * gldi_module_instance_new (
    GldiModule * pModule,
    gchar * cConfFilePath )
```

Create a new instance of a module. Only activates it if bActivate == TRUE (FALSE is only valid for auto-loaded modules).

5.39 cairo-dock-module-manager.h File Reference

Data Structures

- struct [_GldiVisitCard](#)
Definition of the visit card of a module. Contains everything that is statically defined for a module.
- struct [_GldiModuleInterface](#)
Definition of the interface of a module.
- struct [_GldiModule](#)
Definition of an external module.

Macros

- #define [GLDI_ABI_VERSION](#)
- #define [GLDI_OBJECT_IS_MODULE](#)(obj)

Typedefs

- typedef gboolean(* [GldiModulePreInit](#)) ([GldiVisitCard](#) *pVisitCard, [GldiModuleInterface](#) *pInterface)
Pre-init function of a module. Fills the visit card and the interface of a module.

Enumerations

- enum [GldiModuleCategory](#)
Categories a module can be in.

Functions

- [GldiModule](#) * [gldi_module_new](#) ([GldiVisitCard](#) *pVisitCard, [GldiModuleInterface](#) *pInterface)
- [GldiModule](#) * [gldi_module_new_from_so_file](#) (const gchar *cSoFilePath)
- void [gldi_modules_new_from_directory](#) (const gchar *cModuleDirPath, GError **erreur)
- gchar * [gldi_module_get_config_dir](#) ([GldiModule](#) *pModule)
- [GldiModule](#) * [gldi_module_get](#) (const gchar *cModuleName)
- void [gldi_modules_load_auto_config](#) (void)
- void [gldi_module_activate](#) ([GldiModule](#) *module)
- void [gldi_module_deactivate](#) ([GldiModule](#) *module)
- void [gldi_module_add_instance](#) ([GldiModule](#) *pModule)
should maybe be in the module-instance too...

5.39.1 Detailed Description

This class manages the external modules of Cairo-Dock.

A module has an interface and a visit card :

- the visit card allows it to define itself (name, category, default icon, etc)
- the interface defines the entry points for init, stop, reload, read config, and reset data.

Modules can be instanciated several times; each time they are, an instance `_GldiModuleInstance` is created. Each instance holds a set of data: the icon and its container, the config structure and its conf file, the data structure and a slot to plug datas into containers and icons. All these data are optionnal; a module that has an icon is also called an applet.

5.39.2 Macro Definition Documentation

5.39.2.1 GLDI_ABI_VERSION

```
#define GLDI_ABI_VERSION
```

Define the current ABI version. Used by the new plugin loader interface to check compatibility. This version should be incremented if the layout or size of public structures changes, function parameters change, or a macro is converted to a function or vice versa. It is not required the change this when adding a function to the public API (loading the module will fail if it refers to an unresolved symbol anyway).

5.39.2.2 GLDI_OBJECT_IS_MODULE

```
#define GLDI_OBJECT_IS_MODULE(  
    obj )
```

Say if an object is a Module.

Parameters

<i>obj</i>	the object.
------------	-------------

Returns

TRUE if the object is a Module.

5.39.3 Function Documentation

5.39.3.1 gldi_module_new()

```
GldiModule * gldi_module_new (  
    GldiVisitCard * pVisitCard,  
    GldiModuleInterface * pInterface )
```

Create a new module. The module takes ownership of the 2 arguments, unless an error occurred.

Parameters

<i>pVisitCard</i>	the visit card of the module
<i>pInterface</i>	the interface of the module

Returns

the new module, or NULL if the visit card is invalid.

5.39.3.2 gldi_module_new_from_so_file()

```
GldiModule * gldi_module_new_from_so_file (
    const gchar * cSoFilePath )
```

Create a new module from a .so file.

Parameters

<i>cSoFilePath</i>	path to the .so file
--------------------	----------------------

Returns

the new module, or NULL if an error occurred.

5.39.3.3 gldi_modules_new_from_directory()

```
void gldi_modules_new_from_directory (
    const gchar * cModuleDirPath,
    GError ** erreur )
```

Create new modules from all the .so files contained in the given folder.

Parameters

<i>cModuleDirPath</i>	path to the folder
<i>erreur</i>	an error

Returns

the new module, or NULL if an error occurred.

5.39.3.4 gldi_module_get_config_dir()

```
gchar * gldi_module_get_config_dir (
    GldiModule * pModule )
```

Get the path to the folder containing the config files of a module (one file per instance). The folder is created if needed. If the module is not configurable, or if the folder couldn't be created, NULL is returned.

Parameters

<i>pModule</i>	the module
----------------	------------

Returns

the path to the folder (free it after use).

5.39.3.5 gldi_module_get()

```
GldiModule * gldi_module_get (
    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.

5.39.3.6 gldi_modules_load_auto_config()

```
void gldi_modules_load_auto_config (
    void )
```

Load the config of all auto-loaded modules without activating them.

5.39.3.7 gldi_module_activate()

```
void gldi_module_activate (
    GldiModule * module )
```

Create and initialize all the instances of a module.

Parameters

<i>module</i>	the module to activate.
---------------	-------------------------

5.39.3.8 gldi_module_deactivate()

```
void gldi_module_deactivate (
    GldiModule * module )
```

Stop and destroy all the instances of a module.

Parameters

<i>module</i>	the module to deactivate
---------------	--------------------------

5.40 cairo-dock-object.h File Reference

Data Structures

- struct [_GldiObject](#)
Definition of an Object.
- struct [_GldiObjectManager](#)
Definition of an ObjectManager.

Macros

- #define **GLDI_RUN_FIRST**
Use this in [gldi_object_register_notification](#) to be called before the core.
- #define **GLDI_RUN_AFTER**
Use this in [gldi_object_register_notification](#) to be called after the core.
- #define **GLDI_NOTIFICATION_INTERCEPT**
Return this in your callback to prevent the other callbacks from being called after you.
- #define **GLDI_NOTIFICATION_LET_PASS**
Return this in your callback to let pass the notification to the other callbacks after you.
- #define [gldi_object_notify](#)(pObject, iNotifType, ...)

Typedefs

- typedef gboolean(* **GldiNotificationFunc**) (gpointer pUserData,...)
Generic prototype of a notification callback.

Enumerations

- enum [GldiObjectNotifications](#) {
 [NOTIFICATION_NEW](#) ,
 [NOTIFICATION_DESTROY](#) ,
 NB_NOTIFICATIONS_OBJECT }
signals (any object has at least these ones)

Functions

- [GldiObject](#) * [gldi_object_new](#) ([GldiObjectManager](#) *pMgr, gpointer attr)
- void [gldi_object_ref](#) ([GldiObject](#) *pObject)
- void [gldi_object_unref](#) ([GldiObject](#) *pObject)
- void [gldi_object_delete](#) ([GldiObject](#) *pObject)
- void [gldi_object_reload](#) ([GldiObject](#) *pObject, gboolean bReloadConfig)
- void [gldi_object_register_notification](#) (gpointer pObject, GldiNotificationType iNotifType, [GldiNotificationFunc](#) pFunction, gboolean bRunFirst, gpointer pUserData)
- void [gldi_object_remove_notification](#) (gpointer pObject, GldiNotificationType iNotifType, [GldiNotificationFunc](#) pFunction, gpointer pUserData)

5.40.1 Detailed Description

This class defines the Objects, the base class of libgldi. Every element in this library is an Object. An object is defined by an ObjectManager, which defines its capabilities and signals.

Any object is created with [gldi_object_new](#) and destroyed with [gldi_object_unref](#). An object can be deleted from the current theme with [gldi_object_delete](#). An object can be reloaded with [gldi_object_reload](#).

You can listen for notifications on an object with [gldi_object_register_notification](#) and stop listening with [gldi_object_remove_notification](#). To listen for notifications on any object of a given type, simply register yourself on its ObjectManager.

5.40.2 Macro Definition Documentation

5.40.2.1 gldi_object_notify

```
#define gldi_object_notify(  
    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.

5.40.3 Enumeration Type Documentation

5.40.3.1 GldiObjectNotifications

```
enum GldiObjectNotifications
```

signals (any object has at least these ones)

Enumerator

NOTIFICATION_NEW	notification called when an object has been created. data : the object
NOTIFICATION_DESTROY	notification called when the object is going to be destroyed. data : the object

5.40.4 Function Documentation

5.40.4.1 gldi_object_new()

```
GldiObject * gldi_object_new (
```



```
GldiObjectManager * pMgr,  
gpointer attr )
```

Create a new object.

Parameters

<i>pMgr</i>	the ObjectManager
<i>attr</i>	the attributes of the object

Returns

the new object, with a reference of 1; use [gldi_object_unref](#) to destroy it

5.40.4.2 gldi_object_ref()

```
void gldi_object_ref (  
    GldiObject * pObject )
```

Take a reference on an object.

Parameters

<i>pObject</i>	the Object
----------------	------------

5.40.4.3 gldi_object_unref()

```
void gldi_object_unref (  
    GldiObject * pObject )
```

Drop your reference on an object. If it's the last reference, the object is destroyed, otherwise nothing happens.

Parameters

<i>pObject</i>	the Object
----------------	------------

5.40.4.4 gldi_object_delete()

```
void gldi_object_delete (  
    GldiObject * pObject )
```

Delete an object from the current theme. The object is unref'd, and won't be created again on next startup.

Parameters

<i>pObject</i>	the Object
----------------	------------

5.40.4.5 `gldi_object_reload()`

```
void gldi_object_reload (
    GldiObject * pObject,
    gboolean bReloadConfig )
```

Reload an object.

Parameters

<i>pObject</i>	the Object
<i>bReloadConfig</i>	TRUE to read its config file again (if the object has one)

5.40.4.6 `gldi_object_register_notification()`

```
void gldi_object_register_notification (
    gpointer pObject,
    GldiNotificationType iNotifType,
    GldiNotificationFunc pFunction,
    gboolean bRunFirst,
    gpointer pUserData )
```

Register an action to be called when a given notification is broadcasted from a given object.

Parameters

<i>pObject</i>	the object (Icon, Container, Manager).
<i>iNotifType</i>	type of the notification.
<i>pFunction</i>	callback.
<i>bRunFirst</i>	GLDI_RUN_FIRST to be called before Cairo-Dock, GLDI_RUN_AFTER to be called after.
<i>pUserData</i>	data to be passed as the first parameter of the callback.

5.40.4.7 `gldi_object_remove_notification()`

```
void gldi_object_remove_notification (
    gpointer pObject,
    GldiNotificationType iNotifType,
    GldiNotificationFunc pFunction,
    gpointer pUserData )
```

Remove a callback from the list of callbacks of a given object for a given notification and a given data. Note: it is safe to remove the callback when it is called, but not another one.

Parameters

<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.

5.41 cairo-dock-opengl-font.h File Reference

Data Structures

- struct [_CairoDockGLFont](#)

Structure used to load a font for OpenGL text rendering.

Functions

- GLuint [cairo_dock_create_texture_from_text_simple](#) (const gchar *cText, const gchar *cFontDescription, cairo_t *pSourceContext, int *iWidth, int *iHeight)
- [CairoDockGLFont](#) * [cairo_dock_load_textured_font](#) (const gchar *cFontDescription, int first, int count)
- [CairoDockGLFont](#) * [cairo_dock_load_textured_font_from_image](#) (const gchar *cImagePath)
- void [cairo_dock_free_gl_font](#) ([CairoDockGLFont](#) *pFont)
- void [cairo_dock_get_gl_text_extent](#) (const gchar *cText, [CairoDockGLFont](#) *pFont, int *iWidth, int *iHeight)
- void [cairo_dock_draw_gl_text](#) (const 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)

5.41.1 Detailed Description

This class provides different ways to draw text directly in OpenGL. [cairo_dock_create_texture_from_text_simple](#) lets you draw any text in any font, by creating a texture from a Pango font description. This is a convenient function but not very fast. For a more efficient way, you load a font into a [CairoDockGLFont](#) with either : [cairo_dock_load_textured_font](#) to load a subset of a Mono font into textures. You then use [cairo_dock_draw_gl_text_at_position](#) to draw the text.

5.41.2 Function Documentation

5.41.2.1 [cairo_dock_create_texture_from_text_simple\(\)](#)

```
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.

5.41.2.2 cairo_dock_load_textured_font()

```
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.

5.41.2.3 cairo_dock_load_textured_font_from_image()

```
CairoDockGLFont * cairo_dock_load_textured_font_from_image (
    const gchar * cImagePath )
```

Like the previous function, but loads the characters from an image. The image must be squared and contain the 256 extended ASCII characters in the alphabetic order.

Parameters

<i>cImagePath</i>	path to the image.
-------------------	--------------------

Returns

a newly allocated opengl font.

5.41.2.4 cairo_dock_free_gl_font()

```
void cairo_dock_free_gl_font (
    CairoDockGLFont * pFont )
```

Free an opengl font.

Parameters

<i>pFont</i>	the font.
--------------	-----------

5.41.2.5 cairo_dock_get_gl_text_extent()

```
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.

5.41.2.6 cairo_dock_draw_gl_text()

```
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.

5.41.2.7 cairo_dock_draw_gl_text_at_position()

```
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
--------------	----------

Parameters

<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.

5.41.2.8 cairo_dock_draw_gl_text_in_area()

```
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.

5.41.2.9 cairo_dock_draw_gl_text_at_position_in_area()

```
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.

5.42 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)

5.42.1 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](#).

5.42.2 Function Documentation

5.42.2.1 [cairo_dock_new_gl_path\(\)](#)

```
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.

5.42.2.2 cairo_dock_free_gl_path()

```
void cairo_dock_free_gl_path (
    CairoDockGLPath * pPath )
```

Destroy a path and free its allocated ressources.

Parameters

<i>pPath</i>	the path.
--------------	-----------

5.42.2.3 cairo_dock_gl_path_move_to()

```
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

5.42.2.4 cairo_dock_gl_path_set_extent()

```
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

5.42.2.5 cairo_dock_gl_path_line_to()

```
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

5.42.2.6 cairo_dock_gl_path_rel_line_to()

```
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

5.42.2.7 cairo_dock_gl_path_curve_to()

```
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

5.42.2.8 `cairo_dock_gl_path_rel_curve_to()`

```
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

5.42.2.9 `cairo_dock_gl_path_simple_curve_to()`

```
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

5.42.2.10 `cairo_dock_gl_path_rel_simple_curve_to()`

```
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

5.42.2.11 `cairo_dock_gl_path_arc()`

```
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

Parameters

<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).

5.42.2.12 cairo_dock_stroke_gl_path()

```
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.

5.42.2.13 cairo_dock_fill_gl_path()

```
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 GL_COLOR_ARRAY and feed it with a table of colors that matches the vertices.

5.42.2.14 cairo_dock_draw_rounded_rectangle_opengl()

```
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.

5.43 cairo-dock-opengl.h File Reference

Data Structures

- struct [_CairoDockGLConfig](#)

This structure summarizes the available OpenGL configuration on the system.

Macros

- #define [gldi_gl_container_begin_draw](#)(pContainer)

Functions

- gboolean [gldi_gl_backend_init](#) (gboolean bForceOpenGL)
- void [gldi_gl_init_opengl_context](#) (void)
- gboolean [gldi_gl_container_make_current](#) (GldiContainer *pContainer)
- gboolean [gldi_gl_offscreen_context_make_current](#) (void)
- gboolean [gldi_gl_container_begin_draw_full](#) (GldiContainer *pContainer, GdkRectangle *pArea, gboolean bClear)
- void [gldi_gl_container_end_draw](#) (GldiContainer *pContainer)
- void [gldi_gl_container_set_perspective_view](#) (GldiContainer *pContainer)
- void [gldi_gl_container_set_perspective_view_for_icon](#) (Icon *pIcon)
- void [gldi_gl_container_set_ortho_view](#) (GldiContainer *pContainer)
- void [gldi_gl_container_set_ortho_view_for_icon](#) (Icon *pIcon)
- void [gldi_gl_container_init](#) (GldiContainer *pContainer)
- void [gldi_gl_container_resized](#) (GldiContainer *pContainer, int iWidth, int iHeight)

5.43.1 Detailed Description

This class manages the OpenGL backend and context.

5.43.2 Macro Definition Documentation

5.43.2.1 gldi_gl_container_begin_draw

```
#define gldi_gl_container_begin_draw(  
    pContainer )
```

Start drawing on a Container's OpenGL context (draw on the whole Container and clear buffers).

Parameters

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

5.43.3 Function Documentation

5.43.3.1 `gldi_gl_backend_init()`

```
gboolean gldi_gl_backend_init (
    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.

5.43.3.2 `gldi_gl_init_opengl_context()`

```
void gldi_gl_init_opengl_context (
    void )
```

Callback from the backends to perform common initialization for a container after a context is available. The context must be made current before calling this function.

5.43.3.3 `gldi_gl_container_make_current()`

```
gboolean gldi_gl_container_make_current (
    GldiContainer * pContainer )
```

Make a Container's OpenGL context the current one.

Parameters

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

Returns

TRUE if the Container's context is now the current one.

5.43.3.4 gldi_gl_offscreen_context_make_current()

```
gboolean gldi_gl_offscreen_context_make_current (
    void )
```

Try to make current an OpenGL context that is not associated with any container, but is suitable for rendering to offscreen targets (i.e. textures). The caller must attach an FBO and a texture before rendering.

Returns

TRUE if a context was successfully set up.

5.43.3.5 gldi_gl_container_begin_draw_full()

```
gboolean gldi_gl_container_begin_draw_full (
    GldiContainer * pContainer,
    GdkRectangle * pArea,
    gboolean bClear )
```

Start drawing on a Container's OpenGL context.

Parameters

<i>pContainer</i>	the container
<i>pArea</i>	optional area to clip the drawing (NULL to draw on the whole Container)
<i>bClear</i>	whether to clear the color buffer or not

5.43.3.6 gldi_gl_container_end_draw()

```
void gldi_gl_container_end_draw (
    GldiContainer * pContainer )
```

Ends the drawing on a Container's OpenGL context.

Parameters

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

5.43.3.7 gldi_gl_container_set_perspective_view()

```
void gldi_gl_container_set_perspective_view (
    GldiContainer * pContainer )
```

Set a perspective view to the current GL context to fit a given Container. You may want to ensure the Container's context is really the current one.

Parameters

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

5.43.3.8 gldi_gl_container_set_perspective_view_for_icon()

```
void gldi_gl_container_set_perspective_view_for_icon (
    Icon * pIcon )
```

Set a perspective view to the current GL context to fit a given Icon (which must be inside a Container). You may want to ensure the Icon's Container's context is really the current one.

Parameters

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

5.43.3.9 gldi_gl_container_set_ortho_view()

```
void gldi_gl_container_set_ortho_view (
    GldiContainer * pContainer )
```

Set a orthogonal view to the current GL context to fit a given Container. You may want to ensure the Container's context is really the current one.

Parameters

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

5.43.3.10 gldi_gl_container_set_ortho_view_for_icon()

```
void gldi_gl_container_set_ortho_view_for_icon (
    Icon * pIcon )
```

Set a orthogonal view to the current GL context to fit a given Icon (which must be inside a Container). You may want to ensure the Icon's Container's context is really the current one.

Parameters

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

5.43.3.11 gldi_gl_container_init()

```
void gldi_gl_container_init (
    GldiContainer * pContainer )
```

Set a shared default-initialized GL context on a window.

Parameters

<i>pContainer</i>	the container, not yet realized.
-------------------	----------------------------------

5.43.3.12 `gldi_gl_container_resized()`

```
void gldi_gl_container_resized (
    GldiContainer * pContainer,
    int iWidth,
    int iHeight )
```

Should be called when the window is resized so that the associated EGL surface can be resized – needed on Wayland, `pContainer->iWidth` and `iHeight` should be set to the desired new size

5.44 `cairo-dock-overlay.h` File Reference

Data Structures

- struct [_CairoOverlay](#)
Definition of an Icon Overlay.

Macros

- #define [cairo_dock_set_overlay_scale](#)(pOverlay, _fScale)
- #define [cairo_dock_get_overlay_image_buffer](#)(pOverlay)

Enumerations

- enum [CairoOverlayPosition](#)
Available position of an overlay on an icon.

Functions

- [CairoOverlay](#) * [cairo_dock_add_overlay_from_image](#) ([Icon](#) *plcon, const gchar *clmageFile, [CairoOverlayPosition](#) iPosition, gpointer data)
- [CairoOverlay](#) * [cairo_dock_add_overlay_from_surface](#) ([Icon](#) *plcon, cairo_surface_t *pSurface, int iWidth, int iHeight, [CairoOverlayPosition](#) iPosition, gpointer data)
- [CairoOverlay](#) * [cairo_dock_add_overlay_from_texture](#) ([Icon](#) *plcon, GLuint iTexture, [CairoOverlayPosition](#) iPosition, gpointer data)
- void [cairo_dock_remove_overlay_at_position](#) ([Icon](#) *plcon, [CairoOverlayPosition](#) iPosition, gpointer data)
- gboolean [cairo_dock_print_overlay_on_icon_from_image](#) ([Icon](#) *plcon, const gchar *clmageFile, [CairoOverlayPosition](#) iPosition)
- void [cairo_dock_print_overlay_on_icon_from_surface](#) ([Icon](#) *plcon, cairo_surface_t *pSurface, int iWidth, int iHeight, [CairoOverlayPosition](#) iPosition)

5.44.1 Detailed Description

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

To add an overlay to an icon, use [cairo_dock_add_overlay_from_image](#) or [cairo_dock_add_overlay_from_surface](#). The overlay can then be removed from the icon by simply destroying it with [gldi_object_unref](#)

A common feature is to have only 1 overlay at a given position. This can be achieved by passing a non-NULL data to the creation functions. This data will identify all of your overlays. You can then remove an overlay simply from its position with [cairo_dock_remove_overlay_at_position](#), and adding an overlay at a position will automatically remove any previous overlay at this position with the same data.

If you're never going to update nor remove an overlay, you can choose to print it directly onto the icon with [cairo_dock_print_overlay_on_icon_from_image](#) or [cairo_dock_print_overlay_on_icon_from_surface](#), which is slightly faster.

Overlays are drawn at 1/2 of the icon size by default, but this can be set up with [cairo_dock_set_overlay_scale](#). If you need to modify an overlay directly, you can get its image buffer with [cairo_dock_get_overlay_image_buffer](#).

5.44.2 Macro Definition Documentation

5.44.2.1 [cairo_dock_set_overlay_scale](#)

```
#define cairo_dock_set_overlay_scale(  
    pOverlay,  
    _fScale )
```

Set the scale of an overlay; by default it's 0.5

Parameters

<i>pOverlay</i>	the overlay
<i>_fScale</i>	the scale

5.44.2.2 [cairo_dock_get_overlay_image_buffer](#)

```
#define cairo_dock_get_overlay_image_buffer(  
    pOverlay )
```

Get the image buffer of an overlay (only useful if you need to redraw the overlay).

Parameters

<i>pOverlay</i>	the overlay
-----------------	-------------

5.44.3 Function Documentation

5.44.3.1 `cairo_dock_add_overlay_from_image()`

```
CairoOverlay * cairo_dock_add_overlay_from_image (
    Icon * pIcon,
    const gchar * cImageFile,
    CairoOverlayPosition iPosition,
    gpointer data )
```

Add an overlay on an icon from an image.

Parameters

<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

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

Parameters

<i>data</i>	data that will be used to look for the overlay in cairo_dock_remove_overlay_at_position ; if NULL, then this function can't be used
-------------	---

5.44.3.2 `cairo_dock_add_overlay_from_surface()`

```
CairoOverlay * cairo_dock_add_overlay_from_surface (
    Icon * pIcon,
    cairo_surface_t * pSurface,
    int iWidth,
    int iHeight,
    CairoOverlayPosition iPosition,
    gpointer data )
```

Add an overlay on an icon from a surface.

Parameters

<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
<i>data</i>	data that will be used to look for the overlay in cairo_dock_remove_overlay_at_position ; if NULL, then this function can't be used

Returns

the overlay.

5.44.3.3 cairo_dock_add_overlay_from_texture()

```
CairoOverlay * cairo_dock_add_overlay_from_texture (
    Icon * pIcon,
    GLuint iTexture,
    CairoOverlayPosition iPosition,
    gpointer data )
```

Add an overlay on an icon from a texture.

Parameters

<i>pIcon</i>	the icon
<i>iTexture</i>	a texture
<i>iPosition</i>	position where to display the overlay
<i>data</i>	data that will be used to look for the overlay in cairo_dock_remove_overlay_at_position ; if NULL, then this function can't be used

Returns

the overlay.

