

## L4Re – L4 Runtime Environment



# Contents

<b>1</b>	<b>Fiasco.OC &amp; L4 Runtime Environment (L4Re)</b>	<b>1</b>
1.1	Preface . . . . .	1
1.2	General System Structure . . . . .	1
1.3	The Fiasco.OC Microkernel . . . . .	3
1.3.1	Communication . . . . .	3
1.3.2	Kernel Objects . . . . .	3
1.4	L4 Runtime Environment (L4Re) . . . . .	4
1.5	Introduction to L4Re's concepts . . . . .	4
1.6	Memory management - Data Spaces and the Region Map . . . . .	4
1.6.1	User-level paging . . . . .	4
1.6.2	Data spaces . . . . .	4
1.6.3	Virtual Memory Handling . . . . .	5
1.6.4	Memory Allocation . . . . .	5
1.7	Capabilities and Naming . . . . .	5
1.8	Initial Environment and Application Bootstrapping . . . . .	6
1.8.1	Configuring an application before startup . . . . .	7
1.8.2	Connecting clients and servers . . . . .	7
1.9	Program Input and Output . . . . .	8
1.10	Initial Memory Allocator and Factory . . . . .	8
1.11	Application and Server Building Blocks . . . . .	8
1.11.1	Creating Additional Application Threads . . . . .	9
1.11.2	Providing a Service . . . . .	9
<b>2</b>	<b>Getting Started</b>	<b>11</b>
<b>3</b>	<b>L4Re Servers</b>	<b>13</b>
3.1	Sigma0, the Root Pager . . . . .	13
3.2	Moe, the Root Task . . . . .	13
3.3	Ned, the Default Init Process . . . . .	14

3.4	Io, the Platform and Device Resource Manager . . . . .	14
3.5	Mag, the GUI Multiplexer . . . . .	14
3.6	fb-drv, the Low-Level Graphics Driver . . . . .	14
3.7	Rtc, the Real-Time Clock Server . . . . .	14
3.8	Moe, the Root-Task . . . . .	14
3.8.1	Memory Allocator, Generic Factory . . . . .	15
3.8.2	Name-Space Provider . . . . .	15
3.8.3	Boot FS . . . . .	16
3.8.4	Log Subsystem . . . . .	16
3.8.5	Command-Line Options . . . . .	16
3.8.5.1	--debug=<debug flags> . . . . .	16
3.8.5.2	--init=<init process> . . . . .	16
3.8.5.3	--l4re-dbg=<debug flags> . . . . .	16
3.8.5.4	--ldr-flags=<loader flags> . . . . .	16
3.9	Ned, the Init Process . . . . .	16
3.9.1	Lua Bindings for L4Re . . . . .	17
3.9.1.1	Capabilities in Lua . . . . .	17
3.9.1.2	Access to L4Re::Env Capabilities . . . . .	17
3.9.1.3	Constants . . . . .	17
3.9.1.4	Application Startup Details . . . . .	18
3.10	Io, the Io Server . . . . .	19
<b>4</b>	<b>Pthread Support</b>	<b>23</b>
<b>5</b>	<b>Module Index</b>	<b>25</b>
5.1	Modules . . . . .	25
<b>6</b>	<b>Namespace Index</b>	<b>29</b>
6.1	Namespace List . . . . .	29
<b>7</b>	<b>Data Structure Index</b>	<b>31</b>
7.1	Class Hierarchy . . . . .	31
<b>8</b>	<b>Data Structure Index</b>	<b>37</b>
8.1	Data Structures . . . . .	37
<b>9</b>	<b>Module Documentation</b>	<b>43</b>
9.1	C++ Exceptions . . . . .	43
9.2	Small C++ Template Lib . . . . .	44

---

9.2.1	Function Documentation . . . . .	46
9.2.1.1	min . . . . .	46
9.2.1.2	max . . . . .	46
9.2.1.3	operator new . . . . .	46
9.3	L4Re C++ Interface . . . . .	46
9.3.1	Detailed Description . . . . .	48
9.4	L4Re Util C++ Interface . . . . .	49
9.4.1	Detailed Description . . . . .	50
9.5	Dataspace interface . . . . .	50
9.5.1	Detailed Description . . . . .	51
9.5.2	Function Documentation . . . . .	51
9.5.2.1	l4re_ds_clear . . . . .	51
9.5.2.2	l4re_ds_allocate . . . . .	51
9.5.2.3	l4re_ds_copy_in . . . . .	51
9.5.2.4	l4re_ds_size . . . . .	51
9.5.2.5	l4re_ds_flags . . . . .	51
9.5.2.6	l4re_ds_info . . . . .	52
9.5.2.7	l4re_ds_phys . . . . .	52
9.6	Debug interface . . . . .	52
9.6.1	Function Documentation . . . . .	53
9.6.1.1	l4re_debug_obj_debug . . . . .	53
9.7	Event interface . . . . .	53
9.7.1	Detailed Description . . . . .	53
9.7.2	Function Documentation . . . . .	53
9.7.2.1	l4re_event_get . . . . .	53
9.8	Log interface . . . . .	54
9.8.1	Detailed Description . . . . .	54
9.8.2	Function Documentation . . . . .	54
9.8.2.1	l4re_log_print . . . . .	54
9.8.2.2	l4re_log_printn . . . . .	55
9.8.2.3	l4re_log_print_srv . . . . .	56
9.8.2.4	l4re_log_printn_srv . . . . .	56
9.9	Memory allocator . . . . .	57
9.9.1	Detailed Description . . . . .	58
9.9.2	Enumeration Type Documentation . . . . .	58
9.9.2.1	l4re_ma_flags . . . . .	58

9.9.3	Function Documentation . . . . .	58
9.9.3.1	l4re_ma_alloc . . . . .	58
9.9.3.2	l4re_ma_free . . . . .	59
9.9.3.3	l4re_ma_alloc_srv . . . . .	60
9.9.3.4	l4re_ma_free_srv . . . . .	60
9.10	Namespace interface . . . . .	61
9.10.1	Detailed Description . . . . .	62
9.10.2	Enumeration Type Documentation . . . . .	62
9.10.2.1	l4re_ns_register_flags . . . . .	62
9.10.3	Function Documentation . . . . .	62
9.10.3.1	l4re_ns_query_to_srv . . . . .	62
9.10.3.2	l4re_ns_register_obj_srv . . . . .	62
9.11	Region map interface . . . . .	62
9.11.1	Detailed Description . . . . .	64
9.11.2	Enumeration Type Documentation . . . . .	64
9.11.2.1	l4re_rm_flags_t . . . . .	64
9.11.3	Function Documentation . . . . .	64
9.11.3.1	l4re_rm_reserve_area . . . . .	64
9.11.3.2	l4re_rm_free_area . . . . .	65
9.11.3.3	l4re_rm_attach . . . . .	65
9.11.3.4	l4re_rm_detach . . . . .	67
9.11.3.5	l4re_rm_detach_ds . . . . .	67
9.11.3.6	l4re_rm_detach_unmap . . . . .	68
9.11.3.7	l4re_rm_detach_ds_unmap . . . . .	69
9.11.3.8	l4re_rm_find . . . . .	70
9.11.3.9	l4re_rm_show_lists . . . . .	70
9.11.3.10	l4re_rm_reserve_area_srv . . . . .	70
9.11.3.11	l4re_rm_free_area_srv . . . . .	71
9.11.3.12	l4re_rm_attach_srv . . . . .	71
9.11.3.13	l4re_rm_detach_srv . . . . .	72
9.11.3.14	l4re_rm_find_srv . . . . .	72
9.12	Capability allocator . . . . .	73
9.12.1	Detailed Description . . . . .	73
9.12.2	Function Documentation . . . . .	74
9.12.2.1	l4re_util_cap_last . . . . .	74
9.13	Kumem allocator utility . . . . .	74