5.44.3.4 cairo_dock_remove_overlay_at_position()

```
void cairo_dock_remove_overlay_at_position (
    Icon * pIcon,
    CairoOverlayPosition iPosition,
    gpointer data )
```

Remove an overlay from an icon, given its position and data.

Parameters

<i>pIcon</i>	the icon
<i>iPosition</i>	the position of the overlay
<i>data</i>	data that was set on the overlay when created; a NULL pointer is not valid.

5.44.3.5 cairo_dock_print_overlay_on_icon_from_image()

```
gboolean cairo_dock_print_overlay_on_icon_from_image (
    Icon * pIcon,
    const gchar * cImageFile,
    CairoOverlayPosition iPosition )
```

Print an overlay onto an icon from an image at a given position. You can't remove/modify the overlay then. The overlay will be displayed until you modify the icon directly (for instance by setting a new image).

Parameters

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

5.44.3.6 `cairo_dock_print_overlay_on_icon_from_surface()`

```
void cairo_dock_print_overlay_on_icon_from_surface (
    Icon * pIcon,
    cairo_surface_t * pSurface,
    int iWidth,
    int iHeight,
    CairoOverlayPosition iPosition )
```

Print an overlay onto an icon from a surface at a given position. You can't remove/modify the overlay then. The overlay will be displayed until you modify the icon directly (for instance by setting a new image).

Parameters

<i>plcon</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

Returns

TRUE if the overlay has been successfully printed.

5.45 `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 ,
CAIRO_DOCK_NB_TYPE_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)
- GldiTask * **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,...)
- GldiTask * **cairo_dock_get_url_data_async** (const gchar *cURL, GFunc pCallback, gpointer data)
- void **cairo_dock_free_package** (CairoDockPackage *pPackage)
- GHashTable * **cairo_dock_list_packages** (const gchar *cSharePackagesDir, const gchar *cUserPackagesDir, const gchar *cDistantPackagesDir, GHashTable *pTable)
- GldiTask * **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)

5.45.1 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**.

5.45.2 Macro Definition Documentation

5.45.2.1 **cairo_dock_get_url_data**

```
#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.

5.45.3 Enumeration Type Documentation

5.45.3.1 CairoDockPackageType

enum [CairoDockPackageType](#)

Types of packages.

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).

5.45.4 Function Documentation

5.45.4.1 cairo_dock_download_file()

```
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..

5.45.4.2 `cairo_dock_download_file_in_tmp()`

```
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.

5.45.4.3 `cairo_dock_download_archive()`

```
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.

5.45.4.4 `cairo_dock_download_file_async()`

```
GldiTask * cairo_dock_download_file_async (
    const gchar * cURL,
    const gchar * cLocalPath,
    GFunc pCallback,
    gpointer data )
```

Asynchronously download a distant file into a given location. This function is non-blocking, you'll get a CairoTask that you can discard at any time, and you'll get the path of the downloaded file as the first argument of the callback (the second being the data you passed to this function).

Parameters

<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.

5.45.4.5 `cairo_dock_get_url_data_with_post()`

```
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>bGetOutputHeaders</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.

5.45.4.6 `cairo_dock_get_url_data_async()`

```
GldiTask * cairo_dock_get_url_data_async (
    const gchar * cURL,
    GFunc pCallback,
    gpointer data )
```

Asynchronously retrieve the content of a distant URL. This function is non-blocking, you'll get a `CairoTask` that you can discard at any time, and you'll get the content of the downloaded file as the first argument of the callback (the second being the data you passed to this function).

Parameters

<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.

5.45.4.7 `cairo_dock_free_package()`

```
void cairo_dock_free_package (
    CairoDockPackage * pPackage )
```

Destroy a package and free all its allocated memory.

Parameters

<i>pPackage</i>	the package.
-----------------	--------------

5.45.4.8 `cairo_dock_list_packages()`

```
GHashTable * cairo_dock_list_packages (
    const gchar * cSharePackagesDir,
    const gchar * cUserPackagesDir,
    const gchar * cDistantPackagesDir,
    GHashTable * pTable )
```

Get a list of packages from different sources.

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>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.

5.45.4.9 `cairo_dock_list_packages_async()`

```
GldiTask * cairo_dock_list_packages_async (
    const gchar * cSharePackagesDir,
    const gchar * cUserPackagesDir,
    const gchar * cDistantPackagesDir,
    CairoDockGetPackagesFunc pCallback,
    gpointer data,
    GHashTable * pTable )
```

Asynchronously get a list of packages from 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.

5.45.4.10 `cairo_dock_get_package_path()`

```
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 <code>CAIRO_DOCK_ANY_PACKAGE</code> 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.

5.46 `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)

5.46.1 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.

5.46.2 Macro Definition Documentation

5.46.2.1 `cairo_dock_render_particles`

```
#define cairo_dock_render_particles(  
    pParticleSystem )
```

Render all the particles of a particle system.

Parameters

<code>pParticleSystem</code>	the particle system.
------------------------------	----------------------

5.46.3 Function Documentation

5.46.3.1 `cairo_dock_render_particles_full()`

```
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.

5.46.3.2 cairo_dock_create_particle_system()

```
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.

5.46.3.3 cairo_dock_free_particle_system()

```
void cairo_dock_free_particle_system (
    CairoParticleSystem * pParticleSystem )
```

Destroy a particle system, freeing all the ressources it was using.

Parameters

<i>pParticleSystem</i>	the particle system.
------------------------	----------------------

5.46.3.4 cairo_dock_update_default_particle_system()

```
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.

5.47 cairo-dock-separator-manager.h File Reference

Macros

- #define `GLDI_OBJECT_IS_SEPARATOR_ICON(obj)`

5.47.1 Detailed Description

This class handles the Separator Icons, which are user icons doing nothing.

5.47.2 Macro Definition Documentation

5.47.2.1 GLDI_OBJECT_IS_SEPARATOR_ICON

```
#define GLDI_OBJECT_IS_SEPARATOR_ICON (
    obj )
```

Say if an object is a SeparatorIcon.

Parameters

<i>obj</i>	the object.
------------	-------------

Returns

TRUE if the object is a SeparatorIcon.

5.48 cairo-dock-stack-icon-manager.h File Reference

Macros

- #define `GLDI_OBJECT_IS_STACK_ICON(obj)`

5.48.1 Detailed Description

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

5.48.2 Macro Definition Documentation

5.48.2.1 GLDI_OBJECT_IS_STACK_ICON

```
#define GLDI_OBJECT_IS_STACK_ICON(  
    obj )
```

Say if an object is a StackIcon.

Parameters

<i>obj</i>	the object.
------------	-------------

Returns

TRUE if the object is a StackIcon.

5.49 cairo-dock-style-facility.h File Reference

Data Structures

- struct [_GldiTextDescription](#)
Description of the rendering of a text.

Macros

- #define **GLDI_COLOR_SHADE_LIGHT**
A light shade level (dock background, ...)
- #define **GLDI_COLOR_SHADE_MEDIUM**
A medium shade level (selected menu-item, widget inside a dialog/menu, separator, ...)
- #define **GLDI_COLOR_SHADE_STRONG**
A strong shade level (child widget inside a dialog/menu, ...)

Enumerations

- enum [GldiStyleColors](#)
Available types of color.

Functions

- void [gldi_style_color_shade](#) (GldiColor *icolor, double shade, GldiColor *ocolor)

5.49.1 Detailed Description

This file provides a few functions dealing with style elements like colors and text.

5.49.2 Function Documentation

5.49.2.1 `gldi_style_color_shade()`

```
void gldi_style_color_shade (
    GldiColor * icolor,
    double shade,
    GldiColor * ocolor )
```

Shade a color, making it darker if it's light, and lighter if it's dark. Note that the opposite behavior can be obtained by passing a negative shade value. Alpha is copied unchanged. Both pointers can be the same.

Parameters

<i>icolor</i>	input color
<i>shade</i>	amount of light to add/remove, <= 1.
<i>ocolor</i>	output color

5.50 cairo-dock-style-manager.h File Reference

Macros

- `#define gldi_style_colors_set_bg_color(pCairoContext)`

Enumerations

- enum `GldiStyleNotifications` { `NOTIFICATION_STYLE_CHANGED` , `NB_NOTIFICATIONS_STYLE` }
signals

Functions

- void `gldi_style_color_get` (`GldiStyleColors` iColorType, `GldiColor` *pColor)
- void `gldi_style_colors_set_bg_color_full` (`cairo_t` *pCairoContext, gboolean bUseAlpha)
- void `gldi_style_colors_set_selected_bg_color` (`cairo_t` *pCairoContext)
- void `gldi_style_colors_set_line_color` (`cairo_t` *pCairoContext)
- void `gldi_style_colors_set_text_color` (`cairo_t` *pCairoContext)
- void `gldi_style_colors_set_separator_color` (`cairo_t` *pCairoContext)
- void `gldi_style_colors_set_child_color` (`cairo_t` *pCairoContext)
- void `gldi_style_colors_paint_bg_color_with_alpha` (`cairo_t` *pCairoContext, int iWidth, double fAlpha)

5.50.1 Detailed Description

This class defines the global style used by all widgets (Docks, Dialogs, Desklets, Menus, Icons). This includes background color, outline color, text color, linewidth, corner radius.

5.50.2 Macro Definition Documentation

5.50.2.1 gldi_style_colors_set_bg_color

```
#define gldi_style_colors_set_bg_color(  
    pCairoContext )
```

Set the global background color on a context.

Parameters

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

5.50.3 Enumeration Type Documentation

5.50.3.1 GldiStyleNotifications

```
enum GldiStyleNotifications
```

signals

Enumerator

NOTIFICATION_STYLE_CHANGED	notification called when the global style has changed
----------------------------	---

5.50.4 Function Documentation

5.50.4.1 gldi_style_color_get()

```
void gldi_style_color_get (  
    GldiStyleColors iColorType,  
    GldiColor * pColor )
```

Get the value of a color. In case the color is actually a pattern, it gives its dominant color. This function is really only useful when you need to have a color for sure (rather than potentially a pattern/texture), or when you need to apply the color with some transformation. Most of the time, you only want to use the `gldi_style_colors_set_*` functions.

Parameters

<i>iColorType</i>	type of the color
<i>pColor</i>	output color

5.50.4.2 gldi_style_colors_set_bg_color_full()

```
void gldi_style_colors_set_bg_color_full (
```

```
cairo_t * pCairoContext,  
gboolean bUseAlpha )
```

Set the global background color on a context, with or without the alpha component.

Parameters

<i>pCairoContext</i>	a context
<i>bUseAlpha</i>	TRUE to use the alpha, FALSE to set it fully opaque

5.50.4.3 gldi_style_colors_set_selected_bg_color()

```
void gldi_style_colors_set_selected_bg_color (  
    cairo_t * pCairoContext )
```

Set the global selected color on a context.

Parameters

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

5.50.4.4 gldi_style_colors_set_line_color()

```
void gldi_style_colors_set_line_color (  
    cairo_t * pCairoContext )
```

Set the global line color on a context.

Parameters

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

5.50.4.5 gldi_style_colors_set_text_color()

```
void gldi_style_colors_set_text_color (  
    cairo_t * pCairoContext )
```

Set the global text color on a context.

Parameters

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

5.50.4.6 gldi_style_colors_set_separator_color()

```
void gldi_style_colors_set_separator_color (  
    cairo_t * pCairoContext )
```

```
cairo_t * pCairoContext )
```

Set the global separator color on a context.

Parameters

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

5.50.4.7 gldi_style_colors_set_child_color()

```
void gldi_style_colors_set_child_color (
    cairo_t * pCairoContext )
```

Set the global child color on a context.

Parameters

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

5.50.4.8 gldi_style_colors_paint_bg_color_with_alpha()

```
void gldi_style_colors_paint_bg_color_with_alpha (
    cairo_t * pCairoContext,
    int iWidth,
    double fAlpha )
```

Paint a context with a horizontal alpha gradation. If the alpha is negative, the global style is used to find the alpha.

Parameters

<i>pCairoContext</i>	a context
<i>iWidth</i>	width of the gradation
<i>fAlpha</i>	alpha to use

5.51 cairo-dock-surface-factory.h File Reference

Macros

- #define **CAIRO_DOCK_ORIENTATION_MASK**
mask to get the orientation from a CairoDockLoadImageModifier.
- #define **cairo_dock_create_surface_for_square_icon**(cImagePath, flImageSize)
- #define **cairo_dock_create_surface_from_text**(cText, pLabelDescription, iTextWidthPtr, iTextHeightPtr)

Enumerations

- enum [CairoDockLoadImageModifier](#) {
[CAIRO_DOCK_FILL_SPACE](#) ,
[CAIRO_DOCK_KEEP_RATIO](#) ,
[CAIRO_DOCK_DONT_ZOOM_IN](#) ,
[CAIRO_DOCK_ORIENTATION_HFLIP](#) ,
[CAIRO_DOCK_ORIENTATION_ROT_180](#) ,
[CAIRO_DOCK_ORIENTATION_VFLIP](#) ,
[CAIRO_DOCK_ORIENTATION_ROT_90_HFLIP](#) ,
[CAIRO_DOCK_ORIENTATION_ROT_90](#) ,
[CAIRO_DOCK_ORIENTATION_ROT_90_VFLIP](#) ,
[CAIRO_DOCK_ORIENTATION_ROT_270](#) ,
[CAIRO_DOCK_ANIMATED_IMAGE](#) }

Types of image loading modifiers.

Functions

- cairo_surface_t * [cairo_dock_create_surface_from_xicon_buffer](#) (gulong *pXIconBuffer, int 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_full](#) (int iWidth, int iHeight, cairo_t *pSourceContext)
- GdkPixbuf * [cairo_dock_load_gdk_pixbuf](#) (const gchar *clImagePath, int iWidth, int iHeight)
- GdkPixbuf * [cairo_dock_load_gdk_pixbuf_with_max_size](#) (const gchar *clImagePath, int iMaxWidth, int iMaxHeight)
- cairo_surface_t * [cairo_dock_create_surface_from_image](#) (const gchar *clImagePath, 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 *clImageFile, double fImageWidth, double fImageHeight)
- cairo_surface_t * [cairo_dock_create_surface_from_icon](#) (const gchar *clImagePath, double fImageWidth, double fImageHeight)
- cairo_surface_t * [cairo_dock_create_surface_from_pattern](#) (const gchar *clImageFile, 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, [GldiTextDescription](#) *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)

5.51.1 Detailed Description

This class contains functions to load any image/X buffer/GdkPixbuf/text into a cairo-surface. The loading of an image can be modified by a mask, to take into account the ratio, zoom, orientation, etc.

The general way to load an image is by using [cairo_dock_create_surface_from_image](#).

If you just want to load an image at a given size, use [cairo_dock_create_surface_from_image_simple](#), or [cairo_dock_create_surface_from_icon](#).

To load a text into a surface, describe your text look with a [_GldiTextDescription](#), and pass it to [cairo_dock_create_surface_from_text](#).

Note: if you also need to load the image into a texture, it's easier to use the higher level ImageBuffer API (see [cairo_dock_create_image_buffer](#)).

5.51.2 Macro Definition Documentation

5.51.2.1 cairo_dock_create_surface_for_square_icon

```
#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.

5.51.2.2 cairo_dock_create_surface_from_text

```
#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>pLabelDescription</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.

5.51.3 Enumeration Type Documentation

5.51.3.1 CairoDockLoadImageModifier

```
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.

5.51.4 Function Documentation

5.51.4.1 cairo_dock_create_surface_from_xicon_buffer()

```
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.

5.51.4.2 cairo_dock_create_surface_from_pixbuf()

```
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>iHeightConstraint</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.

5.51.4.3 cairo_dock_create_blank_surface_full()

```
cairo_surface_t * cairo_dock_create_blank_surface_full (
    int iWidth,
    int iHeight,
    cairo_t * pSourceContext )
```

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.

5.51.4.4 cairo_dock_load_gdk_pixbuf()

```
GdkPixbuf * cairo_dock_load_gdk_pixbuf (
    const gchar * cImagePath,
    int iWidth,
    int iHeight )
```

Simple helper to read an image into a GdkPixbuf.

Parameters

<i>cImagePath</i>	complete path to the image.
<i>iWidth</i>	desired width.
<i>iHeight</i>	desired height.

Returns

the image loaded into a GdkPixbuf.

If both *iWidth* and *iHeight* are -1, the image is loaded at its natural size, otherwise it's scaled (preserving the aspect ratio).

5.51.4.5 cairo_dock_load_gdk_pixbuf_with_max_size()

```
GdkPixbuf * cairo_dock_load_gdk_pixbuf_with_max_size (
    const gchar * cImagePath,
    int iMaxWidth,
    int iMaxHeight )
```

Simple helper to read an image into a GdkPixbuf, constraining the maximum size to the given dimentions.

Parameters

<i>cImagePath</i>	complete path to the image.
<i>iMaxWidth</i>	desired maximum width. If the loaded image is wider than this, it will be scaled to this width.
<i>iMaxHeight</i>	desired maximum height. If the loaded image is taller than this, it will be scaled to this height.

Returns

the image loaded into a GdkPixbuf.

If both *iWidth* and *iHeight* are -1, the image is loaded at its natural size, otherwise it's scaled (preserving the aspect ratio).

5.51.4.6 cairo_dock_create_surface_from_image()

```
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>iHeightConstraint</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.

5.51.4.7 `cairo_dock_create_surface_from_image_simple()`

```
cairo_surface_t * cairo_dock_create_surface_from_image_simple (
    const gchar * cImageFile,
    double fImageWidth,
    double fImageHeight )
```

Create a surface from any image, at a given size. If the image is given by its sole name, it is searched inside the current theme root folder.

Parameters

<i>cImageFile</i>	path or name of an image.
<i>fImageWidth</i>	the desired surface width.
<i>fImageHeight</i>	the desired surface height.

Returns

the newly allocated surface.

5.51.4.8 `cairo_dock_create_surface_from_icon()`

```
cairo_surface_t * cairo_dock_create_surface_from_icon (
    const gchar * cImagePath,
    double fImageWidth,
    double fImageHeight )
```

Create a surface from any image, at a given size. If the image is given by its sole name, it is searched inside the icons themes known by Cairo-Dock.

Parameters

<i>cImagePath</i>	path or name of an image.
<i>fImageWidth</i>	the desired surface width.
<i>fImageHeight</i>	the desired surface height.

Returns

the newly allocated surface.

5.51.4.9 cairo_dock_create_surface_from_pattern()

```
cairo_surface_t * cairo_dock_create_surface_from_pattern (
    const gchar * cImageFile,
    double fImageWidth,
    double fImageHeight,
    double fAlpha )
```

Create a surface at a given size, and fill it with a pattern. If the pattern image is given by its sole name, it is searched inside the current theme root folder.

Parameters

<i>cImageFile</i>	path or name of an image that will be repeated to fill the surface.
<i>fImageWidth</i>	the desired surface width.
<i>fImageHeight</i>	the desired surface height.
<i>fAlpha</i>	transparency of the pattern (1 means opaque).

Returns

the newly allocated surface.

5.51.4.10 cairo_dock_rotate_surface()

```
cairo_surface_t * cairo_dock_rotate_surface (
    cairo_surface_t * pSurface,
    double fImageWidth,
    double fImageHeight,
    double fRotationAngle )
```

Create a surface by rotating another. Only works for 1/4 of rounds.

Parameters

<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.

5.51.4.11 cairo_dock_create_surface_from_text_full()

```
cairo_surface_t * cairo_dock_create_surface_from_text_full (
    const gchar * cText,
    GldiTextDescription * pLabelDescription,
    double fMaxScale,
    int iMaxWidth,
    int * iTTextWidth,
    int * iTTextHeight )
```

Create a surface representing a text, according to a given text description.

Parameters

<i>cText</i>	the text.
<i>pLabelDescription</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>iTTextWidth</i>	will be filled the width of the resulting surface.
<i>iTTextHeight</i>	will be filled the height of the resulting surface.

Returns

the newly allocated surface.

5.51.4.12 cairo_dock_duplicate_surface()

```
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.

5.52 cairo-dock-task.h File Reference

Data Structures

- struct [_GldiTask](#)

Definition of a periodic and/or asynchronous Task.

Macros

- #define [gldi_task_new](#)(iPeriod, get_data, update, pSharedMemory)
- #define [gldi_task_get_elapsed_time](#)(pTask)

Typedefs

- typedef void(* [GldiGetDataAsyncFunc](#)) (gpointer pSharedMemory)
Definition of the asynchronous job, that does the heavy part.
- typedef gboolean(* [GldiUpdateSyncFunc](#)) (gpointer pSharedMemory)
Definition of the synchronous job, that update the dock with the results of the previous job. Returns TRUE to continue, FALSE to stop.

Functions

- void [gldi_task_launch](#) ([GldiTask](#) *pTask)
- void [gldi_task_launch_delayed](#) ([GldiTask](#) *pTask, guint delay)
- [GldiTask](#) * [gldi_task_new_full](#) (int iPeriod, [GldiGetDataAsyncFunc](#) get_data, [GldiUpdateSyncFunc](#) update, GFreeFunc free_data, gpointer pSharedMemory)
- void [gldi_task_stop](#) ([GldiTask](#) *pTask)
- void [gldi_task_discard](#) ([GldiTask](#) *pTask)
- void [gldi_task_free](#) ([GldiTask](#) *pTask)
- gboolean [gldi_task_is_active](#) ([GldiTask](#) *pTask)
- gboolean [gldi_task_is_running](#) ([GldiTask](#) *pTask)
- void [gldi_task_change_frequency](#) ([GldiTask](#) *pTask, int iNewPeriod)
- void [gldi_task_change_frequency_and_relaunch](#) ([GldiTask](#) *pTask, int iNewPeriod)
- void [gldi_task_downgrade_frequency](#) ([GldiTask](#) *pTask)
- void [gldi_task_set_normal_frequency](#) ([GldiTask](#) *pTask)

5.52.1 Detailed Description

An easy way to define periodic and asynchronous tasks, that can perform heavy jobs without blocking the dock.

A Task is divided in 2 phases :

- the asynchronous phase will be executed in another thread, while the dock continues to run on its own thread, in parallel. During this phase you will do all the heavy job (like downloading a file or computing something) but you can't interact on the dock.
- the synchronous phase will be executed after the first one has finished. There you will update your applet with the result of the first phase.

Attention

A data buffer is used to communicate between the 2 phases. It is important that these datas are never accessed outside the task, and vice versa that the asynchronous thread never accesses other data than this buffer.

If you want to access these datas outside the task, you have to copy them in a safe place during the 2nd phase, or to stop the task before (beware that stopping the task means waiting for the 1st phase to finish, which can take some time).

You create a Task with [gldi_task_new](#), launch it with [gldi_task_launch](#), and destroy it with [gldi_task_free](#) or [gldi_task_discard](#).

A Task can be periodic if you specify a period, otherwise it will be executed once. It also can also be fully synchronous if you don't specify an asynchronous function.

5.52.2 Macro Definition Documentation

5.52.2.1 gldi_task_new

```
#define gldi_task_new(  
    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 get_data and update functions. Must not be accessed outside of these functions !

Returns

the newly allocated Task, ready to be launched with [gldi_task_launch](#). Free it with [gldi_task_free](#) or [gldi_task_discard](#).

5.52.2.2 gldi_task_get_elapsed_time

```
#define gldi_task_get_elapsed_time(  
    pTask )
```

Get the time elapsed since the last time the Task has run.

Parameters

<i>pTask</i>	the periodic Task.
--------------	--------------------

5.52.3 Function Documentation**5.52.3.1 gldi_task_launch()**

```
void gldi_task_launch (  
    GldiTask * pTask )
```

Launch a periodic Task, beforehand prepared with [gldi_task_new](#). The first iteration is executed immediately. The frequency returns to its normal state.

Parameters

<i>pTask</i>	the periodic Task.
--------------	--------------------

5.52.3.2 gldi_task_launch_delayed()

```
void gldi_task_launch_delayed (  
    GldiTask * pTask,  
    quint delay )
```

Same as above but after a delay. If the delay is 0, the task will be launched as soon as the main loop becomes idle.

Parameters

<i>pTask</i>	the periodic Task.
<i>fDelay</i>	delay in ms.

5.52.3.3 gldi_task_new_full()

```
GldiTask * gldi_task_new_full (  
    int iPeriod,
```



```

GldiGetDataAsyncFunc get_data,
GldiUpdateSyncFunc 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 get_data and update functions. Must not be accessed outside of these functions !

Returns

the newly allocated Task, ready to be launched with [gldi_task_launch](#). Free it with [gldi_task_free](#) or [gldi_task_discard](#).

5.52.3.4 gldi_task_stop()

```

void gldi_task_stop (
    GldiTask * pTask )

```

Stop a periodic Task. If the Task is running, it will wait until the asynchronous thread has finished, and skip the update. The Task can be launched again with a call to [gldi_task_launch](#).

Parameters

<i>pTask</i>	the periodic Task.
--------------	--------------------

5.52.3.5 gldi_task_discard()

```

void gldi_task_discard (
    GldiTask * pTask )

```

Discard a periodic Task. The asynchronous thread will continue, and the Task will be freed when it ends. The Task should be considered as destroyed after a call to this function. This function can be used inside the 'update' callback to destroy the Task.

Parameters

<i>pTask</i>	the periodic Task.
--------------	--------------------

5.52.3.6 `gldi_task_free()`

```
void gldi_task_free (
    GldiTask * pTask )
```

Stop and destroy a periodic Task, freeing all the allocated resources. Unlike [gldi_task_discard](#), 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 [gldi_task_discard](#) instead.

Parameters

<i>pTask</i>	the periodic Task.
--------------	--------------------

5.52.3.7 `gldi_task_is_active()`

```
gboolean gldi_task_is_active (
    GldiTask * 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.

5.52.3.8 `gldi_task_is_running()`

```
gboolean gldi_task_is_running (
    GldiTask * pTask )
```

Tell if a Task is running, that is to say it is either in the thread or waiting for the update.

Parameters

<i>pTask</i>	the periodic Task.
--------------	--------------------

Returns

TRUE if the Task is running.

5.52.3.9 `gldi_task_change_frequency()`

```
void gldi_task_change_frequency (
    GldiTask * pTask,
    int iNewPeriod )
```

Change the frequency of a Task. The next iteration is re-scheduled according to the new period.

Parameters

<i>pTask</i>	the periodic Task.
<i>iNewPeriod</i>	the new period between 2 iterations of the Task, in s.

5.52.3.10 gldi_task_change_frequency_and_relaunch()

```
void gldi_task_change_frequency_and_relaunch (
    GldiTask * pTask,
    int iNewPeriod )
```

Change the frequency of a Task and relaunch it immediately. The next iteration is therefore immediately executed.

Parameters

<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.

5.52.3.11 gldi_task_downgrade_frequency()

```
void gldi_task_downgrade_frequency (
    GldiTask * pTask )
```

Downgrade the frequency of a Task. The Task will be executed less often (this is typically useful to put on stand-by a periodic measure).

Parameters

<i>pTask</i>	the periodic Task.
--------------	--------------------

5.52.3.12 gldi_task_set_normal_frequency()

```
void gldi_task_set_normal_frequency (
    GldiTask * pTask )
```

Set the frequency of the Task to its normal state. This is also done automatically when launching the Task.

Parameters

<i>pTask</i>	the periodic Task.
--------------	--------------------

5.53 cairo-dock-themes-manager.h File Reference

Functions

- void [cairo_dock_update_conf_file](#) (const gchar *cConfFilePath, GType iFirstDataType,...)

- void [cairo_dock_write_keys_to_conf_file](#) (GKeyFile *pKeyFile, const gchar *cConfFilePath)
- gchar * [cairo_dock_write_keys_to_new_conf_file](#) (GKeyFile *pKeyFile, const gchar *cConfFilePath)
- gboolean [cairo_dock_export_current_theme](#) (const gchar *cNewThemeName, gboolean bSaveBehavior, gboolean bSaveLaunchers)
- gboolean [cairo_dock_package_current_theme](#) (const gchar *cThemeName, const gchar *cDirPath)
- gchar * [cairo_dock_depackage_theme](#) (const gchar *cPackagePath)
- gboolean [cairo_dock_delete_themes](#) (gchar **cThemesList)
- gboolean [cairo_dock_import_theme](#) (const gchar *cThemeName, gboolean bLoadBehavior, gboolean bLoadLaunchers)
- GldiTask * [cairo_dock_import_theme_async](#) (const gchar *cThemeName, gboolean bLoadBehavior, gboolean bLoadLaunchers, CairoDockImportThemeCB pCallback, gpointer data)
- void [cairo_dock_set_paths](#) (gchar *cRootDataDirPath, gchar *cExtraDirPath, gchar *cThemesDirPath, gchar *cCurrentThemeDirPath, gchar *cLocalThemeDirPath, gchar *cDistantThemeDirName, gchar *cThemeServerAdress)

5.53.1 Detailed Description

This class defines the structure of the global theme (launchers, icons, plug-ins, configuration files, etc). It also provides methods to manage the themes, like exporting the current theme, importing new themes, deleting themes, etc.

5.53.2 Function Documentation

5.53.2.1 [cairo_dock_update_conf_file\(\)](#)

```
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

<i>cConfFilePath</i>	path to the conf file.
<i>iFirstDataType</i>	type of the first value.

5.53.2.2 [cairo_dock_write_keys_to_conf_file\(\)](#)

```
void cairo_dock_write_keys_to_conf_file (
    GKeyFile * pKeyFile,
    const gchar * cConfFilePath )
```

Write a key file on the disk.

Parameters

<i>pKeyFile</i>	the key-file
<i>cConfFilePath</i>	its path on the disk

5.53.2.3 cairo_dock_write_keys_to_new_conf_file()

```
gchar * cairo_dock_write_keys_to_new_conf_file (
    GKeyFile * pKeyFile,
    const gchar * cConfFilePath )
```

Write a key file on the disk to a newly created configuration file.

Parameters

<i>pKeyFile</i>	the key-file
<i>cConfFilePath</i>	its path on the disk

Returns

The name of the new config file in a newly allocated string, or NULL on failure. If *cConfFilePath* does not exist, a new file is created using this path. If *cConfFilePath* already exists, a new filename is generated by using it as a template and adding a unique suffix, and the keyfile is written there instead.

5.53.2.4 cairo_dock_export_current_theme()

```
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 parameters too.
<i>bSaveLaunchers</i>	whether to save the launchers too.

Returns

TRUE if the theme could be exported succefully.

5.53.2.5 cairo_dock_package_current_theme()

```
gboolean cairo_dock_package_current_theme (
    const gchar * cThemeName,
    const gchar * cDirPath )
```

Create a package of the current theme. Packages can be distributed easily, and imported into the dock by a mere drag and drop into the Theme Manager. The package is placed in the *cDirPath* directory (or \$HOME if *cDirPath* is wrong).

Parameters

<i>cThemeName</i>	name of the package.
<i>cDirPath</i>	path to the directory

Returns

TRUE if the theme could be packaged succesfully.

5.53.2.6 cairo_dock_depackage_theme()

```
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 downoladed.
---------------------	---

Returns

the path of the theme folder, or NULL if anerror occured.

5.53.2.7 cairo_dock_delete_themes()

```
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 succesfully deleted.

5.53.2.8 cairo_dock_import_theme()

```
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 succefully.

5.53.2.9 cairo_dock_import_theme_async()

```
GldiTask * cairo_dock_import_theme_async (
    const gchar * cThemeName,
    gboolean bLoadBehavior,
    gboolean bLoadLaunchers,
    CairoDockImportThemeCB pCallback,
    gpointer data )
```

Asynchronously import a theme, which can be : a local theme, a user theme, a distant theme, or even the path to a packaged theme. This function is non-blocking, you'll get a CairoTask that you can discard at any time, and you'll get the result of the import as the first argument of the callback (the second being the data you passed to this function). Note that only downloading or unpacking the theme is done asynchronously, actually copying the files in the current theme folder is not (because it couldn't be cancelled without first making a backup).

Parameters

<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.

5.53.2.10 cairo_dock_set_paths()

```
void cairo_dock_set_paths (
    gchar * cRootDataDirPath,
    gchar * cExtraDirPath,
    gchar * cThemesDirPath,
    gchar * cCurrentThemeDirPath,
    gchar * cLocalThemeDirPath,
    gchar * cDistantThemeDirName,
    gchar * cThemeServerAdress )
```

Define the paths of themes. Do it just after 'gldi_init'.

Parameters

<i>cRootDataDirPath</i>	path to the root folder of libgldi
<i>cExtraDirPath</i>	path to the extras themes (plug-in themes)
<i>cThemesDirPath</i>	path to the user themes
<i>cCurrentThemeDirPath</i>	path to the current theme
<i>cLocalThemeDirPath</i>	path to the installed themes (default themes)
<i>cDistantThemeDirName</i>	folder of the themes on the server
<i>cThemeServerAdress</i>	adress of the themes server

5.54 cairo-dock-user-icon-manager.h File Reference

Macros

- `#define GLDI_OBJECT_IS_USER_ICON(obj)`

5.54.1 Detailed Description

This class handles the User Icons. These are Icons belonging to the user (like launchers, stack-icons, separators), and that have a config file. The config file contains at least the dock the icon belongs to and the position inside the dock.

5.54.2 Macro Definition Documentation

5.54.2.1 GLDI_OBJECT_IS_USER_ICON

```
#define GLDI_OBJECT_IS_USER_ICON(  
    obj )
```

Say if an object is a UserIcon.

Parameters

<i>obj</i>	the object.
------------	-------------

Returns

TRUE if the object is a UserIcon.

5.55 cairo-dock-utils.h File Reference

Data Structures

- struct [_GldiChildProcessManagerBackend](#)

Enumerations

- enum [GldiLaunchFlags](#) { }

Functions

- gboolean [cairo_dock_remove_version_from_string](#) (gchar *cString)
- void [cairo_dock_remove_html_spaces](#) (gchar *cString)
- void [cairo_dock_get_version_from_string](#) (const gchar *cVersionString, int *iMajorVersion, int *iMinorVersion, int *iMicroVersion)
- gboolean [cairo_dock_string_is_address](#) (const gchar *cString)
- gboolean [cairo_dock_launch_command_argv_full](#) (const gchar *const *args, const gchar *cWorkingDirectory, [GldiLaunchFlags](#) flags)
- gboolean [cairo_dock_launch_command_argv_full2](#) (const gchar *const *args, const gchar *cWorkingDirectory, [GldiLaunchFlags](#) flags, GAppInfo *app_info)
- const gchar * [cairo_dock_get_default_terminal](#) (void)

5.55.1 Detailed Description

Some helper functions.

5.55.2 Enumeration Type Documentation

5.55.2.1 GldiLaunchFlags

enum [GldiLaunchFlags](#)

Flags given to [cairo_dock_launch_command_argv_full\(\)](#)

Enumerator

GLDI_LAUNCH_GUI	This is a GUI app, use GdAppLaunchContext to create an activation token for it.
GLDI_LAUNCH_SLICE	This is a potentially long-lived app, try to put it in a separate process accounting group with the session manager. Currently, this is only supported on systemd and means starting the app as a separate, transient service. This has the effect that e.g. resource use is accounted separately, and the app is not automatically killed if cairo-dock exits.

5.55.3 Function Documentation

5.55.3.1 cairo_dock_remove_version_from_string()

```
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.

5.55.3.2 cairo_dock_remove_html_spaces()

```
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)
----------------	--

5.55.3.3 cairo_dock_get_version_from_string()

```
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.

5.55.3.4 cairo_dock_string_is_address()

```
gboolean cairo_dock_string_is_address (
    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.

5.55.3.5 cairo_dock_launch_command_argv_full()

```
gboolean cairo_dock_launch_command_argv_full (
    const gchar *const * args,
    const gchar * cWorkingDirectory,
    GldiLaunchFlags flags )
```

Launch the given command asynchronously (i.e. do not wait for it to exit).

Parameters

<i>args</i>	Argument vector with the executable name and parameters to pass.
<i>cWorkingDirectory</i>	working directory to launch the command in.
<i>flags</i>	Additional options that determine how to launch the command.

Returns

Whether successfully launched the given command. Note: currently, this always returns TRUE if flags includes GLDI_LAUNCH_SLICE; do not rely on its value in this case.

5.55.3.6 cairo_dock_launch_command_argv_full2()

```
gboolean cairo_dock_launch_command_argv_full2 (
    const gchar *const * args,
    const gchar * cWorkingDirectory,
    GldiLaunchFlags flags,
    GAppInfo * app_info )
```

Launch the given command asynchronously (i.e. do not wait for it to exit).

Parameters

<i>args</i>	Argument vector with the executable name and parameters to pass.
<i>cWorkingDirectory</i>	working directory to launch the command in.
<i>flags</i>	Additional options that determine how to launch the command.
<i>app_info</i>	An optional GAppInfo that corresponds to the app to be launched. Used to generate a startup notify ID (if flags includes GLDI_LAUNCH_GUI) and to identify the app to the system service manager (if flags includes GLDI_LAUNCH_SLICE).

Returns

Whether successfully launched the given command. Note: currently, this always returns TRUE if flags includes GLDI_LAUNCH_SLICE; do not rely on its value in this case.

5.55.3.7 cairo_dock_get_default_terminal()

```
const gchar * cairo_dock_get_default_terminal (
    void )
```

Get the command to launch the default terminal

5.56 cairo-dock-windows-manager.h File Reference

Data Structures

- struct [_GldiWindowManagerBackend](#)
Definition of the Windows Manager backend.
- struct [_GldiWindowActor](#)
Definition of a window actor.

Enumerations

- enum [GldiWindowNotifications](#)
signals

Functions

- void [gldi_windows_manager_register_backend](#) ([GldiWindowManagerBackend](#) *pBackend)
- void [gldi_windows_foreach](#) (gboolean bOrderedByZ, GFunc callback, gpointer data)
- [GldiWindowActor](#) * [gldi_windows_find](#) (gboolean(*callback)([GldiWindowActor](#) *, gpointer), gpointer data)
- [GldiWindowActor](#) * [gldi_windows_get_active](#) (void)
- void [gldi_window_set_thumbnail_area](#) ([GldiWindowActor](#) *actor, [GldiContainer](#) *pContainer, int x, int y, int w, int h)
- void [gldi_window_get_menu_address](#) ([GldiWindowActor](#) *actor, char **service_name, char **object_path)
- GPtrArray * [gldi_window_manager_get_all](#) (void)
- gboolean [gldi_window_manager_have_coordinates](#) (void)
- gboolean [gldi_window_manager_can_track_workspaces](#) (void)
- gboolean [gldi_window_manager_is_position_relative_to_current_viewport](#) (void)
- gboolean [gldi_window_manager_can_move_to_desktop](#) (void)

5.56.1 Detailed Description

This class manages the windows actors and notifies for any change on them.

5.56.2 Function Documentation

5.56.2.1 gldi_windows_manager_register_backend()

```
void gldi_windows_manager_register_backend (
    GldiWindowManagerBackend * pBackend )
```

Register a Window Manager backend. NULL functions are simply ignored.

Parameters

<i>pBackend</i>	a Window Manager backend
-----------------	--------------------------

5.56.2.2 gldi_windows_foreach()

```
void gldi_windows_foreach (
    gboolean bOrderedByZ,
    GFunc callback,
    gpointer data )
```

Run a function on each window actor.

Parameters

<i>bOrderedByZ</i>	TRUE to sort by z-order, FALSE to sort by age
<i>callback</i>	the callback
<i>data</i>	user data

5.56.2.3 gldi_windows_find()

```
GldiWindowActor * gldi_windows_find (
    gboolean(*) (GldiWindowActor *, gpointer) callback,
    gpointer data )
```

Run a function on each window actor.

Parameters

<i>callback</i>	the callback (takes the actor and the data, returns TRUE to stop)
<i>data</i>	user data

Returns

the found actor, or NULL

5.56.2.4 gldi_windows_get_active()

```
GldiWindowActor * gldi_windows_get_active (
    void )
```

Get the current active window actor.

Returns

the actor, or NULL if no window is currently active

5.56.2.5 gldi_window_set_thumbnail_area()

```
void gldi_window_set_thumbnail_area (
    GldiWindowActor * actor,
    GldiContainer * pContainer,
    int x,
    int y,
    int w,
    int h )
```

Set the position of this window's icon, to be used by the WM for its minimize animation. Note: coordinates are relative to the passed container's main surface.

5.56.2.6 gldi_window_get_menu_address()

```
void gldi_window_get_menu_address (
    GldiWindowActor * actor,
    char ** service_name,
    char ** object_path )
```

Get the object path at which this window's app might export its global menus if supported by the backend.

Parameters

<i>actor</i>	the window whose menu is requested
<i>service_name</i>	return location for the dbus service name
<i>object_path</i>	return location for the object path Note: the returned values in <i>service_name</i> and <i>object_path</i> point to strings owned by this window actor instance and should not be modified or freed by the caller.

5.56.2.7 gldi_window_manager_get_all()

```
GPtrArray * gldi_window_manager_get_all (
    void )
```

Get all currently managed windows as mebers of a GPtrArray.

Returns

a newly allocated GPtrArray with all windows (the order is unspecified); the caller should free this with `g_ptr_array_free()`

5.56.2.8 gldi_window_manager_have_coordinates()

```
gboolean gldi_window_manager_have_coordinates (
    void )
```

Check whether we can track windows' position.

Returns

whether GldiWindowActor::windowGeometry contains the actual window position; if FALSE, this should not be used

5.56.2.9 `gldi_window_manager_can_track_workspaces()`

```
gboolean gldi_window_manager_can_track_workspaces (
    void )
```

Check whether we can track which workspace / viewport / desktop windows are present on.

Returns

whether `GldiWindowActor::iNumDesktop`, `iViewPortX` and `iViewPortY` contains valid information; if `FALSE`, these should not be used

5.56.2.10 `gldi_window_manager_is_position_relative_to_current_viewport()`

```
gboolean gldi_window_manager_is_position_relative_to_current_viewport (
    void )
```

Check how window position coordinates should be interpreted. Result is only valid if `gldi_window_manager_↵have_coordinates () == TRUE` as well.

Returns

whether window coordinates should be interpreted relative to the currently active workspace / viewport; if false, coordinates are relative to the workspace / viewport that the window is currently on

5.56.2.11 `gldi_window_manager_can_move_to_desktop()`

```
gboolean gldi_window_manager_can_move_to_desktop (
    void )
```

Check whether it is possible to move a window to another desktop / viewport.

Returns

`TRUE` if it is possible to move a window; if `FALSE`, `gldi_window_move_to_current_desktop ()` and `gldi_↵window_move_to_desktop ()` will do nothing

5.57 `gldi-icon-names.h` File Reference

5.57.1 Detailed Description

This file lists the common named icons; these are generic icons that any icon-theme should provide, and they replace `gtk-stock` icons.

5.58 `cairo-dock-cinnamon-integration.h` File Reference

5.58.1 Detailed Description

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

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

5.59.1 Detailed Description

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

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

5.60.1 Detailed Description

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

5.61 cairo-dock-gauge.h File Reference

Typedefs

- typedef struct _CairoGaugeAttribute **CairoGaugeAttribute**
Attributes of a Gauge.

5.61.1 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](#).

5.62 cairo-dock-gnome-shell-integration.h File Reference

5.62.1 Detailed Description

This class implements the integration of Gnome-Shell inside Cairo-Dock.

5.63 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.

5.63.1 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](#).

5.63.2 Enumeration Type Documentation

5.63.2.1 CairoDockTypeGraph

enum `CairoDockTypeGraph`

Types of graph.

Enumerator

<code>CAIRO_DOCK_GRAPH_LINE</code>	a continuous line.
<code>CAIRO_DOCK_GRAPH_PLAIN</code>	a continuous plain graph.
<code>CAIRO_DOCK_GRAPH_BAR</code>	a histogram.
<code>CAIRO_DOCK_GRAPH_CIRCLE</code>	a circle.
<code>CAIRO_DOCK_GRAPH_CIRCLE_PLAIN</code>	a plain circle.

5.64 cairo-dock-hiding-effect.h File Reference

5.64.1 Detailed Description

This class implements the rendering interface for hiding docks.

5.65 cairo-dock-icon-container.h File Reference

5.65.1 Detailed Description

This class implements the rendering interface for icons pointing on a sub-dock.

5.66 cairo-dock-kwin-integration.h File Reference

5.66.1 Detailed Description

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

5.67 cairo-dock-progressbar.h File Reference

Data Structures

- struct [_CairoProgressBarAttribute](#)
Attributes of a Pprogressbar.

5.67.1 Detailed Description

This class defines the ProgressBar, which derives from the DataRenderer. All you need to know is the attributes that define a ProgressBar, the API to use is the common API for DataRenderer, defined in [cairo-dock-data-renderer.h](#).

5.68 cairo-dock-wayfire-integration.h File Reference

5.68.1 Detailed Description

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

Index

- [_CairoDataRenderer, 19](#)
 - [_CairoDataRendererAttribute, 20](#)
 - [_CairoDataRendererInterface, 21](#)
 - [_CairoDesklet, 22](#)
 - [_CairoDeskletAttr, 22](#)
 - [_CairoDeskletDecoration, 22](#)
 - [_CairoDeskletRenderer, 23](#)
 - [_CairoDialog, 23](#)
 - [_CairoDialogDecorator, 24](#)
 - [_CairoDialogRenderer, 24](#)
 - [_CairoDock, 25](#)
 - [iNumScreen, 27](#)
 - [_CairoDockDesktopEnvBackend, 28](#)
 - [_CairoDockGLConfig, 28](#)
 - [_CairoDockGLFont, 28](#)
 - [_CairoDockGLPath, 29](#)
 - [_CairoDockGroupKeyWidget, 29](#)
 - [_CairoDockGuiBackend, 29](#)
 - [_CairoDockHidingEffect, 30](#)
 - [_CairoDockImageBuffer, 30](#)
 - [_CairoDockPackage, 31](#)
 - [_CairoDockRenderer, 32](#)
 - [_CairoDockTransition, 32](#)
 - [_CairoGraphAttribute, 33](#)
 - [_CairoIconContainerRenderer, 34](#)
 - [_CairoOverlay, 34](#)
 - [_CairoParticle, 35](#)
 - [_CairoParticleSystem, 36](#)
 - [_CairoProgressBarAttribute, 36](#)
 - [_GldiChildProcessManagerBackend, 37](#)
 - [spawn_app, 37](#)
 - [_GldiContainer, 38](#)
 - [_GldiContainerManagerBackend, 39](#)
 - [adjust_aimed_point, 40](#)
 - [dock_check_if_mouse_inside_linear, 40](#)
 - [dock_handle_leave, 40](#)
 - [init_layer, 39](#)
 - [move_resize_dock, 40](#)
 - [set_keep_below, 39](#)
 - [update_polling_screen_edge, 40](#)
 - [_GldiDesktopBackground, 41](#)
 - [_GldiDesktopManagerBackend, 41](#)
 - [_GldiManager, 41](#)
 - [_GldiModule, 42](#)
 - [_GldiModuleInstance, 42](#)
 - [_GldiModuleInterface, 43](#)
 - [initModule, 44](#)
 - [load_custom_widget, 45](#)
 - [read_conf_file, 44](#)
 - [reloadModule, 44](#)
 - [reset_config, 44](#)
 - [reset_data, 45](#)
 - [stopModule, 44](#)
 - [_GldiObject, 45](#)
 - [_GldiObjectManager, 45](#)
 - [_GldiTask, 46](#)
 - [_GldiTextDescription, 46](#)
 - [_GldiVisitCard, 47](#)
 - [_GldiWindowActor, 48](#)
 - [_GldiWindowManagerBackend, 48](#)
 - [_Icon, 48](#)
 - [_IconInterface, 49](#)
 - [_cairo_dock_apply_texture](#)
 - [cairo-dock-draw-opengl.h, 173](#)
 - [_cairo_dock_apply_texture_at_size](#)
 - [cairo-dock-draw-opengl.h, 173](#)
 - [_cairo_dock_apply_texture_at_size_with_alpha](#)
 - [cairo-dock-draw-opengl.h, 173](#)
 - [_cairo_dock_delete_texture](#)
 - [cairo-dock-draw-opengl.h, 171](#)
 - [_cairo_dock_disable_texture](#)
 - [cairo-dock-draw-opengl.h, 172](#)
 - [_cairo_dock_enable_texture](#)
 - [cairo-dock-draw-opengl.h, 172](#)
 - [_cairo_dock_set_alpha](#)
 - [cairo-dock-draw-opengl.h, 172](#)
 - [_cairo_dock_set_blend_alpha](#)
 - [cairo-dock-draw-opengl.h, 172](#)
 - [_cairo_dock_set_blend_over](#)
 - [cairo-dock-draw-opengl.h, 172](#)
 - [_cairo_dock_set_blend_pbuffer](#)
 - [cairo-dock-draw-opengl.h, 173](#)
 - [_cairo_dock_set_blend_source](#)
 - [cairo-dock-draw-opengl.h, 172](#)
- [adjust_aimed_point](#)
- [_GldiContainerManagerBackend, 40](#)
- [cairo-dock-animations.h, 51](#)
- [cairo_dock_animation_will_be_visible, 52](#)
 - [cairo_dock_container_is_animating, 52](#)
 - [cairo_dock_get_animation_delta_t, 53](#)
 - [cairo_dock_get_slow_animation_delta_t, 53](#)
 - [cairo_dock_get_transition_count, 54](#)
 - [cairo_dock_get_transition_elapsed_time, 54](#)
 - [cairo_dock_get_transition_fraction, 54](#)
 - [cairo_dock_has_transition, 53](#)
 - [cairo_dock_launch_animation, 55](#)
 - [cairo_dock_pop_down, 55](#)