---

9.13.1	Detailed Description	74
9.13.2	Function Documentation	74
9.13.2.1	<code>l4re_util_kumem_alloc</code>	74
9.14	Video API	75
9.14.1	Typedef Documentation	77
9.14.1.1	<code>l4re_video_view_t</code>	77
9.14.2	Enumeration Type Documentation	77
9.14.2.1	<code>l4re_video_goops_info_flags_t</code>	77
9.14.2.2	<code>l4re_video_view_info_flags_t</code>	78
9.14.3	Function Documentation	78
9.14.3.1	<code>l4re_video_goops_info</code>	78
9.14.3.2	<code>l4re_video_goops_refresh</code>	78
9.14.3.3	<code>l4re_video_goops_create_buffer</code>	79
9.14.3.4	<code>l4re_video_goops_delete_buffer</code>	79
9.14.3.5	<code>l4re_video_goops_get_static_buffer</code>	79
9.14.3.6	<code>l4re_video_goops_create_view</code>	79
9.14.3.7	<code>l4re_video_goops_delete_view</code>	80
9.14.3.8	<code>l4re_video_goops_get_view</code>	80
9.14.3.9	<code>l4re_video_view_refresh</code>	80
9.14.3.10	<code>l4re_video_view_get_info</code>	80
9.14.3.11	<code>l4re_video_view_set_info</code>	81
9.14.3.12	<code>l4re_video_view_set_viewport</code>	81
9.14.3.13	<code>l4re_video_view_stack</code>	81
9.15	Console API	81
9.15.1	Detailed Description	82
9.16	Data-Space API	82
9.16.1	Detailed Description	83
9.17	Debugging API	83
9.17.1	Detailed Description	83
9.18	L4Re ELF Auxiliary Information	83
9.18.1	Detailed Description	85
9.18.2	Define Documentation	85
9.18.2.1	<code>L4RE_ELF_AUX_ELEM</code>	85
9.18.2.2	<code>L4RE_ELF_AUX_ELEM_T</code>	85
9.18.3	Enumeration Type Documentation	86
9.18.3.1	"@52	86

9.19	Initial Environment	86
9.19.1	Detailed Description	87
9.19.2	Typedef Documentation	88
9.19.2.1	<code>l4re_env_t</code>	88
9.19.3	Function Documentation	88
9.19.3.1	<code>l4re_env</code>	88
9.19.3.2	<code>l4re_kip</code>	88
9.19.3.3	<code>l4re_get_env_cap</code>	89
9.19.3.4	<code>l4re_get_env_cap_e</code>	89
9.19.3.5	<code>l4re_get_env_cap_l</code>	90
9.20	Event API	91
9.20.1	Detailed Description	91
9.21	Auxiliary data	92
9.22	Logging interface	92
9.22.1	Detailed Description	93
9.23	Memory allocator API	93
9.23.1	Detailed Description	93
9.24	Name-space API	94
9.24.1	Detailed Description	94
9.25	Parent API	94
9.25.1	Detailed Description	95
9.26	L4Re Protocol identifiers	95
9.26.1	Detailed Description	96
9.26.2	Enumeration Type Documentation	96
9.26.2.1	Protocols	96
9.27	Region map API	96
9.27.1	Detailed Description	97
9.28	L4Re Capability API	97
9.28.1	Variable Documentation	98
9.28.1.1	<code>cap_alloc</code>	98
9.29	Kumem utilities	99
9.29.1	Function Documentation	99
9.29.1.1	<code>kumem_alloc</code>	99
9.30	Goos video API	100
9.31	L4Re C Interface	100
9.31.1	Detailed Description	102

9.32 L4Re Util C Interface . . . . .	102
9.32.1 Detailed Description . . . . .	103
9.32.2 Function Documentation . . . . .	103
9.32.2.1 l4re_util_cap_release . . . . .	103
9.33 Base API . . . . .	103
9.33.1 Detailed Description . . . . .	107
9.34 IPC-Gate API . . . . .	107
9.34.1 Detailed Description . . . . .	108
9.34.2 Enumeration Type Documentation . . . . .	108
9.34.2.1 L4_ipc_gate_ops . . . . .	108
9.34.3 Function Documentation . . . . .	108
9.34.3.1 l4_ipc_gate_bind_thread . . . . .	108
9.34.3.2 l4_ipc_gate_get_infos . . . . .	109
9.35 Basic Macros . . . . .	110
9.35.1 Detailed Description . . . . .	111
9.35.2 Define Documentation . . . . .	111
9.35.2.1 L4_DECLARE_CONSTRUCTOR . . . . .	111
9.35.2.2 L4_NOTHROW . . . . .	112
9.35.2.3 L4_EXPORT . . . . .	112
9.35.2.4 L4_HIDDEN . . . . .	112
9.36 Fiasco extensions . . . . .	113
9.36.1 Detailed Description . . . . .	116
9.36.2 Function Documentation . . . . .	116
9.36.2.1 fiasco_tbuf_get_status . . . . .	116
9.36.2.2 fiasco_tbuf_get_status_phys . . . . .	116
9.36.2.3 fiasco_tbuf_log . . . . .	116
9.36.2.4 fiasco_tbuf_log_3val . . . . .	117
9.36.2.5 fiasco_tbuf_log_binary . . . . .	118
9.36.2.6 fiasco_tbuf_clear . . . . .	118
9.36.2.7 fiasco_tbuf_dump . . . . .	119
9.36.2.8 fiasco_watchdog_takeover . . . . .	119
9.36.2.9 fiasco_watchdog_touch . . . . .	119
9.36.2.10 fiasco_ldt_set . . . . .	119
9.36.2.11 fiasco_gdt_set . . . . .	120
9.36.2.12 fiasco_gdt_get_entry_offset . . . . .	121
9.37 Fiasco real time scheduling extensions . . . . .	121

9.37.1	Detailed Description . . . . .	122
9.37.2	Function Documentation . . . . .	122
9.37.2.1	slice . . . . .	122
9.37.2.2	slice . . . . .	123
9.37.2.3	l4_rt_begin_strictly_periodic . . . . .	123
9.37.2.4	l4_rt_begin_minimal_periodic . . . . .	124
9.37.2.5	l4_rt_end_periodic . . . . .	125
9.37.2.6	l4_rt_remove . . . . .	125
9.37.2.7	l4_rt_set_period . . . . .	126
9.37.2.8	l4_rt_next_reservation . . . . .	127
9.37.2.9	l4_rt_next_period . . . . .	127
9.37.2.10	l4_preemption_id . . . . .	128
9.37.2.11	l4_next_period_id . . . . .	128
9.37.2.12	l4_rt_generic . . . . .	129
9.38	Flex pages . . . . .	130
9.38.1	Detailed Description . . . . .	132
9.38.2	Enumeration Type Documentation . . . . .	132
9.38.2.1	l4_fpage_consts . . . . .	132
9.38.2.2	"@60 . . . . .	133
9.38.2.3	L4_fpage_rights . . . . .	133
9.38.2.4	L4_cap_fpage_rights . . . . .	133
9.38.2.5	l4_fpage_cacheability_opt_t . . . . .	134
9.38.2.6	"@61 . . . . .	134
9.38.3	Function Documentation . . . . .	134
9.38.3.1	l4_fpage . . . . .	134
9.38.3.2	l4_fpage_all . . . . .	134
9.38.3.3	l4_fpage_invalid . . . . .	135
9.38.3.4	l4_iofpage . . . . .	135
9.38.3.5	l4_obj_fpage . . . . .	135
9.38.3.6	l4_is_fpage_writable . . . . .	136
9.38.3.7	l4_fpage_rights . . . . .	136
9.38.3.8	l4_fpage_type . . . . .	137
9.38.3.9	l4_fpage_size . . . . .	137
9.38.3.10	l4_fpage_page . . . . .	137
9.38.3.11	l4_fpage_set_rights . . . . .	138
9.38.3.12	l4_fpage_contains . . . . .	138