- cairo_dock_pop_up, 55
- cairo_dock_remove_transition_on_icon, 57
- cairo_dock_set_transition_on_icon, 57
- cairo_dock_trigger_icon_removal_from_dock, 56
- gldi_icon_request_animation, 56
- gldi_icon_request_attention, 56
- gldi_icon_start_animation, 55
- gldi_icon_stop_animation, 52
- gldi_icon_stop_attention, 56
- cairo-dock-applet-canvas.h, 58
 - CD_APPLET_DEFINE2_ALL_BEGIN, 60
 - CD_APPLET_DEFINE_ALL_BEGIN, 59
 - CD_APPLET_DEFINE_END, 59
 - CD_APPLET_DEFINITION, 59
 - CD_APPLET_GET_CONFIG_ALL_BEGIN, 61
 - CD_APPLET_GET_CONFIG_END, 61
 - CD_APPLET_INIT_ALL_BEGIN, 60
 - CD_APPLET_INIT_END, 60
 - CD_APPLET_ON_BUILD_MENU_BEGIN, 62
 - CD_APPLET_ON_BUILD_MENU_END, 62
 - CD_APPLET_ON_CLICK_BEGIN, 62
 - CD_APPLET_ON_CLICK_END, 62
 - CD_APPLET_ON_DOUBLE_CLICK_BEGIN, 62
 - CD_APPLET_ON_DOUBLE_CLICK_END, 62
 - CD_APPLET_ON_DROP_DATA_BEGIN, 63
 - CD_APPLET_ON_DROP_DATA_END, 63
 - CD_APPLET_ON_MIDDLE_CLICK_BEGIN, 62
 - CD_APPLET_ON_MIDDLE_CLICK_END, 62
 - CD_APPLET_ON_SCROLL_BEGIN, 63
 - CD_APPLET_ON_SCROLL_END, 63
 - CD_APPLET_ON_UPDATE_ICON_BEGIN, 63
 - CD_APPLET_ON_UPDATE_ICON_END, 63
 - CD_APPLET_PAUSE_UPDATE_ICON, 64
 - CD_APPLET_REGISTER_FOR_BUILD_MENU_EVENT, 64
 - CD_APPLET_REGISTER_FOR_CLICK_EVENT, 64
 - CD_APPLET_REGISTER_FOR_DOUBLE_CLICK_EVENT, 64
 - CD_APPLET_REGISTER_FOR_DROP_DATA_EVENT, 65
 - CD_APPLET_REGISTER_FOR_MIDDLE_CLICK_EVENT, 64
 - CD_APPLET_REGISTER_FOR_SCROLL_EVENT, 65
 - CD_APPLET_REGISTER_FOR_UPDATE_ICON_EVENT, 65
 - CD_APPLET_REGISTER_FOR_UPDATE_ICON_SLOW_EVENT, 65
 - CD_APPLET_RELOAD_ALL_BEGIN, 61
 - CD_APPLET_RELOAD_END, 61
 - CD_APPLET_RESET_CONFIG_ALL_BEGIN, 61
 - CD_APPLET_RESET_CONFIG_ALL_END, 61
 - CD_APPLET_RESET_DATA_ALL_END, 61
 - CD_APPLET_RESET_DATA_BEGIN, 61
 - CD_APPLET_SKIP_UPDATE_ICON, 63
 - CD_APPLET_STOP_BEGIN, 60
 - CD_APPLET_STOP_END, 60
- CD_APPLET_STOP_UPDATE_ICON, 63
- CD_APPLET_UNREGISTER_FOR_BUILD_MENU_EVENT, 64
- CD_APPLET_UNREGISTER_FOR_CLICK_EVENT, 64
- CD_APPLET_UNREGISTER_FOR_DOUBLE_CLICK_EVENT, 65
- CD_APPLET_UNREGISTER_FOR_DROP_DATA_EVENT, 65
- CD_APPLET_UNREGISTER_FOR_MIDDLE_CLICK_EVENT, 64
- CD_APPLET_UNREGISTER_FOR_SCROLL_EVENT, 65
- CD_APPLET_UNREGISTER_FOR_UPDATE_ICON_EVENT, 66
- CD_APPLET_UNREGISTER_FOR_UPDATE_ICON_SLOW_EVENT, 65
- cairo-dock-applet-facility.h, 66
 - cairo_dock_get_human_readable_size, 97
 - CAIRO_DOCK_INFO_NONE, 96
 - CAIRO_DOCK_INFO_ON_ICON, 96
 - CAIRO_DOCK_INFO_ON_LABEL, 96
 - cairo_dock_play_sound, 97
 - CAIRO_DOCK_REDRAW_MY_CONTAINER, 83
 - cairo_dock_set_icon_surface, 68
 - cairo_dock_set_icon_surface_full, 96
 - cairo_dock_set_image_on_icon, 96
 - cairo_dock_set_image_on_icon_with_default, 97
 - CairoDockInfoDisplay, 96
 - CD_APPLET_ADD_DATA_RENDERER_ON_MY_ICON, 92
 - CD_APPLET_ADD_ICON_IN_MY_ICONS_LIST, 94
 - CD_APPLET_ADD_IN_MENU, 79
 - CD_APPLET_ADD_IN_MENU_WITH_DATA, 79
 - CD_APPLET_ADD_IN_MENU_WITH_STOCK, 79
 - CD_APPLET_ADD_IN_MENU_WITH_STOCK_AND_DATA, 78
 - CD_APPLET_ADD_IN_MENU_WITH_TOOLTIP_AND_DATA, 78
 - CD_APPLET_ADD_OVERLAY_ON_MY_ICON, 91
 - CD_APPLET_ADD_SEPARATOR_IN_MENU, 79
 - CD_APPLET_ADD_SUB_MENU, 77
 - CD_APPLET_ADD_SUB_MENU_WITH_IMAGE, 77
 - CD_APPLET_ALLOW_NO_CLICKABLE_DESKLET, 93
 - CD_APPLET_ALT_CLICK, 81
 - CD_APPLET_ANIMATE_MY_ICON, 88
 - CD_APPLET_BIND_KEY, 82
 - CD_APPLET_CLICKED_CONTAINER, 81
 - CD_APPLET_CLICKED_ICON, 81
 - CD_APPLET_CTRL_CLICK, 81
 - CD_APPLET_DELETE_MY_ICONS_LIST, 93
 - CD_APPLET_DEMANDS_ATTENTION, 89
 - CD_APPLET_DETACH_ICON_FROM_MY_ICONS_LIST, 94
 - CD_APPLET_FINISH_DRAWING_MY_ICON, 90

CD_APPLET_FINISH_DRAWING_MY_ICON_CAIRO, 90
 CD_APPLET_GET_MY_ICON_EXTENT, 89
 CD_APPLET_LOAD_MY_ICONS_LIST, 94
 CD_APPLET_LOAD_SURFACE_FOR_MY_APPLET, 83
 CD_APPLET_LOAD_SURFACE_FOR_MY_APPLET_WITH_DEFAULT, 83
 CD_APPLET_MANAGE_APPLICATION, 95
 CD_APPLET_MY_CONF_FILE, 80
 CD_APPLET_MY_CONFIG_CHANGED, 80
 CD_APPLET_MY_CONTAINER_IS_OPENGL, 92
 CD_APPLET_MY_CONTAINER_TYPE_CHANGED, 81
 CD_APPLET_MY_ICONS_LIST, 95
 CD_APPLET_MY_ICONS_LIST_CONTAINER, 95
 CD_APPLET_MY_KEY_FILE, 80
 CD_APPLET_MY_MENU, 81
 CD_APPLET_MY_OLD_CONTAINER, 81
 CD_APPLET_POPUP_MENU_ON_MY_ICON, 80
 CD_APPLET_PRINT_OVERLAY_ON_MY_ICON, 91
 CD_APPLET_RECEIVED_DATA, 82
 CD_APPLET_REDRAW_MY_ICON, 82
 CD_APPLET_RELOAD_CONFIG_PANEL, 80
 CD_APPLET_RELOAD_CONFIG_PANEL_WITH_PAGE, 80
 CD_APPLET_RELOAD_MY_DATA_RENDERER, 92
 CD_APPLET_REMOVE_ICON_FROM_MY_ICONS_LIST, 93
 CD_APPLET_REMOVE_MY_DATA_RENDERER, 92
 CD_APPLET_REMOVE_OVERLAY_ON_MY_ICON, 91
 CD_APPLET_RENDER_NEW_DATA_ON_MY_ICON, 92
 CD_APPLET_SCROLL_DOWN, 82
 CD_APPLET_SCROLL_UP, 82
 CD_APPLET_SET_ALWAYS_VISIBLE_ICON, 88
 CD_APPLET_SET_DEFAULT_IMAGE_ON_MY_ICON_IF_NONE, 84
 CD_APPLET_SET_DESKLET_RENDERER, 93
 CD_APPLET_SET_DESKLET_RENDERER_WITH_DATA, 93
 CD_APPLET_SET_HOURS_MINUTES_AS_QUICK_INFO, 86
 CD_APPLET_SET_IMAGE_ON_MY_ICON, 84
 CD_APPLET_SET_MINUTES_SECONDES_AS_QUICK_INFO, 88
 CD_APPLET_SET_MY_DATA_RENDERER_HISTORY_TO_MAX, 92
 CD_APPLET_SET_NAME_FOR_MY_ICON, 84
 CD_APPLET_SET_NAME_FOR_MY_ICON_PRINTF, 86
 CD_APPLET_SET_QUICK_INFO_ON_MY_ICON, 86
 CD_APPLET_SET_QUICK_INFO_ON_MY_ICON_PRINTF, 86
 CD_APPLET_SET_SIZE_AS_QUICK_INFO, 88
 CD_APPLET_SET_STATIC_DESKLET, 93
 CD_APPLET_SET_STATIC_ICON, 88
 CD_APPLET_SET_SURFACE_ON_MY_ICON, 83
 CD_APPLET_SET_USER_IMAGE_ON_MY_ICON, 84
 CD_APPLET_SHIFT_CLICK, 81
 CD_APPLET_START_DRAWING_MY_ICON, 90
 CD_APPLET_START_DRAWING_MY_ICON_CAIRO, 90
 CD_APPLET_START_DRAWING_MY_ICON_OR_RETURN, 90
 CD_APPLET_START_DRAWING_MY_ICON_OR_RETURN_CAIRO, 90
 CD_APPLET_STOP_ANIMATING_MY_ICON, 89
 CD_APPLET_STOP_DEMANDING_ATTENTION, 89
 CD_APPLET_UNSET_STATIC_ICON, 88
 CD_CONFIG_GET_BOOLEAN, 69
 CD_CONFIG_GET_BOOLEAN_WITH_DEFAULT, 69
 CD_CONFIG_GET_COLOR, 76
 CD_CONFIG_GET_COLOR_RGB, 74
 CD_CONFIG_GET_COLOR_RGB_WITH_DEFAULT, 74
 CD_CONFIG_GET_COLOR_RGBA, 74
 CD_CONFIG_GET_COLOR_RGBA_WITH_DEFAULT, 73
 CD_CONFIG_GET_DOUBLE, 71
 CD_CONFIG_GET_DOUBLE_WITH_DEFAULT, 70
 CD_CONFIG_GET_FILE_PATH, 72
 CD_CONFIG_GET_GAUGE_THEME, 76
 CD_CONFIG_GET_INTEGER, 70
 CD_CONFIG_GET_INTEGER_LIST, 71
 CD_CONFIG_GET_INTEGER_WITH_DEFAULT, 69
 CD_CONFIG_GET_STRING, 72
 CD_CONFIG_GET_STRING_LIST, 73
 CD_CONFIG_GET_STRING_LIST_WITH_DEFAULT, 72
 CD_CONFIG_GET_STRING_WITH_DEFAULT, 71
 CD_CONFIG_GET_THEME_PATH, 76
 CD_CONFIG_RENAME_GROUP, 77
 cairo-dock-applet-manager.h, 98
 GLDI_OBJECT_IS_APPLET_ICON, 98
 cairo-dock-applications-manager.h, 98
 cairo_dock_foreach_appli_icon, 100
 cairo_dock_get_appli_icon, 100
 cairo_dock_get_current_active_icon, 100
 cairo_dock_get_current_applis_list, 99
 cairo_dock_start_applications_manager, 99
 GLDI_OBJECT_IS_APPLI_ICON, 99
 cairo-dock-cinnamon-integration.h, 308
 cairo-dock-class-manager.h, 101
 cairo_dock_get_class_app_info, 104

- cairo_dock_register_class, 106
- cairo_dock_register_class2, 105
- cairo_dock_set_data_from_class, 106
- gldi_app_info_from_desktop_app_info, 103
- gldi_app_info_get_desktop_action_name, 103
- gldi_app_info_get_desktop_actions, 102
- gldi_app_info_get_supported_types, 103
- gldi_app_info_launch, 102
- gldi_app_info_launch_action, 102
- gldi_app_info_new_from_commandline, 101
- gldi_app_info_set_run_in_terminal, 104
- gldi_launch_desktop_app_info, 104
- gldi_window_foreach_inhibitor, 104
- cairo-dock-compiz-integration.h, 309
- cairo-dock-config.h, 107
 - cairo_dock_decrypt_string, 107
 - cairo_dock_encrypt_string, 108
 - cairo_dock_is_loading, 107
 - cairo_dock_load_current_theme, 107
- cairo-dock-container.h, 108
 - CAIRO_DOCK_IS_CONTAINER, 110
 - cairo_dock_redraw_container, 116
 - cairo_dock_redraw_container_area, 116
 - cairo_dock_redraw_icon, 117
 - gldi_container_build_menu, 117
 - gldi_container_calculate_aimed_point, 115
 - gldi_container_calculate_aimed_point_base, 115
 - gldi_container_calculate_rect, 114
 - gldi_container_dock_check_if_mouse_inside_linear, 116
 - gldi_container_dock_handle_leave, 115
 - gldi_container_enable_drop, 110
 - gldi_container_get_current_desktop_index, 112
 - gldi_container_init_layer, 113
 - gldi_container_is_active, 113
 - gldi_container_is_wayland_backend, 114
 - gldi_container_move, 112
 - gldi_container_move_resize_dock, 114
 - gldi_container_move_to_rect, 114
 - gldi_container_notify_drop_data, 117
 - gldi_container_present, 113
 - gldi_container_reserve_space, 111
 - gldi_container_set_keep_below, 115
 - gldi_container_set_screen, 114
 - gldi_container_use_new_positioning_code, 116
 - GldiContainerNotifications, 110
 - NOTIFICATION_BUILD_CONTAINER_MENU, 110
 - NOTIFICATION_BUILD_ICON_MENU, 110
 - NOTIFICATION_CLICK_ICON, 110
 - NOTIFICATION_DOUBLE_CLICK_ICON, 111
 - NOTIFICATION_DROP_DATA, 111
 - NOTIFICATION_DROP_DATA_SELECTION, 111
 - NOTIFICATION_ENTER_ICON, 111
 - NOTIFICATION_KEY_PRESSED, 111
 - NOTIFICATION_MIDDLE_CLICK_ICON, 111
 - NOTIFICATION_MOUSE_MOVED, 111
 - NOTIFICATION_RENDER, 111
 - NOTIFICATION_SCROLL_ICON, 111
 - NOTIFICATION_SMOOTH_SCROLL_ICON, 111
 - NOTIFICATION_START_DRAG_DATA, 111
 - NOTIFICATION_UPDATE, 111
 - NOTIFICATION_UPDATE_SLOW, 111
- cairo-dock-core.h, 118
 - gldi_get_diag_msg, 118
- cairo-dock-data-renderer-manager.h, 118
 - cairo_dock_get_default_data_renderer_font, 119
 - GLDI_OBJECT_IS_DATA_RENDERER, 118
- cairo-dock-data-renderer.h, 119
 - CAIRO_DATA_RENDERER, 120
 - CAIRO_DATA_RENDERER_ATTRIBUTE, 121
 - cairo_data_renderer_format_value, 125
 - cairo_data_renderer_format_value_full, 125
 - cairo_data_renderer_get_current_value, 123
 - cairo_data_renderer_get_data, 121
 - cairo_data_renderer_get_max_value, 122
 - cairo_data_renderer_get_min_value, 122
 - cairo_data_renderer_get_nb_values, 121
 - cairo_data_renderer_get_normalized_current_value, 124
 - cairo_data_renderer_get_normalized_current_value_with_latency, 125
 - cairo_data_renderer_get_normalized_previous_value, 124
 - cairo_data_renderer_get_normalized_value, 123
 - cairo_data_renderer_get_previous_value, 123
 - cairo_data_renderer_get_value, 122
 - cairo_dock_add_new_data_renderer_on_icon, 126
 - cairo_dock_get_default_data_renderer_font, 126
 - cairo_dock_get_icon_data_renderer, 120
 - cairo_dock_refresh_data_renderer, 127
 - cairo_dock_reload_data_renderer_on_icon, 127
 - cairo_dock_remove_data_renderer_on_icon, 126
 - cairo_dock_render_new_data_on_icon, 126
 - cairo_dock_resize_data_renderer_history, 127
- cairo-dock-dbus.h, 128
 - cairo_dock_create_new_session_proxy, 129
 - cairo_dock_create_new_system_proxy, 129
 - cairo_dock_dbus_call, 132
 - cairo_dock_dbus_detect_application, 130
 - cairo_dock_dbus_detect_system_application, 130
 - cairo_dock_dbus_get_boolean, 130
 - cairo_dock_dbus_get_integer, 131
 - cairo_dock_dbus_get_string, 131
 - cairo_dock_dbus_get_string_list, 132
 - cairo_dock_dbus_get_uchar, 132
 - cairo_dock_dbus_get_uinteger, 131
 - cairo_dock_dbus_is_enabled, 129
 - cairo_dock_get_session_connection, 128
 - cairo_dock_register_service_name, 128
- cairo-dock-default-view.h, 309
- cairo-dock-desklet-factory.h, 133
 - CAIRO_DESKLET, 134
 - CAIRO_DESKLET_KEEP_ABOVE, 135
 - CAIRO_DESKLET_KEEP_BELOW, 135
 - CAIRO_DESKLET_NORMAL, 135
 - CAIRO_DESKLET_ON_WIDGET_LAYER, 135

- CAIRO_DESKLET_RESERVE_SPACE, 135
- CairoDeskletVisibility, 135
- gldi_desklet_add_interactive_widget, 135
- gldi_desklet_add_interactive_widget_with_margin, 136
- gldi_desklet_hide, 137
- gldi_desklet_lock_position, 138
- gldi_desklet_new, 135
- gldi_desklet_set_accessibility, 137
- gldi_desklet_set_margin, 136
- gldi_desklet_set_sticky, 137
- gldi_desklet_show, 137
- gldi_desklet_steal_interactive_widget, 136
- GLDI_OBJECT_IS_DESKLET, 134
- cairo-dock-desklet-manager.h, 138
 - CairoDeskletNotifications, 139
 - gldi_desklets_foreach, 139
 - gldi_desklets_foreach_icons, 140
 - gldi_desklets_set_visibility_to_default, 140
 - gldi_desklets_set_visible, 140
 - NOTIFICATION_CONFIGURE_DESKLET, 139
 - NOTIFICATION_ENTER_DESKLET, 139
 - NOTIFICATION_LEAVE_DESKLET, 139
- cairo-dock-desktop-file-db.h, 140
 - gldi_desktop_file_db_init, 141
 - gldi_desktop_file_db_lookup, 141
 - gldi_desktop_file_db_stop, 141
- cairo-dock-desktop-manager.h, 142
 - CairoDesktopNotifications, 143
 - gldi_desktop_add_workspace, 146
 - gldi_desktop_get_current, 146
 - gldi_desktop_manager_register_backend, 144
 - gldi_desktop_present_class, 144
 - gldi_desktop_present_desktops, 145
 - gldi_desktop_present_windows, 145
 - gldi_desktop_remove_last_workspace, 146
 - gldi_desktop_set_on_widget_layer, 145
 - gldi_desktop_show_widget_layer, 145
 - NOTIFICATION_DESKTOP_CHANGED, 144
 - NOTIFICATION_DESKTOP_GEOMETRY_CHANGED, 144
 - NOTIFICATION_DESKTOP_MONITOR_ADDED, 144
 - NOTIFICATION_DESKTOP_MONITOR_REMOVED, 144
 - NOTIFICATION_DESKTOP_NAMES_CHANGED, 144
 - NOTIFICATION_DESKTOP_VISIBILITY_CHANGED, 144
 - NOTIFICATION_DESKTOP_WALLPAPER_CHANGED, 144
 - NOTIFICATION_KBD_STATE_CHANGED, 144
 - NOTIFICATION_KEYMAP_CHANGED, 144
 - NOTIFICATION_MENU_REQUEST, 144
 - NOTIFICATION_SHORTKEY_PRESSED, 144
- cairo-dock-dialog-factory.h, 147
 - CAIRO_DIALOG, 148
 - CAIRO_DOCK_IS_DIALOG, 148
 - gldi_dialog_new, 149
 - gldi_dialog_show, 149
 - gldi_dialog_show_and_wait, 154
 - gldi_dialog_show_general_message, 153
 - gldi_dialog_show_temporary, 151
 - gldi_dialog_show_temporary_with_default_icon, 151
 - gldi_dialog_show_temporary_with_icon, 150
 - gldi_dialog_show_temporary_with_icon_printf, 150
 - gldi_dialog_show_with_entry, 152
 - gldi_dialog_show_with_question, 151
 - gldi_dialog_show_with_value, 153
 - gldi_dialog_steal_interactive_widget, 154
- cairo-dock-dialog-manager.h, 155
 - gldi_dialog_hide, 156
 - gldi_dialog_leave, 156
 - gldi_dialog_toggle_visibility, 156
 - gldi_dialog_unhide, 156
 - gldi_dialogs_remove_on_icon, 155
- cairo-dock-dock-facility.h, 157
 - cairo_dock_apply_wave_effect_linear, 159
 - cairo_dock_calculate_dock_icons, 158
 - cairo_dock_calculate_icons_positions_at_rest_linear, 159
 - cairo_dock_check_can_drop_linear, 161
 - cairo_dock_check_if_mouse_inside_linear, 161
 - cairo_dock_get_available_docks, 158
 - cairo_dock_get_available_docks_for_icon, 157
 - cairo_dock_get_current_dock_width_linear, 159
 - cairo_dock_get_first_drawn_element_linear, 161
 - cairo_dock_show_subdock, 158
 - cairo_dock_update_dock_size, 158
- cairo-dock-dock-factory.h, 162
 - CAIRO_DOCK, 163
 - cairo_dock_remove_icons_from_dock, 164
 - gldi_dock_enter_synthetic, 164
 - gldi_dock_leave_synthetic, 164
 - gldi_dock_new, 163
 - GLDI_OBJECT_IS_DOCK, 162
 - gldi_subdock_new, 163
- cairo-dock-dock-manager.h, 165
 - cairo_dock_reload_buffers_in_all_docks, 168
 - cairo_dock_search_icon_pointing_on_dock, 167
 - CairoDocksNotifications, 166
 - gldi_dock_add_conf_file, 169
 - gldi_dock_add_conf_file_for_name, 168
 - gldi_dock_get, 166
 - gldi_dock_get_name, 165
 - gldi_dock_get_readable_name, 166
 - gldi_dock_rename, 167
 - gldi_dock_set_visibility, 169
 - gldi_docks_foreach, 167
 - gldi_docks_foreach_root, 168
 - gldi_docks_redraw_all_root, 169
 - gldi_icons_foreach_in_docks, 168
 - NOTIFICATION_ENTER_DOCK, 166
 - NOTIFICATION_ICON_MOVED, 166
 - NOTIFICATION_INSERT_ICON, 166

- NOTIFICATION_LEAVE_DOCK, 166
- NOTIFICATION_REMOVE_ICON, 166
- cairo-dock-dock-visibility.h, 170
 - gldi_dock_has_overlapping_window, 170
 - gldi_dock_visibility_refresh, 170
- cairo-dock-draw-opengl.h, 170
 - _cairo_dock_apply_texture, 173
 - _cairo_dock_apply_texture_at_size, 173
 - _cairo_dock_apply_texture_at_size_with_alpha, 173
 - _cairo_dock_delete_texture, 171
 - _cairo_dock_disable_texture, 172
 - _cairo_dock_enable_texture, 172
 - _cairo_dock_set_alpha, 172
 - _cairo_dock_set_blend_alpha, 172
 - _cairo_dock_set_blend_over, 172
 - _cairo_dock_set_blend_pbuffer, 173
 - _cairo_dock_set_blend_source, 172
 - cairo_dock_create_texture_from_image, 171
 - cairo_dock_create_texture_from_image_full, 176
 - cairo_dock_create_texture_from_raw_data, 176
 - cairo_dock_create_texture_from_surface, 175
 - cairo_dock_create_texture_from_surface_full, 175
 - cairo_dock_render_one_icon_opengl, 175
 - cairo_dock_update_icon_texture, 177
- cairo-dock-draw.h, 177
 - cairo_dock_create_drawing_context_generic, 178
 - cairo_dock_create_drawing_context_on_area, 178
 - cairo_dock_create_drawing_context_on_container, 178
 - cairo_dock_draw_icon_cairo, 179
 - cairo_dock_draw_rounded_rectangle, 179
 - cairo_dock_draw_string, 180
 - cairo_dock_erase_cairo_context, 177
 - cairo_dock_render_one_icon, 179
- cairo-dock-file-manager.h, 180
 - cairo_dock_fm_add_monitor_full, 183
 - cairo_dock_fm_add_open_with_submenu, 187
 - cairo_dock_fm_can_eject, 184
 - cairo_dock_fm_create_file, 185
 - cairo_dock_fm_create_icon_from_URI, 186
 - cairo_dock_fm_delete_file, 184
 - cairo_dock_fm_eject_drive, 184
 - cairo_dock_fm_empty_trash, 185
 - cairo_dock_fm_get_desktop_path, 185
 - cairo_dock_fm_get_file_info, 182
 - cairo_dock_fm_get_file_properties, 182
 - cairo_dock_fm_get_pid, 187
 - cairo_dock_fm_get_trash_path, 185
 - cairo_dock_fm_is_mounted, 184
 - cairo_dock_fm_launch_uri, 183
 - cairo_dock_fm_list_apps_for_file, 185
 - cairo_dock_fm_list_directory, 182
 - cairo_dock_fm_lock_screen, 186
 - cairo_dock_fm_logout, 185
 - cairo_dock_fm_measure_directory, 182
 - cairo_dock_fm_monitor_pid, 187
 - cairo_dock_fm_mount_full, 183
 - cairo_dock_fm_move_file, 184
 - cairo_dock_fm_reboot, 186
 - cairo_dock_fm_register_vfs_backend, 182
 - cairo_dock_fm_remove_monitor_full, 183
 - cairo_dock_fm_rename_file, 184
 - cairo_dock_fm_setup_time, 186
 - cairo_dock_fm_show_system_monitor, 186
 - cairo_dock_fm_shutdown, 186
 - cairo_dock_fm_unmount_full, 183
 - cairo_dock_get_file_size, 186
- cairo-dock-gauge.h, 309
- cairo-dock-gnome-shell-integration.h, 309
- cairo-dock-graph.h, 309
 - CAIRO_DOCK_GRAPH_BAR, 310
 - CAIRO_DOCK_GRAPH_CIRCLE, 310
 - CAIRO_DOCK_GRAPH_CIRCLE_PLAIN, 310
 - CAIRO_DOCK_GRAPH_LINE, 310
 - CAIRO_DOCK_GRAPH_PLAIN, 310
 - CairoDockTypeGraph, 310
- cairo-dock-gui-factory.h, 188
 - cairo_dock_gui_find_group_key_widget_in_list, 192
 - cairo_dock_gui_image_from_file, 193
 - cairo_dock_gui_menu_item_add, 192
 - CAIRO_DOCK_WIDGET_ANIMATION_LIST, 191
 - CAIRO_DOCK_WIDGET_CHECK_BUTTON, 190
 - CAIRO_DOCK_WIDGET_CHECK_CONTROL_BUTTON, 190
 - CAIRO_DOCK_WIDGET_CLASS_SELECTOR, 191
 - CAIRO_DOCK_WIDGET_COLOR_SELECTOR_RGB, 190
 - CAIRO_DOCK_WIDGET_COLOR_SELECTOR_RGBA, 190
 - CAIRO_DOCK_WIDGET_DESKLET_DECORATION_LIST, 191
 - CAIRO_DOCK_WIDGET_DESKLET_DECORATION_LIST_WITH_D, 191
 - CAIRO_DOCK_WIDGET_DIALOG_DECORATOR_LIST, 191
 - CAIRO_DOCK_WIDGET_DOCK_LIST, 191
 - CAIRO_DOCK_WIDGET_EMPTY_FULL, 192
 - CAIRO_DOCK_WIDGET_EMPTY_WIDGET, 191
 - CAIRO_DOCK_WIDGET_EXPANDER, 192
 - CAIRO_DOCK_WIDGET_FILE_SELECTOR, 191
 - CAIRO_DOCK_WIDGET_FOLDER_SELECTOR, 191
 - CAIRO_DOCK_WIDGET_FONT_SELECTOR, 191
 - CAIRO_DOCK_WIDGET_FRAME, 192
 - CAIRO_DOCK_WIDGET_HANDBOOK, 192
 - CAIRO_DOCK_WIDGET_HSCALE_DOUBLE, 190
 - CAIRO_DOCK_WIDGET_HSCALE_INTEGER, 190
 - CAIRO_DOCK_WIDGET_ICON_THEME_LIST, 191
 - CAIRO_DOCK_WIDGET_ICONS_LIST, 191
 - CAIRO_DOCK_WIDGET_IMAGE_SELECTOR,

- 191
- CAIRO_DOCK_WIDGET_JUMP_TO_MODULE, 191
- CAIRO_DOCK_WIDGET_JUMP_TO_MODULE_IF_EXISTS, 191
- CAIRO_DOCK_WIDGET_LAUNCH_COMMAND, 191
- CAIRO_DOCK_WIDGET_LAUNCH_COMMAND_IF_CONDITION, 191
- CAIRO_DOCK_WIDGET_LINK, 192
- CAIRO_DOCK_WIDGET_LIST, 191
- CAIRO_DOCK_WIDGET_LIST_WITH_ENTRY, 191
- CAIRO_DOCK_WIDGET_NUMBERED_CONTROL_LIST, 191
- CAIRO_DOCK_WIDGET_NUMBERED_CONTROL_LIST_SEPARATOR, 191
- CAIRO_DOCK_WIDGET_NUMBERED_LIST, 191
- CAIRO_DOCK_WIDGET_PASSWORD_ENTRY, 191
- CAIRO_DOCK_WIDGET_SCREEN_LIST, 191
- CAIRO_DOCK_WIDGET_SEPARATOR, 192
- CAIRO_DOCK_WIDGET_SHORTKEY_SELECTOR, 191
- CAIRO_DOCK_WIDGET_SIZE_INTEGER, 190
- CAIRO_DOCK_WIDGET_SOUND_SELECTOR, 191
- CAIRO_DOCK_WIDGET_SPIN_DOUBLE, 190
- CAIRO_DOCK_WIDGET_SPIN_INTEGER, 190
- CAIRO_DOCK_WIDGET_STRING_ENTRY, 191
- CAIRO_DOCK_WIDGET_TEXT_LABEL, 192
- CAIRO_DOCK_WIDGET_THEME_LIST, 191
- CAIRO_DOCK_WIDGET_TREE_VIEW_MULTI_CHOICE, 191
- CAIRO_DOCK_WIDGET_TREE_VIEW_SORT, 191
- CAIRO_DOCK_WIDGET_TREE_VIEW_SORT_AND_MODULE, 191
- CAIRO_DOCK_WIDGET_VIEW_LIST, 190
- CairoDockGUIWidgetType, 190
- cairo-dock-gui-manager.h, 193
 - cairo_dock_reload_current_module_widget, 194
 - cairo_dock_set_status_message, 194
 - cairo_dock_set_status_message_printf, 194
- cairo-dock-hiding-effect.h, 310
- cairo-dock-icon-container.h, 310
- cairo-dock-icon-facility.h, 195
 - cairo_dock_begin_draw_icon, 208
 - cairo_dock_compare_icons_extension, 199
 - cairo_dock_compare_icons_name, 199
 - cairo_dock_compare_icons_order, 198
 - cairo_dock_compute_icon_area, 206
 - cairo_dock_end_draw_icon, 208
 - cairo_dock_get_current_icon_size, 206
 - cairo_dock_get_first_icon, 200
 - cairo_dock_get_first_icon_of_group, 201
 - cairo_dock_get_first_icon_of_order, 201
 - cairo_dock_get_icon_extents, 205
 - cairo_dock_get_icon_order, 196
 - cairo_dock_get_icon_type, 198
 - cairo_dock_get_icon_with_base_uri, 204
 - cairo_dock_get_icon_with_command, 204
 - cairo_dock_get_icon_with_name, 205
 - cairo_dock_get_icon_with_subdock, 205
 - cairo_dock_get_last_icon, 200
 - cairo_dock_get_last_icon_of_group, 201
 - cairo_dock_get_last_icon_of_order, 203
 - cairo_dock_get_next_element, 196
 - cairo_dock_get_next_icon, 203
 - cairo_dock_get_pointed_icon, 203
 - cairo_dock_get_previous_element, 197
 - cairo_dock_get_previous_icon, 204
 - cairo_dock_icon_buffer_to_cairo, 208
 - cairo_dock_icon_is_being_inserted, 196
 - cairo_dock_icon_is_being_removed, 196
 - cairo_dock_set_icon_always_visible, 197
 - cairo_dock_set_icon_static, 197
 - cairo_dock_sort_icons_by_name, 200
 - cairo_dock_sort_icons_by_order, 199
 - gldi_icon_is_launching, 198
 - gldi_icon_mark_as_launching, 198
 - gldi_icon_set_name, 206
 - gldi_icon_set_name_printf, 207
 - gldi_icon_set_quick_info, 207
 - gldi_icon_set_quick_info_printf, 207
- cairo-dock-icon-factory.h, 209
 - cairo_dock_create_dummy_launcher, 212
- CAIRO_DOCK_IS_APPLET, 210
- CAIRO_DOCK_IS_APPLI, 210
- CAIRO_DOCK_IS_AUTOMATIC_SEPARATOR, 211
- CAIRO_DOCK_IS_DETACHABLE_APPLET, 212
- CAIRO_DOCK_IS_ICON, 210
- CAIRO_DOCK_IS_MULTI_APPLI, 211
- CAIRO_DOCK_IS_NORMAL_APPLI, 211
- CAIRO_DOCK_IS_USER_SEPARATOR, 211
- cairo_dock_load_icon_buffers, 214
- cairo_dock_load_icon_image, 213
- cairo_dock_load_icon_quickinfo, 213
- cairo_dock_load_icon_text, 213
- gldi_icon_new, 212
- cairo-dock-icon-manager.h, 214
 - cairo_dock_search_icon_s_path, 215
 - cairo_dock_search_icon_size, 215
- CairoIconNotifications, 214
- gldi_icons_foreach, 215
- NOTIFICATION_PRE_RENDER_ICON, 215
- NOTIFICATION_RENDER_ICON, 215
- NOTIFICATION_REQUEST_ICON_ANIMATION, 215
- NOTIFICATION_STOP_ICON, 215
- NOTIFICATION_UNFOLD_SUBDOCK, 215
- NOTIFICATION_UPDATE_ICON, 215
- NOTIFICATION_UPDATE_ICON_SLOW, 215
- cairo-dock-image-buffer.h, 216
 - cairo_dock_apply_image_buffer_surface, 217

- cairo_dock_apply_image_buffer_surface_at_size, 220
- cairo_dock_apply_image_buffer_surface_with_offset, 220
- cairo_dock_apply_image_buffer_texture, 217
- cairo_dock_apply_image_buffer_texture_at_size, 221
- cairo_dock_apply_image_buffer_texture_with_offset, 220
- cairo_dock_create_icon_fbo, 221
- cairo_dock_create_image_buffer, 219
- cairo_dock_destroy_icon_fbo, 221
- cairo_dock_free_image_buffer, 220
- cairo_dock_image_buffer_copy_scale, 221
- cairo_dock_load_image_buffer, 217
- cairo_dock_load_image_buffer_from_surface, 219
- cairo_dock_load_image_buffer_full, 218
- cairo_dock_search_image_s_path, 218
- cairo_dock_unload_image_buffer, 219
- cairo-dock-indicator-manager.h, 222
- cairo-dock-keybinder.h, 222
 - cairo_dock_trigger_shortkey, 224
 - gldi_shortkey_could_grab, 222
 - gldi_shortkey_new, 223
 - gldi_shortkey_rebind, 223
- cairo-dock-keyfile-utilities.h, 224
 - cairo_dock_add_group_key_to_conf_file, 227
 - cairo_dock_add_remove_element_to_key, 227
 - cairo_dock_conf_file_needs_update, 227
 - cairo_dock_get_conf_file_version, 227
 - cairo_dock_merge_conf_files, 225
 - cairo_dock_open_key_file, 225
 - cairo_dock_remove_group_key_from_conf_file, 228
 - cairo_dock_update_keyfile, 228
 - cairo_dock_upgrade_conf_file_full, 227
 - cairo_dock_write_keys_to_file_full, 225
 - cairo_dock_write_keys_to_new_file, 225
- cairo-dock-kwin-integration.h, 311
- cairo-dock-launcher-manager.h, 228
 - gldi_launcher_add_new, 229
 - gldi_launcher_add_new_full, 229
 - GLDI_OBJECT_IS_LAUNCHER_ICON, 229
- cairo-dock-manager.h, 230
 - GLDI_OBJECT_IS_MANAGER, 230
- cairo-dock-menu.h, 230
 - gldi_menu_add_item, 235
 - gldi_menu_add_item_with_tooltip, 235
 - gldi_menu_add_separator, 236
 - gldi_menu_add_sub_menu, 232
 - gldi_menu_add_sub_menu_full, 236
 - gldi_menu_init, 232
 - gldi_menu_item_get_image, 235
 - gldi_menu_item_new, 231
 - gldi_menu_item_new_full2, 233
 - gldi_menu_item_new_with_action, 233
 - gldi_menu_item_new_with_submenu, 234
 - gldi_menu_item_set_image, 234
 - gldi_menu_new, 232
 - gldi_menu_popup_full, 233
 - gldi_submenu_new, 231
- cairo-dock-module-instance-manager.h, 237
 - gldi_module_instance_new, 238
 - GLDI_OBJECT_IS_MODULE_INSTANCE, 237
- cairo-dock-module-manager.h, 238
 - GLDI_ABI_VERSION, 239
 - gldi_module_activate, 241
 - gldi_module_deactivate, 241
 - gldi_module_get, 241
 - gldi_module_get_config_dir, 240
 - gldi_module_new, 239
 - gldi_module_new_from_so_file, 240
 - gldi_modules_load_auto_config, 241
 - gldi_modules_new_from_directory, 240
 - GLDI_OBJECT_IS_MODULE, 239
- cairo-dock-object.h, 243
 - gldi_object_delete, 245
 - gldi_object_new, 244
 - gldi_object_notify, 244
 - gldi_object_ref, 245
 - gldi_object_register_notification, 246
 - gldi_object_reload, 245
 - gldi_object_remove_notification, 246
 - gldi_object_unref, 245
 - GldiObjectNotifications, 244
 - NOTIFICATION_DESTROY, 244
 - NOTIFICATION_NEW, 244
- cairo-dock-opengl-font.h, 247
 - cairo_dock_create_texture_from_text_simple, 247
 - cairo_dock_draw_gl_text, 249
 - cairo_dock_draw_gl_text_at_position, 249
 - cairo_dock_draw_gl_text_at_position_in_area, 250
 - cairo_dock_draw_gl_text_in_area, 250
 - cairo_dock_free_gl_font, 248
 - cairo_dock_get_gl_text_extent, 249
 - cairo_dock_load_textured_font, 248
 - cairo_dock_load_textured_font_from_image, 248
- cairo-dock-opengl-path.h, 251
 - cairo_dock_draw_rounded_rectangle_opengl, 257
 - cairo_dock_fill_gl_path, 257
 - cairo_dock_free_gl_path, 252
 - cairo_dock_gl_path_arc, 256
 - cairo_dock_gl_path_curve_to, 253
 - cairo_dock_gl_path_line_to, 253
 - cairo_dock_gl_path_move_to, 252
 - cairo_dock_gl_path_rel_curve_to, 255
 - cairo_dock_gl_path_rel_line_to, 253
 - cairo_dock_gl_path_rel_simple_curve_to, 256
 - cairo_dock_gl_path_set_extent, 252
 - cairo_dock_gl_path_simple_curve_to, 255
 - cairo_dock_new_gl_path, 251
 - cairo_dock_stroke_gl_path, 257
- cairo-dock-opengl.h, 258
 - gldi_gl_backend_init, 259
 - gldi_gl_container_begin_draw, 258
 - gldi_gl_container_begin_draw_full, 260

- gldi_gl_container_end_draw, 260
- gldi_gl_container_init, 261
- gldi_gl_container_make_current, 259
- gldi_gl_container_resized, 262
- gldi_gl_container_set_ortho_view, 261
- gldi_gl_container_set_ortho_view_for_icon, 261
- gldi_gl_container_set_perspective_view, 260
- gldi_gl_container_set_perspective_view_for_icon, 261
- gldi_gl_init_opengl_context, 259
- gldi_gl_offscreen_context_make_current, 259
- cairo-dock-overlay.h, 262
 - cairo_dock_add_overlay_from_image, 264
 - cairo_dock_add_overlay_from_surface, 264
 - cairo_dock_add_overlay_from_texture, 265
 - cairo_dock_get_overlay_image_buffer, 263
 - cairo_dock_print_overlay_on_icon_from_image, 265
 - cairo_dock_print_overlay_on_icon_from_surface, 267
 - cairo_dock_remove_overlay_at_position, 265
 - cairo_dock_set_overlay_scale, 263
- cairo-dock-packages.h, 267
 - CAIRO_DOCK_ANY_PACKAGE, 269
 - CAIRO_DOCK_DISTANT_PACKAGE, 269
 - cairo_dock_download_archive, 270
 - cairo_dock_download_file, 269
 - cairo_dock_download_file_async, 270
 - cairo_dock_download_file_in_tmp, 269
 - cairo_dock_free_package, 272
 - cairo_dock_get_package_path, 273
 - cairo_dock_get_url_data, 268
 - cairo_dock_get_url_data_async, 271
 - cairo_dock_get_url_data_with_post, 271
 - cairo_dock_list_packages, 272
 - cairo_dock_list_packages_async, 272
 - CAIRO_DOCK_LOCAL_PACKAGE, 269
 - CAIRO_DOCK_NEW_PACKAGE, 269
 - CAIRO_DOCK_UPDATED_PACKAGE, 269
 - CAIRO_DOCK_USER_PACKAGE, 269
 - CairoDockPackageType, 269
- cairo-dock-particle-system.h, 273
 - cairo_dock_create_particle_system, 275
 - cairo_dock_free_particle_system, 275
 - cairo_dock_render_particles, 274
 - cairo_dock_render_particles_full, 274
 - cairo_dock_update_default_particle_system, 275
- cairo-dock-progressbar.h, 311
- cairo-dock-separator-manager.h, 276
 - GLDI_OBJECT_IS_SEPARATOR_ICON, 276
- cairo-dock-stack-icon-manager.h, 276
 - GLDI_OBJECT_IS_STACK_ICON, 277
- cairo-dock-style-facility.h, 277
 - gldi_style_color_shade, 278
- cairo-dock-style-manager.h, 278
 - gldi_style_color_get, 279
 - gldi_style_colors_paint_bg_color_with_alpha, 281
 - gldi_style_colors_set_bg_color, 279
 - gldi_style_colors_set_bg_color_full, 279
 - gldi_style_colors_set_child_color, 281
 - gldi_style_colors_set_line_color, 280
 - gldi_style_colors_set_selected_bg_color, 280
 - gldi_style_colors_set_separator_color, 280
 - gldi_style_colors_set_text_color, 280
 - GldiStyleNotifications, 279
 - NOTIFICATION_STYLE_CHANGED, 279
- cairo-dock-surface-factory.h, 281
 - CAIRO_DOCK_ANIMATED_IMAGE, 284
 - cairo_dock_create_blank_surface_full, 285
 - cairo_dock_create_surface_for_square_icon, 283
 - cairo_dock_create_surface_from_icon, 287
 - cairo_dock_create_surface_from_image, 286
 - cairo_dock_create_surface_from_image_simple, 287
 - cairo_dock_create_surface_from_pattern, 288
 - cairo_dock_create_surface_from_pixbuf, 284
 - cairo_dock_create_surface_from_text, 283
 - cairo_dock_create_surface_from_text_full, 288
 - cairo_dock_create_surface_from_xicon_buffer, 284
 - CAIRO_DOCK_DONT_ZOOM_IN, 284
 - cairo_dock_duplicate_surface, 289
 - CAIRO_DOCK_FILL_SPACE, 284
 - CAIRO_DOCK_KEEP_RATIO, 284
 - cairo_dock_load_gdk_pixbuf, 285
 - cairo_dock_load_gdk_pixbuf_with_max_size, 286
 - CAIRO_DOCK_ORIENTATION_HFLIP, 284
 - CAIRO_DOCK_ORIENTATION_ROT_180, 284
 - CAIRO_DOCK_ORIENTATION_ROT_270, 284
 - CAIRO_DOCK_ORIENTATION_ROT_90, 284
 - CAIRO_DOCK_ORIENTATION_ROT_90_HFLIP, 284
 - CAIRO_DOCK_ORIENTATION_ROT_90_VFLIP, 284
 - CAIRO_DOCK_ORIENTATION_VFLIP, 284
 - cairo_dock_rotate_surface, 288
 - CairoDockLoadImageModifier, 283
- cairo-dock-task.h, 290
 - gldi_task_change_frequency, 294
 - gldi_task_change_frequency_and_relaunch, 296
 - gldi_task_discard, 293
 - gldi_task_downgrade_frequency, 296
 - gldi_task_free, 293
 - gldi_task_get_elapsed_time, 292
 - gldi_task_is_active, 294
 - gldi_task_is_running, 294
 - gldi_task_launch, 292
 - gldi_task_launch_delayed, 292
 - gldi_task_new, 291
 - gldi_task_new_full, 292
 - gldi_task_set_normal_frequency, 296
 - gldi_task_stop, 293
- cairo-dock-themes-manager.h, 296
 - cairo_dock_delete_themes, 299
 - cairo_dock_depackage_theme, 299
 - cairo_dock_export_current_theme, 298