9.38.3.13 l4_fpage_max_order . . . . .	139
9.39 Message Items . . . . .	140
9.39.1 Detailed Description . . . . .	140
9.39.2 Enumeration Type Documentation . . . . .	141
9.39.2.1 l4_msg_item_consts_t . . . . .	141
9.39.3 Function Documentation . . . . .	141
9.39.3.1 l4_map_control . . . . .	141
9.39.3.2 l4_map_obj_control . . . . .	142
9.40 Timeouts . . . . .	142
9.40.1 Detailed Description . . . . .	144
9.40.2 Define Documentation . . . . .	145
9.40.2.1 L4_IPC_TIMEOUT_0 . . . . .	145
9.40.3 Typedef Documentation . . . . .	145
9.40.3.1 l4_timeout_s . . . . .	145
9.40.3.2 l4_timeout_t . . . . .	145
9.40.4 Enumeration Type Documentation . . . . .	145
9.40.4.1 l4_timeout_abs_validity . . . . .	145
9.40.5 Function Documentation . . . . .	145
9.40.5.1 l4_timeout_rel . . . . .	145
9.40.5.2 l4_ipc_timeout . . . . .	146
9.40.5.3 l4_timeout . . . . .	146
9.40.5.4 l4_snd_timeout . . . . .	146
9.40.5.5 l4_rcv_timeout . . . . .	146
9.40.5.6 l4_timeout_rel_get . . . . .	147
9.40.5.7 l4_timeout_is_absolute . . . . .	147
9.40.5.8 l4_timeout_get . . . . .	148
9.40.5.9 l4_timeout_abs . . . . .	148
9.41 VM API for SVM . . . . .	149
9.41.1 Detailed Description . . . . .	150
9.42 VM API for VMX . . . . .	150
9.42.1 Detailed Description . . . . .	151
9.42.2 Function Documentation . . . . .	151
9.42.2.1 l4_vm_vmx_field_len . . . . .	151
9.42.2.2 l4_vm_vmx_field_ptr . . . . .	151
9.43 Cache Consistency . . . . .	152
9.43.1 Detailed Description . . . . .	153

---

9.43.2 Function Documentation . . . . .	153
9.43.2.1 l4_cache_clean_data . . . . .	153
9.43.2.2 l4_cache_flush_data . . . . .	153
9.43.2.3 l4_cache_inv_data . . . . .	153
9.43.2.4 l4_cache_coherent . . . . .	154
9.43.2.5 l4_cache_dma_coherent . . . . .	154
9.44 Memory related . . . . .	154
9.44.1 Detailed Description . . . . .	156
9.44.2 Define Documentation . . . . .	156
9.44.2.1 L4_PAGEMASK . . . . .	156
9.44.2.2 L4_LOG2_PAGESIZE . . . . .	156
9.44.2.3 L4_SUPERPAGESIZE . . . . .	156
9.44.2.4 L4_SUPERPAGEMASK . . . . .	156
9.44.2.5 L4_LOG2_SUPERPAGESIZE . . . . .	156
9.44.3 Enumeration Type Documentation . . . . .	157
9.44.3.1 l4_addr_consts_t . . . . .	157
9.44.4 Function Documentation . . . . .	157
9.44.4.1 l4_trunc_page . . . . .	157
9.44.4.2 l4_trunc_size . . . . .	157
9.44.4.3 l4_round_page . . . . .	158
9.44.4.4 l4_round_size . . . . .	158
9.45 Kernel Debugger . . . . .	159
9.45.1 Detailed Description . . . . .	161
9.45.2 Define Documentation . . . . .	161
9.45.2.1 enter_kdebug . . . . .	161
9.45.2.2 asm_enter_kdebug . . . . .	162
9.45.2.3 kd_display . . . . .	162
9.45.2.4 ko . . . . .	162
9.45.2.5 enter_kdebug . . . . .	163
9.45.2.6 asm_enter_kdebug . . . . .	163
9.45.2.7 kd_display . . . . .	163
9.45.2.8 ko . . . . .	164
9.45.3 Function Documentation . . . . .	164
9.45.3.1 l4_debugger_set_object_name . . . . .	164
9.45.3.2 l4_debugger_global_id . . . . .	164
9.45.3.3 l4_debugger_kobj_to_id . . . . .	165

9.45.3.4	outchar . . . . .	165
9.45.3.5	outstring . . . . .	166
9.45.3.6	outnstring . . . . .	166
9.45.3.7	outhex32 . . . . .	166
9.45.3.8	outhex20 . . . . .	166
9.45.3.9	outhex16 . . . . .	166
9.45.3.10	outhex12 . . . . .	167
9.45.3.11	outhex8 . . . . .	167
9.45.3.12	outdec . . . . .	167
9.45.3.13	l4kd_inchar . . . . .	167
9.46	Error codes . . . . .	167
9.46.1	Detailed Description . . . . .	168
9.46.2	Enumeration Type Documentation . . . . .	168
9.46.2.1	l4_error_code_t . . . . .	168
9.47	Factory . . . . .	169
9.47.1	Detailed Description . . . . .	170
9.47.2	Function Documentation . . . . .	170
9.47.2.1	l4_factory_create_task . . . . .	170
9.47.2.2	l4_factory_create_thread . . . . .	171
9.47.2.3	l4_factory_create_factory . . . . .	171
9.47.2.4	l4_factory_create_gate . . . . .	172
9.47.2.5	l4_factory_create_semaphore . . . . .	173
9.47.2.6	l4_factory_create_irq . . . . .	173
9.47.2.7	l4_factory_create_vm . . . . .	174
9.48	Virtual Machines . . . . .	174
9.48.1	Detailed Description . . . . .	175
9.49	Interrupt controller . . . . .	175
9.49.1	Detailed Description . . . . .	177
9.49.2	Typedef Documentation . . . . .	177
9.49.2.1	l4_icu_info_t . . . . .	177
9.49.3	Enumeration Type Documentation . . . . .	177
9.49.3.1	L4_icu_flags . . . . .	177
9.49.4	Function Documentation . . . . .	177
9.49.4.1	l4_icu_bind . . . . .	177
9.49.4.2	l4_icu_unbind . . . . .	178
9.49.4.3	l4_icu_set_mode . . . . .	179