- cairo_dock_import_theme, 299
- cairo_dock_import_theme_async, 300
- cairo_dock_package_current_theme, 298
- cairo_dock_set_paths, 300
- cairo_dock_update_conf_file, 297
- cairo_dock_write_keys_to_conf_file, 297
- cairo_dock_write_keys_to_new_conf_file, 298
- cairo-dock-user-icon-manager.h, 301
 - GLDI_OBJECT_IS_USER_ICON, 301
- cairo-dock-utils.h, 301
 - cairo_dock_get_default_terminal, 304
 - cairo_dock_get_version_from_string, 303
 - cairo_dock_launch_command_argv_full, 304
 - cairo_dock_launch_command_argv_full2, 304
 - cairo_dock_remove_html_spaces, 303
 - cairo_dock_remove_version_from_string, 302
 - cairo_dock_string_is_address, 303
 - GLDI_LAUNCH_GUI, 302
 - GLDI_LAUNCH_SLICE, 302
 - GldiLaunchFlags, 302
- cairo-dock-wayfire-integration.h, 311
- cairo-dock-windows-manager.h, 305
 - gldi_window_get_menu_address, 307
 - gldi_window_manager_can_move_to_desktop, 308
 - gldi_window_manager_can_track_workspaces, 307
 - gldi_window_manager_get_all, 307
 - gldi_window_manager_have_coordinates, 307
 - gldi_window_manager_is_position_relative_to_current_viewport, 308
 - gldi_window_set_thumbnail_area, 306
 - gldi_windows_find, 306
 - gldi_windows_foreach, 306
 - gldi_windows_get_active, 306
 - gldi_windows_manager_register_backend, 305
- Cairo-Dock's API documentation., 1
- CAIRO_DATA_RENDERER
 - cairo-dock-data-renderer.h, 120
- CAIRO_DATA_RENDERER_ATTRIBUTE
 - cairo-dock-data-renderer.h, 121
- cairo_data_renderer_format_value
 - cairo-dock-data-renderer.h, 125
- cairo_data_renderer_format_value_full
 - cairo-dock-data-renderer.h, 125
- cairo_data_renderer_get_current_value
 - cairo-dock-data-renderer.h, 123
- cairo_data_renderer_get_data
 - cairo-dock-data-renderer.h, 121
- cairo_data_renderer_get_max_value
 - cairo-dock-data-renderer.h, 122
- cairo_data_renderer_get_min_value
 - cairo-dock-data-renderer.h, 122
- cairo_data_renderer_get_nb_values
 - cairo-dock-data-renderer.h, 121
- cairo_data_renderer_get_normalized_current_value
 - cairo-dock-data-renderer.h, 124
- cairo_data_renderer_get_normalized_current_value_with_animation
 - cairo-dock-data-renderer.h, 124
- cairo-dock-data-renderer.h, 125
 - cairo_data_renderer_get_normalized_previous_value, 124
 - cairo_data_renderer_get_normalized_value, 123
 - cairo_data_renderer_get_previous_value, 123
 - cairo_data_renderer_get_value, 122
- CAIRO_DESKLET
 - cairo-dock-desklet-factory.h, 134
- CAIRO_DESKLET_KEEP_ABOVE
 - cairo-dock-desklet-factory.h, 135
- CAIRO_DESKLET_KEEP_BELOW
 - cairo-dock-desklet-factory.h, 135
- CAIRO_DESKLET_NORMAL
 - cairo-dock-desklet-factory.h, 135
- CAIRO_DESKLET_ON_WIDGET_LAYER
 - cairo-dock-desklet-factory.h, 135
- CAIRO_DESKLET_RESERVE_SPACE
 - cairo-dock-desklet-factory.h, 135
- CAIRO_DIALOG
 - cairo-dock-dialog-factory.h, 148
- CAIRO_DOCK
 - cairo-dock-dock-factory.h, 163
- cairo_dock_add_group_key_to_conf_file
 - cairo-dock-keyfile-utilities.h, 227
- cairo_dock_add_new_data_renderer_on_icon
 - cairo-dock-data-renderer.h, 126
- cairo_dock_add_overlay_from_image
 - cairo-dock-overlay.h, 264
- cairo_dock_add_overlay_from_surface
 - cairo-dock-overlay.h, 264
- cairo_dock_add_overlay_from_texture
 - cairo-dock-overlay.h, 265
- cairo_dock_add_remove_element_to_key
 - cairo-dock-keyfile-utilities.h, 227
- CAIRO_DOCK_ANIMATED_IMAGE
 - cairo-dock-surface-factory.h, 284
- cairo_dock_animation_will_be_visible
 - cairo-dock-animations.h, 52
- CAIRO_DOCK_ANY_PACKAGE
 - cairo-dock-packages.h, 269
- cairo_dock_apply_image_buffer_surface
 - cairo-dock-image-buffer.h, 217
- cairo_dock_apply_image_buffer_surface_at_size
 - cairo-dock-image-buffer.h, 220
- cairo_dock_apply_image_buffer_surface_with_offset
 - cairo-dock-image-buffer.h, 220
- cairo_dock_apply_image_buffer_texture
 - cairo-dock-image-buffer.h, 217
- cairo_dock_apply_image_buffer_texture_at_size
 - cairo-dock-image-buffer.h, 221
- cairo_dock_apply_image_buffer_texture_with_offset
 - cairo-dock-image-buffer.h, 220
- cairo_dock_apply_wave_effect_linear
 - cairo-dock-dock-facility.h, 159
- cairo_dock_begin_draw_icon
 - cairo-dock-dock-facility.h, 159

- cairo-dock-icon-facility.h, 208
- cairo_dock_calculate_dock_icons
 - cairo-dock-dock-facility.h, 158
- cairo_dock_calculate_icons_positions_at_rest_linear
 - cairo-dock-dock-facility.h, 159
- cairo_dock_check_can_drop_linear
 - cairo-dock-dock-facility.h, 161
- cairo_dock_check_if_mouse_inside_linear
 - cairo-dock-dock-facility.h, 161
- cairo_dock_compare_icons_extension
 - cairo-dock-icon-facility.h, 199
- cairo_dock_compare_icons_name
 - cairo-dock-icon-facility.h, 199
- cairo_dock_compare_icons_order
 - cairo-dock-icon-facility.h, 198
- cairo_dock_compute_icon_area
 - cairo-dock-icon-facility.h, 206
- cairo_dock_conf_file_needs_update
 - cairo-dock-keyfile-utilities.h, 227
- cairo_dock_container_is_animating
 - cairo-dock-animations.h, 52
- cairo_dock_create_blank_surface_full
 - cairo-dock-surface-factory.h, 285
- cairo_dock_create_drawing_context_generic
 - cairo-dock-draw.h, 178
- cairo_dock_create_drawing_context_on_area
 - cairo-dock-draw.h, 178
- cairo_dock_create_drawing_context_on_container
 - cairo-dock-draw.h, 178
- cairo_dock_create_dummy_launcher
 - cairo-dock-icon-factory.h, 212
- cairo_dock_create_icon_fbo
 - cairo-dock-image-buffer.h, 221
- cairo_dock_create_image_buffer
 - cairo-dock-image-buffer.h, 219
- cairo_dock_create_new_session_proxy
 - cairo-dock-dbus.h, 129
- cairo_dock_create_new_system_proxy
 - cairo-dock-dbus.h, 129
- cairo_dock_create_particle_system
 - cairo-dock-particle-system.h, 275
- cairo_dock_create_surface_for_square_icon
 - cairo-dock-surface-factory.h, 283
- cairo_dock_create_surface_from_icon
 - cairo-dock-surface-factory.h, 287
- cairo_dock_create_surface_from_image
 - cairo-dock-surface-factory.h, 286
- cairo_dock_create_surface_from_image_simple
 - cairo-dock-surface-factory.h, 287
- cairo_dock_create_surface_from_pattern
 - cairo-dock-surface-factory.h, 288
- cairo_dock_create_surface_from_pixbuf
 - cairo-dock-surface-factory.h, 284
- cairo_dock_create_surface_from_text
 - cairo-dock-surface-factory.h, 283
- cairo_dock_create_surface_from_text_full
 - cairo-dock-surface-factory.h, 288
- cairo_dock_create_surface_from_xicon_buffer
 - cairo-dock-surface-factory.h, 284
- cairo_dock_create_texture_from_image
 - cairo-dock-draw-opengl.h, 171
- cairo_dock_create_texture_from_image_full
 - cairo-dock-draw-opengl.h, 176
- cairo_dock_create_texture_from_raw_data
 - cairo-dock-draw-opengl.h, 176
- cairo_dock_create_texture_from_surface
 - cairo-dock-draw-opengl.h, 175
- cairo_dock_create_texture_from_surface_full
 - cairo-dock-draw-opengl.h, 175
- cairo_dock_create_texture_from_text_simple
 - cairo-dock-opengl-font.h, 247
- cairo_dock_dbus_call
 - cairo-dock-dbus.h, 132
- cairo_dock_dbus_detect_application
 - cairo-dock-dbus.h, 130
- cairo_dock_dbus_detect_system_application
 - cairo-dock-dbus.h, 130
- cairo_dock_dbus_get_boolean
 - cairo-dock-dbus.h, 130
- cairo_dock_dbus_get_integer
 - cairo-dock-dbus.h, 131
- cairo_dock_dbus_get_string
 - cairo-dock-dbus.h, 131
- cairo_dock_dbus_get_string_list
 - cairo-dock-dbus.h, 132
- cairo_dock_dbus_get_uchar
 - cairo-dock-dbus.h, 132
- cairo_dock_dbus_get_uinteger
 - cairo-dock-dbus.h, 131
- cairo_dock_dbus_is_enabled
 - cairo-dock-dbus.h, 129
- cairo_dock_decrypt_string
 - cairo-dock-config.h, 107
- cairo_dock_delete_themes
 - cairo-dock-themes-manager.h, 299
- cairo_dock_depackage_theme
 - cairo-dock-themes-manager.h, 299
- cairo_dock_destroy_icon_fbo
 - cairo-dock-image-buffer.h, 221
- CAIRO_DOCK_DISTANT_PACKAGE
 - cairo-dock-packages.h, 269
- CAIRO_DOCK_DONT_ZOOM_IN
 - cairo-dock-surface-factory.h, 284
- cairo_dock_download_archive
 - cairo-dock-packages.h, 270
- cairo_dock_download_file
 - cairo-dock-packages.h, 269
- cairo_dock_download_file_async
 - cairo-dock-packages.h, 270
- cairo_dock_download_file_in_tmp
 - cairo-dock-packages.h, 269
- cairo_dock_draw_gl_text
 - cairo-dock-opengl-font.h, 249
- cairo_dock_draw_gl_text_at_position
 - cairo-dock-opengl-font.h, 249
- cairo_dock_draw_gl_text_at_position_in_area

- cairo-dock-opengl-font.h, 250
- cairo_dock_draw_gl_text_in_area
 - cairo-dock-opengl-font.h, 250
- cairo_dock_draw_icon_cairo
 - cairo-dock-draw.h, 179
- cairo_dock_draw_rounded_rectangle
 - cairo-dock-draw.h, 179
- cairo_dock_draw_rounded_rectangle_opengl
 - cairo-dock-opengl-path.h, 257
- cairo_dock_draw_string
 - cairo-dock-draw.h, 180
- cairo_dock_duplicate_surface
 - cairo-dock-surface-factory.h, 289
- cairo_dock_encrypt_string
 - cairo-dock-config.h, 108
- cairo_dock_end_draw_icon
 - cairo-dock-icon-facility.h, 208
- cairo_dock_erase_cairo_context
 - cairo-dock-draw.h, 177
- cairo_dock_export_current_theme
 - cairo-dock-themes-manager.h, 298
- cairo_dock_fill_gl_path
 - cairo-dock-opengl-path.h, 257
- CAIRO_DOCK_FILL_SPACE
 - cairo-dock-surface-factory.h, 284
- cairo_dock_fm_add_monitor_full
 - cairo-dock-file-manager.h, 183
- cairo_dock_fm_add_open_with_submenu
 - cairo-dock-file-manager.h, 187
- cairo_dock_fm_can_eject
 - cairo-dock-file-manager.h, 184
- cairo_dock_fm_create_file
 - cairo-dock-file-manager.h, 185
- cairo_dock_fm_create_icon_from_URI
 - cairo-dock-file-manager.h, 186
- cairo_dock_fm_delete_file
 - cairo-dock-file-manager.h, 184
- cairo_dock_fm_eject_drive
 - cairo-dock-file-manager.h, 184
- cairo_dock_fm_empty_trash
 - cairo-dock-file-manager.h, 185
- cairo_dock_fm_get_desktop_path
 - cairo-dock-file-manager.h, 185
- cairo_dock_fm_get_file_info
 - cairo-dock-file-manager.h, 182
- cairo_dock_fm_get_file_properties
 - cairo-dock-file-manager.h, 182
- cairo_dock_fm_get_pid
 - cairo-dock-file-manager.h, 187
- cairo_dock_fm_get_trash_path
 - cairo-dock-file-manager.h, 185
- cairo_dock_fm_is_mounted
 - cairo-dock-file-manager.h, 184
- cairo_dock_fm_launch_uri
 - cairo-dock-file-manager.h, 183
- cairo_dock_fm_list_apps_for_file
 - cairo-dock-file-manager.h, 185
- cairo_dock_fm_list_directory
 - cairo-dock-file-manager.h, 182
- cairo_dock_fm_lock_screen
 - cairo-dock-file-manager.h, 186
- cairo_dock_fm_logout
 - cairo-dock-file-manager.h, 185
- cairo_dock_fm_measure_directory
 - cairo-dock-file-manager.h, 182
- cairo_dock_fm_monitor_pid
 - cairo-dock-file-manager.h, 187
- cairo_dock_fm_mount_full
 - cairo-dock-file-manager.h, 183
- cairo_dock_fm_move_file
 - cairo-dock-file-manager.h, 184
- cairo_dock_fm_reboot
 - cairo-dock-file-manager.h, 186
- cairo_dock_fm_register_vfs_backend
 - cairo-dock-file-manager.h, 182
- cairo_dock_fm_remove_monitor_full
 - cairo-dock-file-manager.h, 183
- cairo_dock_fm_rename_file
 - cairo-dock-file-manager.h, 184
- cairo_dock_fm_setup_time
 - cairo-dock-file-manager.h, 186
- cairo_dock_fm_show_system_monitor
 - cairo-dock-file-manager.h, 186
- cairo_dock_fm_shutdown
 - cairo-dock-file-manager.h, 186
- cairo_dock_fm_unmount_full
 - cairo-dock-file-manager.h, 183
- cairo_dock_foreach_appli_icon
 - cairo-dock-applications-manager.h, 100
- cairo_dock_free_gl_font
 - cairo-dock-opengl-font.h, 248
- cairo_dock_free_gl_path
 - cairo-dock-opengl-path.h, 252
- cairo_dock_free_image_buffer
 - cairo-dock-image-buffer.h, 220
- cairo_dock_free_package
 - cairo-dock-packages.h, 272
- cairo_dock_free_particle_system
 - cairo-dock-particle-system.h, 275
- cairo_dock_get_animation_delta_t
 - cairo-dock-animations.h, 53
- cairo_dock_get_appli_icon
 - cairo-dock-applications-manager.h, 100
- cairo_dock_get_available_docks
 - cairo-dock-dock-facility.h, 158
- cairo_dock_get_available_docks_for_icon
 - cairo-dock-dock-facility.h, 157
- cairo_dock_get_class_app_info
 - cairo-dock-class-manager.h, 104
- cairo_dock_get_conf_file_version
 - cairo-dock-keyfile-utilities.h, 227
- cairo_dock_get_current_active_icon
 - cairo-dock-applications-manager.h, 100
- cairo_dock_get_current_applis_list
 - cairo-dock-applications-manager.h, 99
- cairo_dock_get_current_dock_width_linear

- cairo-dock-dock-facility.h, 159
- cairo_dock_get_current_icon_size
 - cairo-dock-icon-facility.h, 206
- cairo_dock_get_default_data_renderer_font
 - cairo-dock-data-renderer-manager.h, 119
 - cairo-dock-data-renderer.h, 126
- cairo_dock_get_default_terminal
 - cairo-dock-utils.h, 304
- cairo_dock_get_file_size
 - cairo-dock-file-manager.h, 186
- cairo_dock_get_first_drawn_element_linear
 - cairo-dock-dock-facility.h, 161
- cairo_dock_get_first_icon
 - cairo-dock-icon-facility.h, 200
- cairo_dock_get_first_icon_of_group
 - cairo-dock-icon-facility.h, 201
- cairo_dock_get_first_icon_of_order
 - cairo-dock-icon-facility.h, 201
- cairo_dock_get_gl_text_extent
 - cairo-dock-opengl-font.h, 249
- cairo_dock_get_human_readable_size
 - cairo-dock-applet-facility.h, 97
- cairo_dock_get_icon_data_renderer
 - cairo-dock-data-renderer.h, 120
- cairo_dock_get_icon_extent
 - cairo-dock-icon-facility.h, 205
- cairo_dock_get_icon_order
 - cairo-dock-icon-facility.h, 196
- cairo_dock_get_icon_type
 - cairo-dock-icon-facility.h, 198
- cairo_dock_get_icon_with_base_uri
 - cairo-dock-icon-facility.h, 204
- cairo_dock_get_icon_with_command
 - cairo-dock-icon-facility.h, 204
- cairo_dock_get_icon_with_name
 - cairo-dock-icon-facility.h, 205
- cairo_dock_get_icon_with_subdock
 - cairo-dock-icon-facility.h, 205
- cairo_dock_get_last_icon
 - cairo-dock-icon-facility.h, 200
- cairo_dock_get_last_icon_of_group
 - cairo-dock-icon-facility.h, 201
- cairo_dock_get_last_icon_of_order
 - cairo-dock-icon-facility.h, 203
- cairo_dock_get_next_element
 - cairo-dock-icon-facility.h, 196
- cairo_dock_get_next_icon
 - cairo-dock-icon-facility.h, 203
- cairo_dock_get_overlay_image_buffer
 - cairo-dock-overlay.h, 263
- cairo_dock_get_package_path
 - cairo-dock-packages.h, 273
- cairo_dock_get_pointed_icon
 - cairo-dock-icon-facility.h, 203
- cairo_dock_get_previous_element
 - cairo-dock-icon-facility.h, 197
- cairo_dock_get_previous_icon
 - cairo-dock-icon-facility.h, 204
- cairo_dock_get_session_connection
 - cairo-dock-dbus.h, 128
- cairo_dock_get_slow_animation_delta_t
 - cairo-dock-animations.h, 53
- cairo_dock_get_transition_count
 - cairo-dock-animations.h, 54
- cairo_dock_get_transition_elapsed_time
 - cairo-dock-animations.h, 54
- cairo_dock_get_transition_fraction
 - cairo-dock-animations.h, 54
- cairo_dock_get_url_data
 - cairo-dock-packages.h, 268
- cairo_dock_get_url_data_async
 - cairo-dock-packages.h, 271
- cairo_dock_get_url_data_with_post
 - cairo-dock-packages.h, 271
- cairo_dock_get_version_from_string
 - cairo-dock-utils.h, 303
- cairo_dock_gl_path_arc
 - cairo-dock-opengl-path.h, 256
- cairo_dock_gl_path_curve_to
 - cairo-dock-opengl-path.h, 253
- cairo_dock_gl_path_line_to
 - cairo-dock-opengl-path.h, 253
- cairo_dock_gl_path_move_to
 - cairo-dock-opengl-path.h, 252
- cairo_dock_gl_path_rel_curve_to
 - cairo-dock-opengl-path.h, 255
- cairo_dock_gl_path_rel_line_to
 - cairo-dock-opengl-path.h, 253
- cairo_dock_gl_path_rel_simple_curve_to
 - cairo-dock-opengl-path.h, 256
- cairo_dock_gl_path_set_extent
 - cairo-dock-opengl-path.h, 252
- cairo_dock_gl_path_simple_curve_to
 - cairo-dock-opengl-path.h, 255
- CAIRO_DOCK_GRAPH_BAR
 - cairo-dock-graph.h, 310
- CAIRO_DOCK_GRAPH_CIRCLE
 - cairo-dock-graph.h, 310
- CAIRO_DOCK_GRAPH_CIRCLE_PLAIN
 - cairo-dock-graph.h, 310
- CAIRO_DOCK_GRAPH_LINE
 - cairo-dock-graph.h, 310
- CAIRO_DOCK_GRAPH_PLAIN
 - cairo-dock-graph.h, 310
- cairo_dock_gui_find_group_key_widget_in_list
 - cairo-dock-gui-factory.h, 192
- cairo_dock_gui_image_from_file
 - cairo-dock-gui-factory.h, 193
- cairo_dock_gui_menu_item_add
 - cairo-dock-gui-factory.h, 192
- cairo_dock_has_transition
 - cairo-dock-animations.h, 53
- cairo_dock_icon_buffer_to_cairo
 - cairo-dock-icon-facility.h, 208
- cairo_dock_icon_is_being_inserted
 - cairo-dock-icon-facility.h, 196

- cairo_dock_icon_is_being_removed
 - cairo-dock-icon-facility.h, 196
- cairo_dock_image_buffer_copy_scale
 - cairo-dock-image-buffer.h, 221
- cairo_dock_import_theme
 - cairo-dock-themes-manager.h, 299
- cairo_dock_import_theme_async
 - cairo-dock-themes-manager.h, 300
- CAIRO_DOCK_INFO_NONE
 - cairo-dock-applet-facility.h, 96
- CAIRO_DOCK_INFO_ON_ICON
 - cairo-dock-applet-facility.h, 96
- CAIRO_DOCK_INFO_ON_LABEL
 - cairo-dock-applet-facility.h, 96
- CAIRO_DOCK_IS_APPLET
 - cairo-dock-icon-factory.h, 210
- CAIRO_DOCK_IS_APPLI
 - cairo-dock-icon-factory.h, 210
- CAIRO_DOCK_IS_AUTOMATIC_SEPARATOR
 - cairo-dock-icon-factory.h, 211
- CAIRO_DOCK_IS_CONTAINER
 - cairo-dock-container.h, 110
- CAIRO_DOCK_IS_DETACHABLE_APPLET
 - cairo-dock-icon-factory.h, 212
- CAIRO_DOCK_IS_DIALOG
 - cairo-dock-dialog-factory.h, 148
- CAIRO_DOCK_IS_ICON
 - cairo-dock-icon-factory.h, 210
- cairo_dock_is_loading
 - cairo-dock-config.h, 107
- CAIRO_DOCK_IS_MULTI_APPLI
 - cairo-dock-icon-factory.h, 211
- CAIRO_DOCK_IS_NORMAL_APPLI
 - cairo-dock-icon-factory.h, 211
- CAIRO_DOCK_IS_USER_SEPARATOR
 - cairo-dock-icon-factory.h, 211
- CAIRO_DOCK_KEEP_RATIO
 - cairo-dock-surface-factory.h, 284
- cairo_dock_launch_animation
 - cairo-dock-animations.h, 55
- cairo_dock_launch_command_argv_full
 - cairo-dock-utils.h, 304
- cairo_dock_launch_command_argv_full2
 - cairo-dock-utils.h, 304
- cairo_dock_list_packages
 - cairo-dock-packages.h, 272
- cairo_dock_list_packages_async
 - cairo-dock-packages.h, 272
- cairo_dock_load_current_theme
 - cairo-dock-config.h, 107
- cairo_dock_load_gdk_pixbuf
 - cairo-dock-surface-factory.h, 285
- cairo_dock_load_gdk_pixbuf_with_max_size
 - cairo-dock-surface-factory.h, 286
- cairo_dock_load_icon_buffers
 - cairo-dock-icon-factory.h, 214
- cairo_dock_load_icon_image
 - cairo-dock-icon-factory.h, 213
- cairo_dock_load_icon_quickinfo
 - cairo-dock-icon-factory.h, 213
- cairo_dock_load_icon_text
 - cairo-dock-icon-factory.h, 213
- cairo_dock_load_image_buffer
 - cairo-dock-image-buffer.h, 217
- cairo_dock_load_image_buffer_from_surface
 - cairo-dock-image-buffer.h, 219
- cairo_dock_load_image_buffer_full
 - cairo-dock-image-buffer.h, 218
- cairo_dock_load_textured_font
 - cairo-dock-opengl-font.h, 248
- cairo_dock_load_textured_font_from_image
 - cairo-dock-opengl-font.h, 248
- CAIRO_DOCK_LOCAL_PACKAGE
 - cairo-dock-packages.h, 269
- cairo_dock_merge_conf_files
 - cairo-dock-keyfile-utilities.h, 225
- cairo_dock_new_gl_path
 - cairo-dock-opengl-path.h, 251
- CAIRO_DOCK_NEW_PACKAGE
 - cairo-dock-packages.h, 269
- cairo_dock_open_key_file
 - cairo-dock-keyfile-utilities.h, 225
- CAIRO_DOCK_ORIENTATION_HFLIP
 - cairo-dock-surface-factory.h, 284
- CAIRO_DOCK_ORIENTATION_ROT_180
 - cairo-dock-surface-factory.h, 284
- CAIRO_DOCK_ORIENTATION_ROT_270
 - cairo-dock-surface-factory.h, 284
- CAIRO_DOCK_ORIENTATION_ROT_90
 - cairo-dock-surface-factory.h, 284
- CAIRO_DOCK_ORIENTATION_ROT_90_HFLIP
 - cairo-dock-surface-factory.h, 284
- CAIRO_DOCK_ORIENTATION_ROT_90_VFLIP
 - cairo-dock-surface-factory.h, 284
- CAIRO_DOCK_ORIENTATION_VFLIP
 - cairo-dock-surface-factory.h, 284
- cairo_dock_package_current_theme
 - cairo-dock-themes-manager.h, 298
- cairo_dock_play_sound
 - cairo-dock-applet-facility.h, 97
- cairo_dock_pop_down
 - cairo-dock-animations.h, 55
- cairo_dock_pop_up
 - cairo-dock-animations.h, 55
- cairo_dock_print_overlay_on_icon_from_image
 - cairo-dock-overlay.h, 265
- cairo_dock_print_overlay_on_icon_from_surface
 - cairo-dock-overlay.h, 267
- cairo_dock_redraw_container
 - cairo-dock-container.h, 116
- cairo_dock_redraw_container_area
 - cairo-dock-container.h, 116
- cairo_dock_redraw_icon
 - cairo-dock-container.h, 117
- CAIRO_DOCK_REDRAW_MY_CONTAINER
 - cairo-dock-applet-facility.h, 83

- cairo_dock_refresh_data_renderer
 - cairo-dock-data-renderer.h, 127
- cairo_dock_register_class
 - cairo-dock-class-manager.h, 106
- cairo_dock_register_class2
 - cairo-dock-class-manager.h, 105
- cairo_dock_register_service_name
 - cairo-dock-dbus.h, 128
- cairo_dock_reload_buffers_in_all_docks
 - cairo-dock-dock-manager.h, 168
- cairo_dock_reload_current_module_widget
 - cairo-dock-gui-manager.h, 194
- cairo_dock_reload_data_renderer_on_icon
 - cairo-dock-data-renderer.h, 127
- cairo_dock_remove_data_renderer_on_icon
 - cairo-dock-data-renderer.h, 126
- cairo_dock_remove_group_key_from_conf_file
 - cairo-dock-keyfile-utilities.h, 228
- cairo_dock_remove_html_spaces
 - cairo-dock-utils.h, 303
- cairo_dock_remove_icons_from_dock
 - cairo-dock-dock-factory.h, 164
- cairo_dock_remove_overlay_at_position
 - cairo-dock-overlay.h, 265
- cairo_dock_remove_transition_on_icon
 - cairo-dock-animations.h, 57
- cairo_dock_remove_version_from_string
 - cairo-dock-utils.h, 302
- cairo_dock_render_new_data_on_icon
 - cairo-dock-data-renderer.h, 126
- cairo_dock_render_one_icon
 - cairo-dock-draw.h, 179
- cairo_dock_render_one_icon_opengl
 - cairo-dock-draw-opengl.h, 175
- cairo_dock_render_particles
 - cairo-dock-particle-system.h, 274
- cairo_dock_render_particles_full
 - cairo-dock-particle-system.h, 274
- cairo_dock_resize_data_renderer_history
 - cairo-dock-data-renderer.h, 127
- cairo_dock_rotate_surface
 - cairo-dock-surface-factory.h, 288
- cairo_dock_search_icon_pointing_on_dock
 - cairo-dock-dock-manager.h, 167
- cairo_dock_search_icon_s_path
 - cairo-dock-icon-manager.h, 215
- cairo_dock_search_icon_size
 - cairo-dock-icon-manager.h, 215
- cairo_dock_search_image_s_path
 - cairo-dock-image-buffer.h, 218
- cairo_dock_set_data_from_class
 - cairo-dock-class-manager.h, 106
- cairo_dock_set_icon_always_visible
 - cairo-dock-icon-facility.h, 197
- cairo_dock_set_icon_static
 - cairo-dock-icon-facility.h, 197
- cairo_dock_set_icon_surface
 - cairo-dock-applet-facility.h, 68
- cairo_dock_set_icon_surface_full
 - cairo-dock-applet-facility.h, 96
- cairo_dock_set_image_on_icon
 - cairo-dock-applet-facility.h, 96
- cairo_dock_set_image_on_icon_with_default
 - cairo-dock-applet-facility.h, 97
- cairo_dock_set_overlay_scale
 - cairo-dock-overlay.h, 263
- cairo_dock_set_paths
 - cairo-dock-themes-manager.h, 300
- cairo_dock_set_status_message
 - cairo-dock-gui-manager.h, 194
- cairo_dock_set_status_message_printf
 - cairo-dock-gui-manager.h, 194
- cairo_dock_set_transition_on_icon
 - cairo-dock-animations.h, 57
- cairo_dock_show_subdock
 - cairo-dock-dock-facility.h, 158
- cairo_dock_sort_icons_by_name
 - cairo-dock-icon-facility.h, 200
- cairo_dock_sort_icons_by_order
 - cairo-dock-icon-facility.h, 199
- cairo_dock_start_applications_manager
 - cairo-dock-applications-manager.h, 99
- cairo_dock_string_is_address
 - cairo-dock-utils.h, 303
- cairo_dock_stroke_gl_path
 - cairo-dock-opengl-path.h, 257
- cairo_dock_trigger_icon_removal_from_dock
 - cairo-dock-animations.h, 56
- cairo_dock_trigger_shortkey
 - cairo-dock-keybinder.h, 224
- cairo_dock_unload_image_buffer
 - cairo-dock-image-buffer.h, 219
- cairo_dock_update_conf_file
 - cairo-dock-themes-manager.h, 297
- cairo_dock_update_default_particle_system
 - cairo-dock-particle-system.h, 275
- cairo_dock_update_dock_size
 - cairo-dock-dock-facility.h, 158
- cairo_dock_update_icon_texture
 - cairo-dock-draw-opengl.h, 177
- cairo_dock_update_keyfile
 - cairo-dock-keyfile-utilities.h, 228
- CAIRO_DOCK_UPDATED_PACKAGE
 - cairo-dock-packages.h, 269
- cairo_dock_upgrade_conf_file_full
 - cairo-dock-keyfile-utilities.h, 227
- CAIRO_DOCK_USER_PACKAGE
 - cairo-dock-packages.h, 269
- CAIRO_DOCK_WIDGET_ANIMATION_LIST
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_CHECK_BUTTON
 - cairo-dock-gui-factory.h, 190
- CAIRO_DOCK_WIDGET_CHECK_CONTROL_BUTTON
 - cairo-dock-gui-factory.h, 190
- CAIRO_DOCK_WIDGET_CLASS_SELECTOR
 - cairo-dock-gui-factory.h, 191

- CAIRO_DOCK_WIDGET_COLOR_SELECTOR_RGB
 - cairo-dock-gui-factory.h, 190
- CAIRO_DOCK_WIDGET_COLOR_SELECTOR_RGBA
 - cairo-dock-gui-factory.h, 190
- CAIRO_DOCK_WIDGET_DESKLET_DECORATION_LIST
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_DESKLET_DECORATION_LIST_WIDGET_DECAL
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_DIALOG_DECORATOR_LIST
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_DOCK_LIST
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_EMPTY_FULL
 - cairo-dock-gui-factory.h, 192
- CAIRO_DOCK_WIDGET_EMPTY_WIDGET
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_EXPANDER
 - cairo-dock-gui-factory.h, 192
- CAIRO_DOCK_WIDGET_FILE_SELECTOR
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_FOLDER_SELECTOR
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_FONT_SELECTOR
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_FRAME
 - cairo-dock-gui-factory.h, 192
- CAIRO_DOCK_WIDGET_HANDBOOK
 - cairo-dock-gui-factory.h, 192
- CAIRO_DOCK_WIDGET_HSCALE_DOUBLE
 - cairo-dock-gui-factory.h, 190
- CAIRO_DOCK_WIDGET_HSCALE_INTEGER
 - cairo-dock-gui-factory.h, 190
- CAIRO_DOCK_WIDGET_ICON_THEME_LIST
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_ICONS_LIST
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_IMAGE_SELECTOR
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_JUMP_TO_MODULE
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_JUMP_TO_MODULE_IF_EXISTS
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_LAUNCH_COMMAND
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_LAUNCH_COMMAND_IF_CONDITION
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_LINK
 - cairo-dock-gui-factory.h, 192
- CAIRO_DOCK_WIDGET_LIST
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_LIST_WITH_ENTRY
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_NUMBERED_CONTROL_LIST
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_NUMBERED_CONTROL_LIST_WIDGET
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_NUMBERED_LIST
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_PASSWORD_ENTRY
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_SCREENSHOTS_LIST
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_SEPARATOR
 - cairo-dock-gui-factory.h, 192
- CAIRO_DOCK_WIDGET_SHORTKEY_SELECTOR
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_SIZE_INTEGER
 - cairo-dock-gui-factory.h, 190
- CAIRO_DOCK_WIDGET_SOUND_SELECTOR
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_SPIN_DOUBLE
 - cairo-dock-gui-factory.h, 190
- CAIRO_DOCK_WIDGET_SPIN_INTEGER
 - cairo-dock-gui-factory.h, 190
- CAIRO_DOCK_WIDGET_STRING_ENTRY
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_TEXT_LABEL
 - cairo-dock-gui-factory.h, 192
- CAIRO_DOCK_WIDGET_THEME_LIST
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_TREE_VIEW_MULTI_CHOICE
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_TREE_VIEW_SORT
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_TREE_VIEW_SORT_AND_MODIFY
 - cairo-dock-gui-factory.h, 191
- CAIRO_DOCK_WIDGET_VIEW_LIST
 - cairo-dock-gui-factory.h, 190
- cairo_dock_write_keys_to_conf_file
 - cairo-dock-themes-manager.h, 297
- cairo_dock_write_keys_to_file_full
 - cairo-dock-keyfile-utilities.h, 225
- cairo_dock_write_keys_to_new_conf_file
 - cairo-dock-themes-manager.h, 298
- cairo_dock_write_keys_to_new_file
 - cairo-dock-keyfile-utilities.h, 225
- CairoDeskletNotifications
 - cairo-dock-desklet-manager.h, 139
- CairoDeskletVisibility
 - cairo-dock-desklet-factory.h, 135
- CairoDesktopNotifications
 - cairo-dock-desktop-manager.h, 143
- CairoDockGUIWidgetType
 - cairo-dock-gui-factory.h, 190
- CairoDockInfoDisplay
 - cairo-dock-applet-facility.h, 96
- CairoDockLoadImageModifier
 - cairo-dock-surface-factory.h, 283
- CairoDockPackageType
 - cairo-dock-packages.h, 269
- CairoDocksNotifications
 - cairo-dock-dock-manager.h, 166
- CairoDockTypeGraph
 - cairo-dock-graph.h, 310
- CairoIconNotifications
 - cairo-dock-icon-manager.h, 214

- CD_APPLET_ADD_DATA_RENDERER_ON_MY_ICON
cairo-dock-applet-facility.h, [92](#)
- CD_APPLET_ADD_ICON_IN_MY_ICONS_LIST
cairo-dock-applet-facility.h, [94](#)
- CD_APPLET_ADD_IN_MENU
cairo-dock-applet-facility.h, [79](#)
- CD_APPLET_ADD_IN_MENU_WITH_DATA
cairo-dock-applet-facility.h, [79](#)
- CD_APPLET_ADD_IN_MENU_WITH_STOCK
cairo-dock-applet-facility.h, [79](#)
- CD_APPLET_ADD_IN_MENU_WITH_STOCK_AND_DATA
cairo-dock-applet-facility.h, [78](#)
- CD_APPLET_ADD_IN_MENU_WITH_TOOLTIP_AND_DATA
cairo-dock-applet-facility.h, [78](#)
- CD_APPLET_ADD_OVERLAY_ON_MY_ICON
cairo-dock-applet-facility.h, [91](#)
- CD_APPLET_ADD_SEPARATOR_IN_MENU
cairo-dock-applet-facility.h, [79](#)
- CD_APPLET_ADD_SUB_MENU
cairo-dock-applet-facility.h, [77](#)
- CD_APPLET_ADD_SUB_MENU_WITH_IMAGE
cairo-dock-applet-facility.h, [77](#)
- CD_APPLET_ALLOW_NO_CLICKABLE_DESKLET
cairo-dock-applet-facility.h, [93](#)
- CD_APPLET_ALT_CLICK
cairo-dock-applet-facility.h, [81](#)
- CD_APPLET_ANIMATE_MY_ICON
cairo-dock-applet-facility.h, [88](#)
- CD_APPLET_BIND_KEY
cairo-dock-applet-facility.h, [82](#)
- CD_APPLET_CLICKED_CONTAINER
cairo-dock-applet-facility.h, [81](#)
- CD_APPLET_CLICKED_ICON
cairo-dock-applet-facility.h, [81](#)
- CD_APPLET_CTRL_CLICK
cairo-dock-applet-facility.h, [81](#)
- CD_APPLET_DEFINE2_ALL_BEGIN
cairo-dock-applet-canvas.h, [60](#)
- CD_APPLET_DEFINE_ALL_BEGIN
cairo-dock-applet-canvas.h, [59](#)
- CD_APPLET_DEFINE_END
cairo-dock-applet-canvas.h, [59](#)
- CD_APPLET_DEFINITION
cairo-dock-applet-canvas.h, [59](#)
- CD_APPLET_DELETE_MY_ICONS_LIST
cairo-dock-applet-facility.h, [93](#)
- CD_APPLET_DEMANDS_ATTENTION
cairo-dock-applet-facility.h, [89](#)
- CD_APPLET_DETACH_ICON_FROM_MY_ICONS_LIST
cairo-dock-applet-facility.h, [94](#)
- CD_APPLET_FINISH_DRAWING_MY_ICON
cairo-dock-applet-facility.h, [90](#)
- CD_APPLET_FINISH_DRAWING_MY_ICON_CAIRO
cairo-dock-applet-facility.h, [90](#)
- CD_APPLET_GET_CONFIG_ALL_BEGIN
cairo-dock-applet-canvas.h, [61](#)
- CD_APPLET_GET_CONFIG_END
cairo-dock-applet-canvas.h, [61](#)
- CD_APPLET_GET_MY_ICON_EXTENT
cairo-dock-applet-facility.h, [89](#)
- CD_APPLET_INIT_ALL_BEGIN
cairo-dock-applet-canvas.h, [60](#)
- CD_APPLET_INIT_END
cairo-dock-applet-canvas.h, [60](#)
- CD_APPLET_LOAD_MY_ICONS_LIST
cairo-dock-applet-facility.h, [94](#)
- CD_APPLET_LOAD_SURFACE_FOR_MY_APPLET
cairo-dock-applet-facility.h, [83](#)
- CD_APPLET_LOAD_SURFACE_FOR_MY_APPLET_WITH_DEFAULT
cairo-dock-applet-facility.h, [83](#)
- CD_APPLET_MANAGE_APPLICATION
cairo-dock-applet-facility.h, [95](#)
- CD_APPLET_MY_CONF_FILE
cairo-dock-applet-facility.h, [80](#)
- CD_APPLET_MY_CONFIG_CHANGED
cairo-dock-applet-facility.h, [80](#)
- CD_APPLET_MY_CONTAINER_IS_OPENGL
cairo-dock-applet-facility.h, [92](#)
- CD_APPLET_MY_CONTAINER_TYPE_CHANGED
cairo-dock-applet-facility.h, [81](#)
- CD_APPLET_MY_ICONS_LIST
cairo-dock-applet-facility.h, [95](#)
- CD_APPLET_MY_ICONS_LIST_CONTAINER
cairo-dock-applet-facility.h, [95](#)
- CD_APPLET_MY_KEY_FILE
cairo-dock-applet-facility.h, [80](#)
- CD_APPLET_MY_MENU
cairo-dock-applet-facility.h, [81](#)
- CD_APPLET_MY_OLD_CONTAINER
cairo-dock-applet-facility.h, [81](#)
- CD_APPLET_ON_BUILD_MENU_BEGIN
cairo-dock-applet-canvas.h, [62](#)
- CD_APPLET_ON_BUILD_MENU_END
cairo-dock-applet-canvas.h, [62](#)
- CD_APPLET_ON_CLICK_BEGIN
cairo-dock-applet-canvas.h, [62](#)
- CD_APPLET_ON_CLICK_END
cairo-dock-applet-canvas.h, [62](#)
- CD_APPLET_ON_DOUBLE_CLICK_BEGIN
cairo-dock-applet-canvas.h, [62](#)
- CD_APPLET_ON_DOUBLE_CLICK_END
cairo-dock-applet-canvas.h, [62](#)
- CD_APPLET_ON_DROP_DATA_BEGIN
cairo-dock-applet-canvas.h, [63](#)
- CD_APPLET_ON_DROP_DATA_END
cairo-dock-applet-canvas.h, [63](#)
- CD_APPLET_ON_MIDDLE_CLICK_BEGIN
cairo-dock-applet-canvas.h, [62](#)
- CD_APPLET_ON_MIDDLE_CLICK_END
cairo-dock-applet-canvas.h, [62](#)
- CD_APPLET_ON_SCROLL_BEGIN
cairo-dock-applet-canvas.h, [63](#)
- CD_APPLET_ON_SCROLL_END
cairo-dock-applet-canvas.h, [63](#)
- CD_APPLET_ON_UPDATE_ICON_BEGIN
cairo-dock-applet-canvas.h, [63](#)

- CD_APPLET_ON_UPDATE_ICON_END
cairo-dock-applet-canvas.h, [63](#)
- CD_APPLET_PAUSE_UPDATE_ICON
cairo-dock-applet-canvas.h, [64](#)
- CD_APPLET_POPUP_MENU_ON_MY_ICON
cairo-dock-applet-facility.h, [80](#)
- CD_APPLET_PRINT_OVERLAY_ON_MY_ICON
cairo-dock-applet-facility.h, [91](#)
- CD_APPLET_RECEIVED_DATA
cairo-dock-applet-facility.h, [82](#)
- CD_APPLET_REDRAW_MY_ICON
cairo-dock-applet-facility.h, [82](#)
- CD_APPLET_REGISTER_FOR_BUILD_MENU_EVENT
cairo-dock-applet-canvas.h, [64](#)
- CD_APPLET_REGISTER_FOR_CLICK_EVENT
cairo-dock-applet-canvas.h, [64](#)
- CD_APPLET_REGISTER_FOR_DOUBLE_CLICK_EVENT
cairo-dock-applet-canvas.h, [64](#)
- CD_APPLET_REGISTER_FOR_DROP_DATA_EVENT
cairo-dock-applet-canvas.h, [65](#)
- CD_APPLET_REGISTER_FOR_MIDDLE_CLICK_EVENT
cairo-dock-applet-canvas.h, [64](#)
- CD_APPLET_REGISTER_FOR_SCROLL_EVENT
cairo-dock-applet-canvas.h, [65](#)
- CD_APPLET_REGISTER_FOR_UPDATE_ICON_EVENT
cairo-dock-applet-canvas.h, [65](#)
- CD_APPLET_REGISTER_FOR_UPDATE_ICON_SLOW_EVENT
cairo-dock-applet-canvas.h, [65](#)
- CD_APPLET_RELOAD_ALL_BEGIN
cairo-dock-applet-canvas.h, [61](#)
- CD_APPLET_RELOAD_CONFIG_PANEL
cairo-dock-applet-facility.h, [80](#)
- CD_APPLET_RELOAD_CONFIG_PANEL_WITH_PAGE
cairo-dock-applet-facility.h, [80](#)
- CD_APPLET_RELOAD_END
cairo-dock-applet-canvas.h, [61](#)
- CD_APPLET_RELOAD_MY_DATA_RENDERER
cairo-dock-applet-facility.h, [92](#)
- CD_APPLET_REMOVE_ICON_FROM_MY_ICONS_LIST
cairo-dock-applet-facility.h, [93](#)
- CD_APPLET_REMOVE_MY_DATA_RENDERER
cairo-dock-applet-facility.h, [92](#)
- CD_APPLET_REMOVE_OVERLAY_ON_MY_ICON
cairo-dock-applet-facility.h, [91](#)
- CD_APPLET_RENDER_NEW_DATA_ON_MY_ICON
cairo-dock-applet-facility.h, [92](#)
- CD_APPLET_RESET_CONFIG_ALL_BEGIN
cairo-dock-applet-canvas.h, [61](#)
- CD_APPLET_RESET_CONFIG_ALL_END
cairo-dock-applet-canvas.h, [61](#)
- CD_APPLET_RESET_DATA_ALL_END
cairo-dock-applet-canvas.h, [61](#)
- CD_APPLET_RESET_DATA_BEGIN
cairo-dock-applet-canvas.h, [61](#)
- CD_APPLET_SCROLL_DOWN
cairo-dock-applet-facility.h, [82](#)
- CD_APPLET_SCROLL_UP
cairo-dock-applet-facility.h, [82](#)
- CD_APPLET_SET_ALWAYS_VISIBLE_ICON
cairo-dock-applet-facility.h, [88](#)
- CD_APPLET_SET_DEFAULT_IMAGE_ON_MY_ICON_IF_NONE
cairo-dock-applet-facility.h, [84](#)
- CD_APPLET_SET_DESKLET_RENDERER
cairo-dock-applet-facility.h, [93](#)
- CD_APPLET_SET_DESKLET_RENDERER_WITH_DATA
cairo-dock-applet-facility.h, [93](#)
- CD_APPLET_SET_HOURS_MINUTES_AS_QUICK_INFO
cairo-dock-applet-facility.h, [86](#)
- CD_APPLET_SET_IMAGE_ON_MY_ICON
cairo-dock-applet-facility.h, [84](#)
- CD_APPLET_SET_MINUTES_SECONDES_AS_QUICK_INFO
cairo-dock-applet-facility.h, [88](#)
- CD_APPLET_SET_MY_DATA_RENDERER_HISTORY_TO_MAX
cairo-dock-applet-facility.h, [92](#)
- CD_APPLET_SET_NAME_FOR_MY_ICON
cairo-dock-applet-facility.h, [84](#)
- CD_APPLET_SET_NAME_FOR_MY_ICON_PRINTF
cairo-dock-applet-facility.h, [86](#)
- CD_APPLET_SET_QUICK_INFO_ON_MY_ICON
cairo-dock-applet-facility.h, [86](#)
- CD_APPLET_SET_QUICK_INFO_ON_MY_ICON_PRINTF
cairo-dock-applet-facility.h, [86](#)
- CD_APPLET_SET_SIZE_AS_QUICK_INFO
cairo-dock-applet-facility.h, [88](#)
- CD_APPLET_SET_STATIC_DESKLET
cairo-dock-applet-facility.h, [93](#)
- CD_APPLET_SET_STATIC_ICON
cairo-dock-applet-facility.h, [88](#)
- CD_APPLET_SET_SURFACE_ON_MY_ICON
cairo-dock-applet-facility.h, [83](#)
- CD_APPLET_SET_USER_IMAGE_ON_MY_ICON
cairo-dock-applet-facility.h, [84](#)
- CD_APPLET_SHIFT_CLICK
cairo-dock-applet-facility.h, [81](#)
- CD_APPLET_SKIP_UPDATE_ICON
cairo-dock-applet-canvas.h, [63](#)
- CD_APPLET_START_DRAWING_MY_ICON
cairo-dock-applet-facility.h, [90](#)
- CD_APPLET_START_DRAWING_MY_ICON_CAIRO
cairo-dock-applet-facility.h, [90](#)
- CD_APPLET_START_DRAWING_MY_ICON_OR_RETURN
cairo-dock-applet-facility.h, [90](#)
- CD_APPLET_START_DRAWING_MY_ICON_OR_RETURN_CAIRO
cairo-dock-applet-facility.h, [90](#)
- CD_APPLET_STOP_ANIMATING_MY_ICON
cairo-dock-applet-facility.h, [89](#)
- CD_APPLET_STOP_BEGIN
cairo-dock-applet-canvas.h, [60](#)
- CD_APPLET_STOP_DEMANDING_ATTENTION
cairo-dock-applet-facility.h, [89](#)
- CD_APPLET_STOP_END
cairo-dock-applet-canvas.h, [60](#)
- CD_APPLET_STOP_UPDATE_ICON
cairo-dock-applet-canvas.h, [63](#)
- CD_APPLET_UNREGISTER_FOR_BUILD_MENU_EVENT
cairo-dock-applet-canvas.h, [64](#)