9.49.4.4	<code>l4_icu_info</code>	179
9.49.4.5	<code>l4_icu_msi_info</code>	180
9.49.4.6	<code>l4_icu_unmask</code>	181
9.49.4.7	<code>l4_icu_mask</code>	181
9.50	<code>Object Invocation</code>	182
9.50.1	<code>Detailed Description</code>	184
9.50.2	<code>Enumeration Type Documentation</code>	184
9.50.2.1	<code>l4_syscall_flags_t</code>	184
9.50.3	<code>Function Documentation</code>	185
9.50.3.1	<code>l4_ipc_send</code>	185
9.50.3.2	<code>l4_ipc_wait</code>	186
9.50.3.3	<code>l4_ipc_receive</code>	187
9.50.3.4	<code>l4_ipc_call</code>	188
9.50.3.5	<code>l4_ipc_reply_and_wait</code>	189
9.50.3.6	<code>l4_ipc_send_and_wait</code>	190
9.50.3.7	<code>l4_ipc</code>	191
9.50.3.8	<code>l4_ipc_sleep</code>	191
9.50.3.9	<code>l4_sndfpage_add</code>	192
9.51	<code>Error Handling</code>	193
9.51.1	<code>Detailed Description</code>	194
9.51.2	<code>Enumeration Type Documentation</code>	194
9.51.2.1	<code>l4_ipc_tcr_error_t</code>	194
9.51.3	<code>Function Documentation</code>	195
9.51.3.1	<code>l4_ipc_error</code>	195
9.51.3.2	<code>l4_error</code>	195
9.51.3.3	<code>l4_ipc_is_snd_error</code>	197
9.51.3.4	<code>l4_ipc_is_rcv_error</code>	197
9.51.3.5	<code>l4_ipc_error_code</code>	197
9.52	<code>Realtime API</code>	198
9.53	<code>IRQs</code>	198
9.53.1	<code>Detailed Description</code>	199
9.53.2	<code>Enumeration Type Documentation</code>	199
9.53.2.1	<code>L4_irq_flow_type</code>	199
9.53.3	<code>Function Documentation</code>	200
9.53.3.1	<code>l4_irq_attach</code>	200
9.53.3.2	<code>l4_irq_chain</code>	200

9.53.3.3	<code>l4_irq_detach</code>	201
9.53.3.4	<code>l4_irq_trigger</code>	202
9.53.3.5	<code>l4_irq_receive</code>	202
9.53.3.6	<code>l4_irq_wait</code>	203
9.53.3.7	<code>l4_irq_unmask</code>	204
9.54	Kernel Objects	204
9.54.1	Detailed Description	206
9.54.2	Define Documentation	206
9.54.2.1	<code>L4_DISABLE_COPY</code>	206
9.54.2.2	<code>L4_KOBJECT_DISABLE_COPY</code>	207
9.54.2.3	<code>L4_KOBJECT</code>	207
9.55	Kernel Interface Page	208
9.55.1	Detailed Description	209
9.55.2	Function Documentation	209
9.55.2.1	<code>l4_kip_version</code>	209
9.55.2.2	<code>l4_kip_version_string</code>	209
9.55.2.3	<code>l4_kernel_info_version_offset</code>	210
9.56	Memory descriptors (C version)	211
9.56.1	Detailed Description	212
9.56.2	Typedef Documentation	212
9.56.2.1	<code>l4_kernel_info_mem_desc_t</code>	212
9.56.3	Enumeration Type Documentation	212
9.56.3.1	<code>l4_mem_type_t</code>	212
9.56.4	Function Documentation	213
9.56.4.1	<code>l4_kernel_info_get_num_mem_descs</code>	213
9.56.4.2	<code>l4_kernel_info_set_mem_desc</code>	213
9.56.4.3	<code>l4_kernel_info_get_mem_desc_start</code>	213
9.56.4.4	<code>l4_kernel_info_get_mem_desc_end</code>	214
9.56.4.5	<code>l4_kernel_info_get_mem_desc_type</code>	214
9.56.4.6	<code>l4_kernel_info_get_mem_desc_subtype</code>	214
9.56.4.7	<code>l4_kernel_info_get_mem_desc_is_virtual</code>	214
9.57	Scheduler	214
9.57.1	Detailed Description	216
9.57.2	Enumeration Type Documentation	216
9.57.2.1	<code>L4_scheduler_ops</code>	216
9.57.3	Function Documentation	216

9.57.3.1 <code>l4_sched_cpu_set</code>	216
9.57.3.2 <code>l4_scheduler_info</code>	217
9.57.3.3 <code>l4_scheduler_run_thread</code>	218
9.57.3.4 <code>l4_scheduler_idle_time</code>	218
9.57.3.5 <code>l4_scheduler_is_online</code>	219
9.58 Task	219
9.58.1 Detailed Description	221
9.58.2 Enumeration Type Documentation	221
9.58.2.1 <code>l4_unmap_flags_t</code>	221
9.58.3 Function Documentation	221
9.58.3.1 <code>l4_task_map</code>	221
9.58.3.2 <code>l4_task_unmap</code>	222
9.58.3.3 <code>l4_task_unmap_batch</code>	223
9.58.3.4 <code>l4_task_cap_valid</code>	224
9.58.3.5 <code>l4_task_cap_has_child</code>	224
9.58.3.6 <code>l4_task_cap_equal</code>	225
9.58.3.7 <code>l4_task_add_ku_mem</code>	226
9.59 Thread	226
9.59.1 Detailed Description	228
9.59.2 Enumeration Type Documentation	229
9.59.2.1 <code>L4_thread_ops</code>	229
9.59.2.2 <code>L4_thread_control_flags</code>	230
9.59.2.3 <code>L4_thread_control_mr_indices</code>	230
9.59.2.4 <code>L4_thread_ex_regs_flags</code>	230
9.59.3 Function Documentation	230
9.59.3.1 <code>l4_thread_ex_regs</code>	230
9.59.3.2 <code>l4_thread_ex_regs_ret</code>	231
9.59.3.3 <code>l4_thread_yield</code>	232
9.59.3.4 <code>l4_thread_switch</code>	232
9.59.3.5 <code>l4_thread_stats_time</code>	233
9.59.3.6 <code>l4_thread_vcpu_resume_start</code>	233
9.59.3.7 <code>l4_thread_vcpu_resume_commit</code>	234
9.59.3.8 <code>l4_thread_vcpu_control</code>	235
9.59.3.9 <code>l4_thread_vcpu_control_ext</code>	235
9.59.3.10 <code>l4_thread_register_del_irq</code>	236
9.59.3.11 <code>l4_thread_modify_sender_start</code>	237

9.59.3.12 <code>l4_thread_modify_sender_add</code>	237
9.59.3.13 <code>l4_thread_modify_sender_commit</code>	238
9.60 Thread control	239
9.60.1 Detailed Description	240
9.60.2 Function Documentation	240
9.60.2.1 <code>l4_thread_control_start</code>	240
9.60.2.2 <code>l4_thread_control_pager</code>	241
9.60.2.3 <code>l4_thread_control_exc_handler</code>	241
9.60.2.4 <code>l4_thread_control_bind</code>	242
9.60.2.5 <code>l4_thread_control_alien</code>	243
9.60.2.6 <code>l4_thread_control_ux_host_syscall</code>	243
9.60.2.7 <code>l4_thread_control_commit</code>	244
9.61 Message Tag	245
9.61.1 Detailed Description	247
9.61.2 Typedef Documentation	247
9.61.2.1 <code>l4_mshtag_t</code>	247
9.61.3 Enumeration Type Documentation	247
9.61.3.1 <code>l4_mshtag_protocol</code>	247
9.61.3.2 <code>l4_mshtag_flags</code>	248
9.61.4 Function Documentation	248
9.61.4.1 <code>l4_mshtag</code>	248
9.61.4.2 <code>l4_mshtag_label</code>	249
9.61.4.3 <code>l4_mshtag_words</code>	250
9.61.4.4 <code>l4_mshtag_items</code>	250
9.61.4.5 <code>l4_mshtag_flags</code>	251
9.61.4.6 <code>l4_mshtag_has_error</code>	251
9.61.4.7 <code>l4_mshtag_is_page_fault</code>	252
9.61.4.8 <code>l4_mshtag_is_preemption</code>	252
9.61.4.9 <code>l4_mshtag_is_sys_exception</code>	253
9.61.4.10 <code>l4_mshtag_is_exception</code>	253
9.61.4.11 <code>l4_mshtag_is_sigma0</code>	254
9.61.4.12 <code>l4_mshtag_is_io_page_fault</code>	254
9.62 Capabilities	255
9.62.1 Detailed Description	256
9.62.2 Typedef Documentation	256
9.62.2.1 <code>l4_cap_idx_t</code>	256