- CD_APPLET_UNREGISTER_FOR_CLICK_EVENT
 - cairo-dock-applet-canvas.h, [64](#)
- CD_APPLET_UNREGISTER_FOR_DOUBLE_CLICK_EVENT
 - cairo-dock-applet-canvas.h, [65](#)
- CD_APPLET_UNREGISTER_FOR_DROP_DATA_EVENT
 - cairo-dock-applet-canvas.h, [65](#)
- CD_APPLET_UNREGISTER_FOR_MIDDLE_CLICK_EVENT
 - cairo-dock-applet-canvas.h, [64](#)
- CD_APPLET_UNREGISTER_FOR_SCROLL_EVENT
 - cairo-dock-applet-canvas.h, [65](#)
- CD_APPLET_UNREGISTER_FOR_UPDATE_ICON_EVENT
 - cairo-dock-applet-canvas.h, [66](#)
- CD_APPLET_UNREGISTER_FOR_UPDATE_ICON_SLOW_EVENT
 - cairo-dock-applet-canvas.h, [65](#)
- CD_APPLET_UNSET_STATIC_ICON
 - cairo-dock-applet-facility.h, [88](#)
- CD_CONFIG_GET_BOOLEAN
 - cairo-dock-applet-facility.h, [69](#)
- CD_CONFIG_GET_BOOLEAN_WITH_DEFAULT
 - cairo-dock-applet-facility.h, [69](#)
- CD_CONFIG_GET_COLOR
 - cairo-dock-applet-facility.h, [76](#)
- CD_CONFIG_GET_COLOR_RGB
 - cairo-dock-applet-facility.h, [74](#)
- CD_CONFIG_GET_COLOR_RGB_WITH_DEFAULT
 - cairo-dock-applet-facility.h, [74](#)
- CD_CONFIG_GET_COLOR_RGBA
 - cairo-dock-applet-facility.h, [74](#)
- CD_CONFIG_GET_COLOR_RGBA_WITH_DEFAULT
 - cairo-dock-applet-facility.h, [73](#)
- CD_CONFIG_GET_DOUBLE
 - cairo-dock-applet-facility.h, [71](#)
- CD_CONFIG_GET_DOUBLE_WITH_DEFAULT
 - cairo-dock-applet-facility.h, [70](#)
- CD_CONFIG_GET_FILE_PATH
 - cairo-dock-applet-facility.h, [72](#)
- CD_CONFIG_GET_GAUGE_THEME
 - cairo-dock-applet-facility.h, [76](#)
- CD_CONFIG_GET_INTEGER
 - cairo-dock-applet-facility.h, [70](#)
- CD_CONFIG_GET_INTEGER_LIST
 - cairo-dock-applet-facility.h, [71](#)
- CD_CONFIG_GET_INTEGER_WITH_DEFAULT
 - cairo-dock-applet-facility.h, [69](#)
- CD_CONFIG_GET_STRING
 - cairo-dock-applet-facility.h, [72](#)
- CD_CONFIG_GET_STRING_LIST
 - cairo-dock-applet-facility.h, [73](#)
- CD_CONFIG_GET_STRING_LIST_WITH_DEFAULT
 - cairo-dock-applet-facility.h, [72](#)
- CD_CONFIG_GET_STRING_WITH_DEFAULT
 - cairo-dock-applet-facility.h, [71](#)
- CD_CONFIG_GET_THEME_PATH
 - cairo-dock-applet-facility.h, [76](#)
- CD_CONFIG_RENAME_GROUP
 - cairo-dock-applet-facility.h, [77](#)
- D_
 - cairo-dock-applet-facility.h, [95](#)
- dock_check_if_mouse_inside_linear
 - _GldiContainerManagerBackend, [40](#)
- dock_handle_leave
 - _GldiContainerManagerBackend, [40](#)
- gldi-icon-names.h, [308](#)
- GDI_ABI_VERSION
 - cairo-dock-module-manager.h, [239](#)
- gldi_app_info_from_desktop_app_info
 - cairo-dock-class-manager.h, [103](#)
- gldi_app_info_get_desktop_action_name
 - cairo-dock-class-manager.h, [103](#)
- gldi_app_info_get_desktop_actions
 - cairo-dock-class-manager.h, [102](#)
- gldi_app_info_get_supported_types
 - cairo-dock-class-manager.h, [103](#)
- gldi_app_info_launch
 - cairo-dock-class-manager.h, [102](#)
- gldi_app_info_launch_action
 - cairo-dock-class-manager.h, [102](#)
- gldi_app_info_new_from_commandline
 - cairo-dock-class-manager.h, [101](#)
- gldi_app_info_set_run_in_terminal
 - cairo-dock-class-manager.h, [104](#)
- gldi_container_build_menu
 - cairo-dock-container.h, [117](#)
- gldi_container_calculate_aimed_point
 - cairo-dock-container.h, [115](#)
- gldi_container_calculate_aimed_point_base
 - cairo-dock-container.h, [115](#)
- gldi_container_calculate_rect
 - cairo-dock-container.h, [114](#)
- gldi_container_dock_check_if_mouse_inside_linear
 - cairo-dock-container.h, [116](#)
- gldi_container_dock_handle_leave
 - cairo-dock-container.h, [115](#)
- gldi_container_enable_drop
 - cairo-dock-container.h, [110](#)
- gldi_container_get_current_desktop_index
 - cairo-dock-container.h, [112](#)
- gldi_container_init_layer
 - cairo-dock-container.h, [113](#)
- gldi_container_is_active
 - cairo-dock-container.h, [113](#)
- gldi_container_is_wayland_backend
 - cairo-dock-container.h, [114](#)
- gldi_container_move
 - cairo-dock-container.h, [112](#)
- gldi_container_move_resize_dock
 - cairo-dock-container.h, [114](#)
- gldi_container_move_to_rect
 - cairo-dock-container.h, [114](#)
- gldi_container_notify_drop_data
 - cairo-dock-container.h, [117](#)
- gldi_container_present
 - cairo-dock-container.h, [113](#)
- gldi_container_reserve_space
 - cairo-dock-container.h, [111](#)
- gldi_container_set_keep_below

- cairo-dock-container.h, 115
- gdi_container_set_screen
 - cairo-dock-container.h, 114
- gdi_container_use_new_positioning_code
 - cairo-dock-container.h, 116
- gdi_desklet_add_interactive_widget
 - cairo-dock-desklet-factory.h, 135
- gdi_desklet_add_interactive_widget_with_margin
 - cairo-dock-desklet-factory.h, 136
- gdi_desklet_hide
 - cairo-dock-desklet-factory.h, 137
- gdi_desklet_lock_position
 - cairo-dock-desklet-factory.h, 138
- gdi_desklet_new
 - cairo-dock-desklet-factory.h, 135
- gdi_desklet_set_accessibility
 - cairo-dock-desklet-factory.h, 137
- gdi_desklet_set_margin
 - cairo-dock-desklet-factory.h, 136
- gdi_desklet_set_sticky
 - cairo-dock-desklet-factory.h, 137
- gdi_desklet_show
 - cairo-dock-desklet-factory.h, 137
- gdi_desklet_steal_interactive_widget
 - cairo-dock-desklet-factory.h, 136
- gdi_desklets_foreach
 - cairo-dock-desklet-manager.h, 139
- gdi_desklets_foreach_icons
 - cairo-dock-desklet-manager.h, 140
- gdi_desklets_set_visibility_to_default
 - cairo-dock-desklet-manager.h, 140
- gdi_desklets_set_visible
 - cairo-dock-desklet-manager.h, 140
- gdi_desktop_add_workspace
 - cairo-dock-desktop-manager.h, 146
- gdi_desktop_file_db_init
 - cairo-dock-desktop-file-db.h, 141
- gdi_desktop_file_db_lookup
 - cairo-dock-desktop-file-db.h, 141
- gdi_desktop_file_db_stop
 - cairo-dock-desktop-file-db.h, 141
- gdi_desktop_get_current
 - cairo-dock-desktop-manager.h, 146
- gdi_desktop_manager_register_backend
 - cairo-dock-desktop-manager.h, 144
- gdi_desktop_present_class
 - cairo-dock-desktop-manager.h, 144
- gdi_desktop_present_desktops
 - cairo-dock-desktop-manager.h, 145
- gdi_desktop_present_windows
 - cairo-dock-desktop-manager.h, 145
- gdi_desktop_remove_last_workspace
 - cairo-dock-desktop-manager.h, 146
- gdi_desktop_set_on_widget_layer
 - cairo-dock-desktop-manager.h, 145
- gdi_desktop_show_widget_layer
 - cairo-dock-desktop-manager.h, 145
- gdi_dialog_hide
 - cairo-dock-dialog-manager.h, 156
- gdi_dialog_leave
 - cairo-dock-dialog-manager.h, 156
- gdi_dialog_new
 - cairo-dock-dialog-factory.h, 149
- gdi_dialog_show
 - cairo-dock-dialog-factory.h, 149
- gdi_dialog_show_and_wait
 - cairo-dock-dialog-factory.h, 154
- gdi_dialog_show_general_message
 - cairo-dock-dialog-factory.h, 153
- gdi_dialog_show_temporary
 - cairo-dock-dialog-factory.h, 151
- gdi_dialog_show_temporary_with_default_icon
 - cairo-dock-dialog-factory.h, 151
- gdi_dialog_show_temporary_with_icon
 - cairo-dock-dialog-factory.h, 150
- gdi_dialog_show_temporary_with_icon_printf
 - cairo-dock-dialog-factory.h, 150
- gdi_dialog_show_with_entry
 - cairo-dock-dialog-factory.h, 152
- gdi_dialog_show_with_question
 - cairo-dock-dialog-factory.h, 151
- gdi_dialog_show_with_value
 - cairo-dock-dialog-factory.h, 153
- gdi_dialog_steal_interactive_widget
 - cairo-dock-dialog-factory.h, 154
- gdi_dialog_toggle_visibility
 - cairo-dock-dialog-manager.h, 156
- gdi_dialog_unhide
 - cairo-dock-dialog-manager.h, 156
- gdi_dialogs_remove_on_icon
 - cairo-dock-dialog-manager.h, 155
- gdi_dock_add_conf_file
 - cairo-dock-dock-manager.h, 169
- gdi_dock_add_conf_file_for_name
 - cairo-dock-dock-manager.h, 168
- gdi_dock_enter_synthetic
 - cairo-dock-dock-factory.h, 164
- gdi_dock_get
 - cairo-dock-dock-manager.h, 166
- gdi_dock_get_name
 - cairo-dock-dock-manager.h, 165
- gdi_dock_get_readable_name
 - cairo-dock-dock-manager.h, 166
- gdi_dock_has_overlapping_window
 - cairo-dock-dock-visibility.h, 170
- gdi_dock_leave_synthetic
 - cairo-dock-dock-factory.h, 164
- gdi_dock_new
 - cairo-dock-dock-factory.h, 163
- gdi_dock_rename
 - cairo-dock-dock-manager.h, 167
- gdi_dock_set_visibility
 - cairo-dock-dock-manager.h, 169
- gdi_dock_visibility_refresh
 - cairo-dock-dock-visibility.h, 170
- gdi_docks_foreach

- cairo-dock-dock-manager.h, 167
- gldi_docks_foreach_root
 - cairo-dock-dock-manager.h, 168
- gldi_docks_redraw_all_root
 - cairo-dock-dock-manager.h, 169
- gldi_get_diag_msg
 - cairo-dock-core.h, 118
- gldi_gl_backend_init
 - cairo-dock-opengl.h, 259
- gldi_gl_container_begin_draw
 - cairo-dock-opengl.h, 258
- gldi_gl_container_begin_draw_full
 - cairo-dock-opengl.h, 260
- gldi_gl_container_end_draw
 - cairo-dock-opengl.h, 260
- gldi_gl_container_init
 - cairo-dock-opengl.h, 261
- gldi_gl_container_make_current
 - cairo-dock-opengl.h, 259
- gldi_gl_container_resized
 - cairo-dock-opengl.h, 262
- gldi_gl_container_set_ortho_view
 - cairo-dock-opengl.h, 261
- gldi_gl_container_set_ortho_view_for_icon
 - cairo-dock-opengl.h, 261
- gldi_gl_container_set_perspective_view
 - cairo-dock-opengl.h, 260
- gldi_gl_container_set_perspective_view_for_icon
 - cairo-dock-opengl.h, 261
- gldi_gl_init_opengl_context
 - cairo-dock-opengl.h, 259
- gldi_gl_offscreen_context_make_current
 - cairo-dock-opengl.h, 259
- gldi_icon_is_launching
 - cairo-dock-icon-facility.h, 198
- gldi_icon_mark_as_launching
 - cairo-dock-icon-facility.h, 198
- gldi_icon_new
 - cairo-dock-icon-factory.h, 212
- gldi_icon_request_animation
 - cairo-dock-animations.h, 56
- gldi_icon_request_attention
 - cairo-dock-animations.h, 56
- gldi_icon_set_name
 - cairo-dock-icon-facility.h, 206
- gldi_icon_set_name_printf
 - cairo-dock-icon-facility.h, 207
- gldi_icon_set_quick_info
 - cairo-dock-icon-facility.h, 207
- gldi_icon_set_quick_info_printf
 - cairo-dock-icon-facility.h, 207
- gldi_icon_start_animation
 - cairo-dock-animations.h, 55
- gldi_icon_stop_animation
 - cairo-dock-animations.h, 52
- gldi_icon_stop_attention
 - cairo-dock-animations.h, 56
- gldi_icons_foreach
 - cairo-dock-icon-manager.h, 215
- gldi_icons_foreach_in_docks
 - cairo-dock-dock-manager.h, 168
- gldi_launch_desktop_app_info
 - cairo-dock-class-manager.h, 104
- GLDI_LAUNCH_GUI
 - cairo-dock-utils.h, 302
- GLDI_LAUNCH_SLICE
 - cairo-dock-utils.h, 302
- gldi_launcher_add_new
 - cairo-dock-launcher-manager.h, 229
- gldi_launcher_add_new_full
 - cairo-dock-launcher-manager.h, 229
- gldi_menu_add_item
 - cairo-dock-menu.h, 235
- gldi_menu_add_item_with_tooltip
 - cairo-dock-menu.h, 235
- gldi_menu_add_separator
 - cairo-dock-menu.h, 236
- gldi_menu_add_sub_menu
 - cairo-dock-menu.h, 232
- gldi_menu_add_sub_menu_full
 - cairo-dock-menu.h, 236
- gldi_menu_init
 - cairo-dock-menu.h, 232
- gldi_menu_item_get_image
 - cairo-dock-menu.h, 235
- gldi_menu_item_new
 - cairo-dock-menu.h, 231
- gldi_menu_item_new_full2
 - cairo-dock-menu.h, 233
- gldi_menu_item_new_with_action
 - cairo-dock-menu.h, 233
- gldi_menu_item_new_with_submenu
 - cairo-dock-menu.h, 234
- gldi_menu_item_set_image
 - cairo-dock-menu.h, 234
- gldi_menu_new
 - cairo-dock-menu.h, 232
- gldi_menu_popup_full
 - cairo-dock-menu.h, 233
- gldi_module_activate
 - cairo-dock-module-manager.h, 241
- gldi_module_deactivate
 - cairo-dock-module-manager.h, 241
- gldi_module_get
 - cairo-dock-module-manager.h, 241
- gldi_module_get_config_dir
 - cairo-dock-module-manager.h, 240
- gldi_module_instance_new
 - cairo-dock-module-instance-manager.h, 238
- gldi_module_new
 - cairo-dock-module-manager.h, 239
- gldi_module_new_from_so_file
 - cairo-dock-module-manager.h, 240
- gldi_modules_load_auto_config
 - cairo-dock-module-manager.h, 241
- gldi_modules_new_from_directory

- cairo-dock-module-manager.h, 240
- gldi_object_delete
 - cairo-dock-object.h, 245
- GLDI_OBJECT_IS_APPLET_ICON
 - cairo-dock-applet-manager.h, 98
- GLDI_OBJECT_IS_APPLI_ICON
 - cairo-dock-applications-manager.h, 99
- GLDI_OBJECT_IS_DATA_RENDERER
 - cairo-dock-data-renderer-manager.h, 118
- GLDI_OBJECT_IS_DESKLET
 - cairo-dock-desklet-factory.h, 134
- GLDI_OBJECT_IS_DOCK
 - cairo-dock-dock-factory.h, 162
- GLDI_OBJECT_IS_LAUNCHER_ICON
 - cairo-dock-launcher-manager.h, 229
- GLDI_OBJECT_IS_MANAGER
 - cairo-dock-manager.h, 230
- GLDI_OBJECT_IS_MODULE
 - cairo-dock-module-manager.h, 239
- GLDI_OBJECT_IS_MODULE_INSTANCE
 - cairo-dock-module-instance-manager.h, 237
- GLDI_OBJECT_IS_SEPARATOR_ICON
 - cairo-dock-separator-manager.h, 276
- GLDI_OBJECT_IS_STACK_ICON
 - cairo-dock-stack-icon-manager.h, 277
- GLDI_OBJECT_IS_USER_ICON
 - cairo-dock-user-icon-manager.h, 301
- gldi_object_new
 - cairo-dock-object.h, 244
- gldi_object_notify
 - cairo-dock-object.h, 244
- gldi_object_ref
 - cairo-dock-object.h, 245
- gldi_object_register_notification
 - cairo-dock-object.h, 246
- gldi_object_reload
 - cairo-dock-object.h, 245
- gldi_object_remove_notification
 - cairo-dock-object.h, 246
- gldi_object_unref
 - cairo-dock-object.h, 245
- gldi_shortkey_could_grab
 - cairo-dock-keybinder.h, 222
- gldi_shortkey_new
 - cairo-dock-keybinder.h, 223
- gldi_shortkey_rebind
 - cairo-dock-keybinder.h, 223
- gldi_style_color_get
 - cairo-dock-style-manager.h, 279
- gldi_style_color_shade
 - cairo-dock-style-facility.h, 278
- gldi_style_colors_paint_bg_color_with_alpha
 - cairo-dock-style-manager.h, 281
- gldi_style_colors_set_bg_color
 - cairo-dock-style-manager.h, 279
- gldi_style_colors_set_bg_color_full
 - cairo-dock-style-manager.h, 279
- gldi_style_colors_set_child_color
 - cairo-dock-style-manager.h, 281
- gldi_style_colors_set_line_color
 - cairo-dock-style-manager.h, 280
- gldi_style_colors_set_selected_bg_color
 - cairo-dock-style-manager.h, 280
- gldi_style_colors_set_separator_color
 - cairo-dock-style-manager.h, 280
- gldi_style_colors_set_text_color
 - cairo-dock-style-manager.h, 280
- gldi_subdock_new
 - cairo-dock-dock-factory.h, 163
- gldi_submenu_new
 - cairo-dock-menu.h, 231
- gldi_task_change_frequency
 - cairo-dock-task.h, 294
- gldi_task_change_frequency_and_relaunch
 - cairo-dock-task.h, 296
- gldi_task_discard
 - cairo-dock-task.h, 293
- gldi_task_downgrade_frequency
 - cairo-dock-task.h, 296
- gldi_task_free
 - cairo-dock-task.h, 293
- gldi_task_get_elapsed_time
 - cairo-dock-task.h, 292
- gldi_task_is_active
 - cairo-dock-task.h, 294
- gldi_task_is_running
 - cairo-dock-task.h, 294
- gldi_task_launch
 - cairo-dock-task.h, 292
- gldi_task_launch_delayed
 - cairo-dock-task.h, 292
- gldi_task_new
 - cairo-dock-task.h, 291
- gldi_task_new_full
 - cairo-dock-task.h, 292
- gldi_task_set_normal_frequency
 - cairo-dock-task.h, 296
- gldi_task_stop
 - cairo-dock-task.h, 293
- gldi_window_foreach_inhibitor
 - cairo-dock-class-manager.h, 104
- gldi_window_get_menu_address
 - cairo-dock-windows-manager.h, 307
- gldi_window_manager_can_move_to_desktop
 - cairo-dock-windows-manager.h, 308
- gldi_window_manager_can_track_workspaces
 - cairo-dock-windows-manager.h, 307
- gldi_window_manager_get_all
 - cairo-dock-windows-manager.h, 307
- gldi_window_manager_have_coordinates
 - cairo-dock-windows-manager.h, 307
- gldi_window_manager_is_position_relative_to_current_viewport
 - cairo-dock-windows-manager.h, 308
- gldi_window_set_thumbnail_area
 - cairo-dock-windows-manager.h, 306
- gldi_windows_find

- cairo-dock-windows-manager.h, 306
- gdi_windows_foreach
 - cairo-dock-windows-manager.h, 306
- gdi_windows_get_active
 - cairo-dock-windows-manager.h, 306
- gdi_windows_manager_register_backend
 - cairo-dock-windows-manager.h, 305
- GdiAppInfo, 50
- GdiContainerNotifications
 - cairo-dock-container.h, 110
- GdiLaunchFlags
 - cairo-dock-utils.h, 302
- GdiObjectNotifications
 - cairo-dock-object.h, 244
- GdiStyleNotifications
 - cairo-dock-style-manager.h, 279
- init_layer
 - _GdiContainerManagerBackend, 39
- initModule
 - _GdiModuleInterface, 44
- iNumScreen
 - _CairoDock, 27
- load_custom_widget
 - _GdiModuleInterface, 45
- move_resize_dock
 - _GdiContainerManagerBackend, 40
- NOTIFICATION_BUILD_CONTAINER_MENU
 - cairo-dock-container.h, 110
- NOTIFICATION_BUILD_ICON_MENU
 - cairo-dock-container.h, 110
- NOTIFICATION_CLICK_ICON
 - cairo-dock-container.h, 110
- NOTIFICATION_CONFIGURE_DESKLET
 - cairo-dock-desklet-manager.h, 139
- NOTIFICATION_DESKTOP_CHANGED
 - cairo-dock-desktop-manager.h, 144
- NOTIFICATION_DESKTOP_GEOMETRY_CHANGED
 - cairo-dock-desktop-manager.h, 144
- NOTIFICATION_DESKTOP_MONITOR_ADDED
 - cairo-dock-desktop-manager.h, 144
- NOTIFICATION_DESKTOP_MONITOR_REMOVED
 - cairo-dock-desktop-manager.h, 144
- NOTIFICATION_DESKTOP_NAMES_CHANGED
 - cairo-dock-desktop-manager.h, 144
- NOTIFICATION_DESKTOP_VISIBILITY_CHANGED
 - cairo-dock-desktop-manager.h, 144
- NOTIFICATION_DESKTOP_WALLPAPER_CHANGED
 - cairo-dock-desktop-manager.h, 144
- NOTIFICATION_DESTROY
 - cairo-dock-object.h, 244
- NOTIFICATION_DOUBLE_CLICK_ICON
 - cairo-dock-container.h, 111
- NOTIFICATION_DROP_DATA
 - cairo-dock-container.h, 111
- NOTIFICATION_DROP_DATA_SELECTION
 - cairo-dock-container.h, 111
- NOTIFICATION_ENTER_DESKLET
 - cairo-dock-desklet-manager.h, 139
- NOTIFICATION_ENTER_DOCK
 - cairo-dock-dock-manager.h, 166
- NOTIFICATION_ENTER_ICON
 - cairo-dock-container.h, 111
- NOTIFICATION_ICON_MOVED
 - cairo-dock-dock-manager.h, 166
- NOTIFICATION_INSERT_ICON
 - cairo-dock-dock-manager.h, 166
- NOTIFICATION_KBD_STATE_CHANGED
 - cairo-dock-desktop-manager.h, 144
- NOTIFICATION_KEY_PRESSED
 - cairo-dock-container.h, 111
- NOTIFICATION_KEYMAP_CHANGED
 - cairo-dock-desktop-manager.h, 144
- NOTIFICATION_LEAVE_DESKLET
 - cairo-dock-desklet-manager.h, 139
- NOTIFICATION_LEAVE_DOCK
 - cairo-dock-dock-manager.h, 166
- NOTIFICATION_MENU_REQUEST
 - cairo-dock-desktop-manager.h, 144
- NOTIFICATION_MIDDLE_CLICK_ICON
 - cairo-dock-container.h, 111
- NOTIFICATION_MOUSE_MOVED
 - cairo-dock-container.h, 111
- NOTIFICATION_NEW
 - cairo-dock-object.h, 244
- NOTIFICATION_PRE_RENDER_ICON
 - cairo-dock-icon-manager.h, 215
- NOTIFICATION_REMOVE_ICON
 - cairo-dock-dock-manager.h, 166
- NOTIFICATION_RENDER
 - cairo-dock-container.h, 111
- NOTIFICATION_RENDER_ICON
 - cairo-dock-icon-manager.h, 215
- NOTIFICATION_REQUEST_ICON_ANIMATION
 - cairo-dock-icon-manager.h, 215
- NOTIFICATION_SCROLL_ICON
 - cairo-dock-container.h, 111
- NOTIFICATION_SHORTKEY_PRESSED
 - cairo-dock-desktop-manager.h, 144
- NOTIFICATION_SMOOTH_SCROLL_ICON
 - cairo-dock-container.h, 111
- NOTIFICATION_START_DRAG_DATA
 - cairo-dock-container.h, 111
- NOTIFICATION_STOP_ICON
 - cairo-dock-icon-manager.h, 215
- NOTIFICATION_STYLE_CHANGED
 - cairo-dock-style-manager.h, 279
- NOTIFICATION_UNFOLD_SUBDOCK
 - cairo-dock-icon-manager.h, 215
- NOTIFICATION_UPDATE
 - cairo-dock-container.h, 111
- NOTIFICATION_UPDATE_ICON
 - cairo-dock-icon-manager.h, 215
- NOTIFICATION_UPDATE_ICON_SLOW

cairo-dock-icon-manager.h, [215](#)
NOTIFICATION_UPDATE_SLOW
 cairo-dock-container.h, [111](#)

read_conf_file
 _GldiModuleInterface, [44](#)
reloadModule
 _GldiModuleInterface, [44](#)
reset_config
 _GldiModuleInterface, [44](#)
reset_data
 _GldiModuleInterface, [45](#)

set_keep_below
 _GldiContainerManagerBackend, [39](#)
spawn_app
 _GldiChildProcessManagerBackend, [37](#)
stopModule
 _GldiModuleInterface, [44](#)

update_polling_screen_edge
 _GldiContainerManagerBackend, [40](#)