9.62.3 Enumeration Type Documentation . . . . .	257
9.62.3.1 l4_cap_consts_t . . . . .	257
9.62.3.2 l4_default_caps_t . . . . .	257
9.62.4 Function Documentation . . . . .	258
9.62.4.1 cap_cast . . . . .	258
9.62.4.2 cap_reinterpret_cast . . . . .	258
9.62.4.3 cap_dynamic_cast . . . . .	259
9.62.4.4 l4_is_invalid_cap . . . . .	260
9.62.4.5 l4_is_valid_cap . . . . .	260
9.62.4.6 l4_capability_equal . . . . .	260
9.63 Virtual Registers (UTCBs) . . . . .	261
9.63.1 Detailed Description . . . . .	262
9.63.2 Typedef Documentation . . . . .	263
9.63.2.1 l4_utcb_t . . . . .	263
9.63.3 Function Documentation . . . . .	263
9.63.3.1 l4_utcb_mr . . . . .	263
9.63.3.2 l4_utcb_br . . . . .	264
9.63.3.3 l4_utcb_tcr . . . . .	264
9.64 Message Registers (MRs) . . . . .	265
9.65 Buffer Registers (BRs) . . . . .	265
9.65.1 Enumeration Type Documentation . . . . .	266
9.65.1.1 l4_buffer_desc_consts_t . . . . .	266
9.66 Thread Control Registers (TCRs) . . . . .	266
9.67 Exception registers . . . . .	267
9.67.1 Detailed Description . . . . .	268
9.67.2 Function Documentation . . . . .	268
9.67.2.1 l4_utcb_exc . . . . .	268
9.67.2.2 l4_utcb_exc_pc . . . . .	268
9.67.2.3 l4_utcb_exc_pc_set . . . . .	269
9.67.2.4 l4_utcb_exc_is_pf . . . . .	269
9.68 Virtual Console . . . . .	269
9.68.1 Detailed Description . . . . .	270
9.68.2 Enumeration Type Documentation . . . . .	271
9.68.2.1 L4_vcon_write_consts . . . . .	271
9.68.2.2 L4_vcon_i_flags . . . . .	271
9.68.2.3 L4_vcon_o_flags . . . . .	271

9.68.2.4	L4_vcon_l_flags . . . . .	271
9.68.2.5	L4_vcon_ops . . . . .	272
9.68.3	Function Documentation . . . . .	272
9.68.3.1	l4_vcon_send . . . . .	272
9.68.3.2	l4_vcon_write . . . . .	272
9.68.3.3	l4_vcon_read . . . . .	273
9.68.3.4	l4_vcon_set_attr . . . . .	274
9.68.3.5	l4_vcon_get_attr . . . . .	274
9.69	vCPU API . . . . .	275
9.69.1	Detailed Description . . . . .	276
9.69.2	Enumeration Type Documentation . . . . .	276
9.69.2.1	L4_vcpu_state_flags . . . . .	276
9.69.2.2	L4_vcpu_sticky_flags . . . . .	277
9.69.2.3	L4_vcpu_state_offset . . . . .	277
9.70	Fiasco-UX Virtual devices . . . . .	277
9.70.1	Detailed Description . . . . .	278
9.70.2	Enumeration Type Documentation . . . . .	278
9.70.2.1	l4_vhw_entry_type . . . . .	278
9.71	Memory operations. . . . .	278
9.71.1	Detailed Description . . . . .	279
9.71.2	Enumeration Type Documentation . . . . .	279
9.71.2.1	L4_mem_op_widths . . . . .	279
9.71.3	Function Documentation . . . . .	279
9.71.3.1	l4_mem_read . . . . .	279
9.71.3.2	l4_mem_write . . . . .	279
9.72	ARM Virtual Registers (UTCB) . . . . .	280
9.73	VM API for TZ . . . . .	280
9.73.1	Detailed Description . . . . .	281
9.73.2	Function Documentation . . . . .	281
9.73.2.1	l4_vm_run . . . . .	281
9.74	amd64 Virtual Registers (UTCB) . . . . .	282
9.75	x86 Virtual Registers (UTCB) . . . . .	282
9.75.1	Enumeration Type Documentation . . . . .	283
9.75.1.1	L4_utcb_consts_x86 . . . . .	283
9.76	CPU related functions . . . . .	283
9.76.1	Function Documentation . . . . .	284

9.76.1.1 l4util_cpu_has_cpuid . . . . .	284
9.76.1.2 l4util_cpu_capabilities . . . . .	284
9.76.1.3 l4util_cpu_capabilities_nocheck . . . . .	285
9.77 Functions to manipulate the local IDT . . . . .	286
9.78 Timestamp Counter . . . . .	286
9.78.1 Function Documentation . . . . .	288
9.78.1.1 l4_rdtsc . . . . .	288
9.78.1.2 l4_rdtsc_32 . . . . .	288
9.78.1.3 l4_rdpmc . . . . .	289
9.78.1.4 l4_rdpmc_32 . . . . .	289
9.78.1.5 l4_tsc_to_ns . . . . .	289
9.78.1.6 l4_tsc_to_us . . . . .	289
9.78.1.7 l4_tsc_to_s_and_ns . . . . .	290
9.78.1.8 l4_ns_to_tsc . . . . .	290
9.78.1.9 l4_busy_wait_ns . . . . .	290
9.78.1.10 l4_busy_wait_us . . . . .	291
9.78.1.11 l4_calibrate_tsc . . . . .	292
9.78.1.12 l4_tsc_init . . . . .	292
9.78.1.13 l4_get_hz . . . . .	293
9.79 Atomic Instructions . . . . .	294
9.79.1 Function Documentation . . . . .	296
9.79.1.1 l4util_cmpxchg64 . . . . .	296
9.79.1.2 l4util_cmpxchg32 . . . . .	296
9.79.1.3 l4util_cmpxchg16 . . . . .	297
9.79.1.4 l4util_cmpxchg8 . . . . .	297
9.79.1.5 l4util_cmpxchg . . . . .	297
9.79.1.6 l4util_xchg32 . . . . .	298
9.79.1.7 l4util_xchg16 . . . . .	298
9.79.1.8 l4util_xchg8 . . . . .	299
9.79.1.9 l4util_xchg . . . . .	299
9.79.1.10 l4util_add8 . . . . .	299
9.79.1.11 l4util_add8_res . . . . .	299
9.79.1.12 l4util_inc8 . . . . .	300
9.79.1.13 l4util_inc8_res . . . . .	300
9.79.1.14 l4util_atomic_add . . . . .	300
9.79.1.15 l4util_atomic_inc . . . . .	300

9.80 Internal functions . . . . .	300
9.81 Bit Manipulation . . . . .	301
9.81.1 Function Documentation . . . . .	302
9.81.1.1 l4util_set_bit . . . . .	302
9.81.1.2 l4util_clear_bit . . . . .	303
9.81.1.3 l4util_complement_bit . . . . .	303
9.81.1.4 l4util_test_bit . . . . .	303
9.81.1.5 l4util_bts . . . . .	304
9.81.1.6 l4util_btr . . . . .	304
9.81.1.7 l4util_btc . . . . .	305
9.81.1.8 l4util_bsr . . . . .	305
9.81.1.9 l4util_bsf . . . . .	306
9.81.1.10 l4util_find_first_set_bit . . . . .	306
9.81.1.11 l4util_find_first_zero_bit . . . . .	307
9.81.1.12 l4util_next_power2 . . . . .	307
9.82 ELF binary format . . . . .	307
9.82.1 Detailed Description . . . . .	324
9.82.2 Define Documentation . . . . .	324
9.82.2.1 EI_CLASS . . . . .	324
9.82.2.2 EL_CLASS . . . . .	324
9.82.2.3 ELFCLASSNONE . . . . .	324
9.82.2.4 ELFCLASSNONE . . . . .	324
9.82.2.5 EI_DATA . . . . .	324
9.82.2.6 EL_DATA . . . . .	325
9.82.2.7 ELFDATANONE . . . . .	325
9.82.2.8 ELFDATANONE . . . . .	325
9.82.2.9 ELFDATA2LSB . . . . .	325
9.82.2.10 ELFDATA2LSB . . . . .	325
9.82.2.11 ELFDATA2MSB . . . . .	325
9.82.2.12 ELFDATA2MSB . . . . .	325
9.82.2.13 EI_VERSION . . . . .	326
9.82.2.14 EL_VERSION . . . . .	326
9.82.2.15 EI_OSABI . . . . .	326
9.82.2.16 EL_OSABI . . . . .	326
9.82.2.17 ELFOSABI_SYSV . . . . .	326
9.82.2.18 ELFOSABI_SYSV . . . . .	326

9.82.2.19 ELFOSABI_HPUX . . . . .	326
9.82.2.20 ELFOSABI_HPUX . . . . .	327
9.82.2.21 ELFOSABI_NETBSD . . . . .	327
9.82.2.22 ELFOSABI_LINUX . . . . .	327
9.82.2.23 ELFOSABI_SOLARIS . . . . .	327
9.82.2.24 ELFOSABI_AIX . . . . .	327
9.82.2.25 ELFOSABI_IRIX . . . . .	327
9.82.2.26 ELFOSABI_FREEBSD . . . . .	327
9.82.2.27 ELFOSABI_TRU64 . . . . .	327
9.82.2.28 ELFOSABI_MODESTO . . . . .	328
9.82.2.29 ELFOSABI_OPENBSD . . . . .	328
9.82.2.30 EI_PAD . . . . .	328
9.82.2.31 EI_PAD . . . . .	328
9.82.2.32 EM_ARC . . . . .	328
9.82.2.33 SHT_NUM . . . . .	328
9.82.2.34 SHF_GROUP . . . . .	328
9.82.2.35 SHF_TLS . . . . .	328
9.82.2.36 SHF_MASKOS . . . . .	329
9.82.2.37 PT_LOOS . . . . .	329
9.82.2.38 PT_HIOS . . . . .	329
9.82.2.39 PT_LOPROC . . . . .	329
9.82.2.40 PT_HIPROC . . . . .	329
9.82.2.41 PT_GNU_EH_FRAME . . . . .	329
9.82.2.42 PT_GNU_STACK . . . . .	329
9.82.2.43 PT_GNU_RELRO . . . . .	329
9.82.2.44 PT_L4_STACK . . . . .	329
9.82.2.45 PT_L4_KIP . . . . .	330
9.82.2.46 PT_L4_AUX . . . . .	330
9.82.2.47 NT_VERSION . . . . .	330
9.82.2.48 DT_NULL . . . . .	330
9.82.2.49 DT_LOPROC . . . . .	330
9.82.2.50 DT_HIPROC . . . . .	330
9.82.2.51 DF_1_NOW . . . . .	330
9.82.2.52 DF_1_GLOBAL . . . . .	330
9.82.2.53 DF_1_GROUP . . . . .	331
9.82.2.54 DF_1_NODELETE . . . . .	331

9.82.2.55 DF_1_LOADFLTR . . . . .	331
9.82.2.56 DF_1_NOOPEN . . . . .	331
9.82.2.57 DF_1_ORIGIN . . . . .	331
9.82.2.58 DF_1_DIRECT . . . . .	331
9.82.2.59 DF_1_INTERPOSE . . . . .	331
9.82.2.60 DF_1_NODEFLIB . . . . .	331
9.82.2.61 DF_1_NODUMP . . . . .	331
9.82.2.62 DF_1_CONFALT . . . . .	332
9.82.2.63 DF_1_ENDFILTEE . . . . .	332
9.82.2.64 DF_1_DISPRELDNE . . . . .	332
9.82.2.65 DF_1_DISPRELPPND . . . . .	332
9.82.2.66 DF_P1_LAZYLOAD . . . . .	332
9.82.2.67 DF_P1_GROUPPERM . . . . .	332
9.83 Kernel Interface Page API . . . . .	332
9.83.1 Define Documentation . . . . .	333
9.83.1.1 l4util_kip_for_each_feature . . . . .	333
9.83.2 Function Documentation . . . . .	333
9.83.2.1 l4util_kip_kernel_is_ux . . . . .	333
9.83.2.2 l4util_kip_kernel_has_feature . . . . .	334
9.83.2.3 l4util_kip_kernel_abi_version . . . . .	334
9.83.2.4 l4util_memdesc_vm_high . . . . .	334
9.84 Comfortable Command Line Parsing . . . . .	334
9.84.1 Function Documentation . . . . .	335
9.84.1.1 parse_cmdline . . . . .	335
9.85 Priority related functions . . . . .	336
9.86 Random number support . . . . .	337
9.86.1 Function Documentation . . . . .	337
9.86.1.1 l4util_rand . . . . .	337
9.86.1.2 l4util_srand . . . . .	337
9.87 Machine Restarting Function . . . . .	338
9.88 Low-Level Thread Functions . . . . .	338
9.89 Utility Functions . . . . .	340
9.89.1 Function Documentation . . . . .	342
9.89.1.1 l4_sleep_forever . . . . .	342
9.89.1.2 l4util_splitlog2_hdl . . . . .	342
9.89.1.3 l4util_splitlog2_size . . . . .	343

9.89.1.4	l4util_micros2l4to . . . . .	344
9.90	IA32 Port I/O API . . . . .	344
9.90.1	Function Documentation . . . . .	345
9.90.1.1	l4util_in8 . . . . .	345
9.90.1.2	l4util_in16 . . . . .	345
9.90.1.3	l4util_in32 . . . . .	346
9.90.1.4	l4util_ins8 . . . . .	346
9.90.1.5	l4util_ins16 . . . . .	346
9.90.1.6	l4util_ins32 . . . . .	346
9.90.1.7	l4util_out8 . . . . .	347
9.90.1.8	l4util_out16 . . . . .	347
9.90.1.9	l4util_out32 . . . . .	347
9.91	Bitmap graphics and fonts . . . . .	347
9.91.1	Detailed Description . . . . .	348
9.92	Functions for rendering bitmap data in frame buffers . . . . .	348
9.92.1	Typedef Documentation . . . . .	349
9.92.1.1	gfxbitmap_color_t . . . . .	349
9.92.1.2	gfxbitmap_color_pix_t . . . . .	349
9.92.2	Function Documentation . . . . .	349
9.92.2.1	gfxbitmap_convert_color . . . . .	349
9.92.2.2	gfxbitmap_fill . . . . .	350
9.92.2.3	gfxbitmap_bmap . . . . .	350
9.92.2.4	gfxbitmap_set . . . . .	350
9.92.2.5	gfxbitmap_copy . . . . .	351
9.93	Functions for rendering bitmap fonts to frame buffers . . . . .	351
9.93.1	Enumeration Type Documentation . . . . .	352
9.93.1.1	"@97 . . . . .	352
9.93.2	Function Documentation . . . . .	353
9.93.2.1	gfxbitmap_font_init . . . . .	353
9.93.2.2	gfxbitmap_font_get . . . . .	353
9.93.2.3	gfxbitmap_font_width . . . . .	353
9.93.2.4	gfxbitmap_font_height . . . . .	353
9.93.2.5	gfxbitmap_font_data . . . . .	354
9.93.2.6	gfxbitmap_font_text . . . . .	354
9.93.2.7	gfxbitmap_font_text_scale . . . . .	354
9.94	IRQ handling library . . . . .	355

9.95 Interface using direct functionality.	355
9.95.1 Function Documentation	356
9.95.1.1 l4irq_attach	356
9.95.1.2 l4irq_attach_ft	356
9.95.1.3 l4irq_attach_thread	357
9.95.1.4 l4irq_attach_thread_ft	357
9.95.1.5 l4irq_wait	357
9.95.1.6 l4irq_unmask_and_wait_any	358
9.95.1.7 l4irq_wait_any	358
9.95.1.8 l4irq_unmask	358
9.95.1.9 l4irq_detach	358
9.96 Interface for asynchronous ISR handlers.	359
9.96.1 Detailed Description	359
9.96.2 Function Documentation	359
9.96.2.1 l4irq_request	359
9.96.2.2 l4irq_release	360
9.97 Interface using direct functionality.	360
9.97.1 Function Documentation	361
9.97.1.1 l4irq_attach_cap	361
9.97.1.2 l4irq_attach_cap_ft	361
9.97.1.3 l4irq_attach_thread_cap	361
9.97.1.4 l4irq_attach_thread_cap_ft	362
9.98 Interface for asynchronous ISR handlers with a given IRQ capability.	362
9.98.1 Detailed Description	362
9.98.2 Function Documentation	363
9.98.2.1 l4irq_request_cap	363
9.99 Sigma0 API	363
9.99.1 Detailed Description	364
9.99.2 Enumeration Type Documentation	364
9.99.2.1 l4sigma0_return_flags_t	364
9.99.3 Function Documentation	365
9.99.3.1 l4sigma0_map_kip	365
9.99.3.2 l4sigma0_map_mem	365
9.99.3.3 l4sigma0_map_iomem	365
9.99.3.4 l4sigma0_map_anypage	366
9.99.3.5 l4sigma0_map_tbuf	366

9.99.3.6 l4sigma0_debug_dump . . . . .	367
9.99.3.7 l4sigma0_new_client . . . . .	367
9.99.3.8 l4sigma0_map_errstr . . . . .	367
9.100 Internal constants . . . . .	367
9.100.1 Detailed Description . . . . .	369
9.101 vCPU Support Library . . . . .	369
9.101.1 Detailed Description . . . . .	371
9.101.2 Enumeration Type Documentation . . . . .	371
9.101.2.1 l4vcpu_irq_state_t . . . . .	371
9.101.3 Function Documentation . . . . .	371
9.101.3.1 l4vcpu_state . . . . .	371
9.101.3.2 l4vcpu_irq_disable . . . . .	371
9.101.3.3 l4vcpu_irq_disable_save . . . . .	372
9.101.3.4 l4vcpu_irq_enable . . . . .	373
9.101.3.5 l4vcpu_irq_restore . . . . .	374
9.101.3.6 l4vcpu_halt . . . . .	374
9.101.3.7 l4vcpu_print_state . . . . .	375
9.101.3.8 l4vcpu_is_irq_entry . . . . .	375
9.101.3.9 l4vcpu_is_page_fault_entry . . . . .	375
9.102 Extended vCPU support . . . . .	376
9.102.1 Detailed Description . . . . .	376
9.102.2 Function Documentation . . . . .	376
9.102.2.1 l4vcpu_ext_alloc . . . . .	376
9.103 Shared Memory Library . . . . .	377
9.103.1 Detailed Description . . . . .	377
9.103.2 Function Documentation . . . . .	378
9.103.2.1 l4shmc_create . . . . .	378
9.103.2.2 l4shmc_attach . . . . .	378
9.103.2.3 l4shmc_attach_to . . . . .	379
9.103.2.4 l4shmc_connect_chunk_signal . . . . .	379
9.103.2.5 l4shmc_area_size . . . . .	379
9.104 Chunks . . . . .	380
9.104.1 Function Documentation . . . . .	381
9.104.1.1 l4shmc_add_chunk . . . . .	381
9.104.1.2 l4shmc_get_chunk . . . . .	381
9.104.1.3 l4shmc_get_chunk_to . . . . .	381

9.104.1.4 l4shmc_chunk_ptr . . . . .	382
9.104.1.5 l4shmc_chunk_capacity . . . . .	382
9.104.1.6 l4shmc_chunk_signal . . . . .	382
9.105Producer . . . . .	383
9.105.1 Function Documentation . . . . .	383
9.105.1.1 l4shmc_chunk_try_to_take . . . . .	383
9.105.1.2 l4shmc_chunk_ready . . . . .	384
9.105.1.3 l4shmc_chunk_ready_sig . . . . .	384
9.105.1.4 l4shmc_is_chunk_clear . . . . .	384
9.106Consumer . . . . .	385
9.106.1 Function Documentation . . . . .	385
9.106.1.1 l4shmc_enable_chunk . . . . .	385
9.106.1.2 l4shmc_wait_chunk . . . . .	386
9.106.1.3 l4shmc_wait_chunk_to . . . . .	386
9.106.1.4 l4shmc_wait_chunk_try . . . . .	386
9.106.1.5 l4shmc_chunk_consumed . . . . .	387
9.106.1.6 l4shmc_is_chunk_ready . . . . .	387
9.106.1.7 l4shmc_chunk_size . . . . .	387
9.107Signals . . . . .	388
9.107.1 Function Documentation . . . . .	389
9.107.1.1 l4shmc_add_signal . . . . .	389
9.107.1.2 l4shmc_attach_signal . . . . .	389
9.107.1.3 l4shmc_attach_signal_to . . . . .	389
9.107.1.4 l4shmc_get_signal_to . . . . .	390
9.107.1.5 l4shmc_signal_cap . . . . .	390
9.107.1.6 l4shmc_check_magic . . . . .	390
9.108Producer . . . . .	391
9.108.1 Function Documentation . . . . .	391
9.108.1.1 l4shmc_trigger . . . . .	391
9.109Consumer . . . . .	391
9.109.1 Function Documentation . . . . .	392
9.109.1.1 l4shmc_enable_signal . . . . .	392
9.109.1.2 l4shmc_wait_any . . . . .	392
9.109.1.3 l4shmc_wait_any_try . . . . .	393
9.109.1.4 l4shmc_wait_any_to . . . . .	393
9.109.1.5 l4shmc_wait_signal . . . . .	393

9.109.1.6 l4shmc_wait_signal_to . . . . .	394
9.109.1.7 l4shmc_wait_signal_try . . . . .	394
9.110 Integer Types . . . . .	394
9.110.1 Detailed Description . . . . .	396
9.110.2 Typedef Documentation . . . . .	396
9.110.2.1 l4_int8_t . . . . .	396
9.110.2.2 l4_uint8_t . . . . .	396
9.110.2.3 l4_int16_t . . . . .	397
9.110.2.4 l4_uint16_t . . . . .	397
9.110.2.5 l4_int32_t . . . . .	397
9.110.2.6 l4_uint32_t . . . . .	397
9.110.2.7 l4_int64_t . . . . .	397
9.110.2.8 l4_uint64_t . . . . .	397
<b>10 Namespace Documentation</b> . . . . .	<b>399</b>
10.1 cxx Namespace Reference . . . . .	399
10.1.1 Detailed Description . . . . .	401
10.2 cxx::Bits Namespace Reference . . . . .	401
10.2.1 Detailed Description . . . . .	401
10.3 L4 Namespace Reference . . . . .	401
10.3.1 Detailed Description . . . . .	404
10.3.2 Function Documentation . . . . .	404
10.3.2.1 kobject_typeid . . . . .	404
10.4 L4::Ipc_svr Namespace Reference . . . . .	405
10.4.1 Detailed Description . . . . .	405
10.4.2 Enumeration Type Documentation . . . . .	405
10.4.2.1 Reply_mode . . . . .	405
10.5 L4Re Namespace Reference . . . . .	406
10.5.1 Detailed Description . . . . .	407
10.6 L4Re::Vfs Namespace Reference . . . . .	407
10.6.1 Detailed Description . . . . .	408
<b>11 Data Structure Documentation</b> . . . . .	<b>409</b>
11.1 L4::Alloc_list Class Reference . . . . .	409
11.1.1 Detailed Description . . . . .	409
11.2 L4::Thread::Attr Class Reference . . . . .	409
11.2.1 Detailed Description . . . . .	410

---

11.2.2	Constructor & Destructor Documentation . . . . .	410
11.2.2.1	Attr . . . . .	410
11.2.3	Member Function Documentation . . . . .	410
11.2.3.1	pager . . . . .	410
11.2.3.2	pager . . . . .	411
11.2.3.3	exc_handler . . . . .	411
11.2.3.4	exc_handler . . . . .	412
11.2.3.5	bind . . . . .	412
11.2.3.6	ux_host_syscall . . . . .	413
11.3	L4Re::Util::Auto_cap< T > Struct Template Reference . . . . .	413
11.3.1	Detailed Description . . . . .	414
11.4	L4Re::Util::Auto_del_cap< T > Struct Template Reference . . . . .	414
11.4.1	Detailed Description . . . . .	415
11.5	cxx::Auto_ptr< T > Class Template Reference . . . . .	416
11.5.1	Detailed Description . . . . .	417
11.5.2	Member Typedef Documentation . . . . .	417
11.5.2.1	Ref_type . . . . .	417
11.5.3	Constructor & Destructor Documentation . . . . .	417
11.5.3.1	Auto_ptr . . . . .	417
11.5.3.2	Auto_ptr . . . . .	417
11.5.3.3	~Auto_ptr . . . . .	417
11.5.4	Member Function Documentation . . . . .	418
11.5.4.1	operator= . . . . .	418
11.5.4.2	operator* . . . . .	418
11.5.4.3	operator-> . . . . .	418
11.5.4.4	get . . . . .	418
11.5.4.5	release . . . . .	419
11.5.4.6	operator Priv_type * . . . . .	419
11.6	cxx::Avl_map< Key, Data, Compare, Alloc > Class Template Reference . . . . .	419
11.6.1	Detailed Description . . . . .	423
11.6.2	Constructor & Destructor Documentation . . . . .	423
11.6.2.1	Avl_map . . . . .	423
11.6.3	Member Function Documentation . . . . .	423
11.6.3.1	insert . . . . .	423
11.6.3.2	find_node . . . . .	424
11.6.3.3	lower_bound_node . . . . .	424

11.6.3.4	find . . . . .	424
11.6.3.5	remove . . . . .	425
11.6.3.6	erase . . . . .	425
11.6.3.7	operator[] . . . . .	425
11.6.3.8	operator[] . . . . .	425
11.7	cxx::Avl_set< Item, Compare, Alloc > Class Template Reference . . . . .	426
11.7.1	Detailed Description . . . . .	430
11.7.2	Constructor & Destructor Documentation . . . . .	430
11.7.2.1	Avl_set . . . . .	430
11.7.2.2	Avl_set . . . . .	431
11.7.3	Member Function Documentation . . . . .	431
11.7.3.1	insert . . . . .	431
11.7.3.2	remove . . . . .	432
11.7.3.3	find_node . . . . .	432
11.7.3.4	lower_bound_node . . . . .	433
11.7.3.5	begin . . . . .	433
11.7.3.6	end . . . . .	433
11.7.3.7	begin . . . . .	434
11.7.3.8	end . . . . .	434
11.7.3.9	rbegin . . . . .	434
11.7.3.10	rend . . . . .	435
11.7.3.11	rbegin . . . . .	435
11.7.3.12	rend . . . . .	435
11.8	cxx::Avl_tree< Node, Get_key, Compare > Class Template Reference . . . . .	435
11.8.1	Detailed Description . . . . .	438
11.8.2	Member Typedef Documentation . . . . .	439
11.8.2.1	Iterator . . . . .	439
11.8.3	Member Function Documentation . . . . .	439
11.8.3.1	insert . . . . .	439
11.8.3.2	remove . . . . .	440
11.9	cxx::Avl_tree_node Class Reference . . . . .	441
11.9.1	Detailed Description . . . . .	444
11.10	L4::Base_exception Class Reference . . . . .	444
11.10.1	Detailed Description . . . . .	445
11.11	cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference . . . . .	446
11.11.1	Detailed Description . . . . .	448

---

11.11.2 Member Enumeration Documentation . . . . .	448
11.11.2.1 "@48 . . . . .	448
11.11.3 Member Function Documentation . . . . .	449
11.11.3.1 total_objects . . . . .	449
11.11.3.2 free_objects . . . . .	449
11.12cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc > Class Template Reference	449
11.12.1 Detailed Description . . . . .	452
11.12.2 Member Enumeration Documentation . . . . .	452
11.12.2.1 "@49 . . . . .	452
11.12.3 Member Function Documentation . . . . .	452
11.12.3.1 alloc . . . . .	452
11.12.3.2 free . . . . .	453
11.12.3.3 total_objects . . . . .	453
11.12.3.4 free_objects . . . . .	453
11.13L4::Basic_registry Class Reference . . . . .	454
11.13.1 Detailed Description . . . . .	454
11.13.2 Member Function Documentation . . . . .	454
11.13.2.1 dispatch . . . . .	454
11.14L4Re::Vfs::Be_file Class Reference . . . . .	454
11.14.1 Detailed Description . . . . .	458
11.15L4Re::Vfs::Be_file_system Class Reference . . . . .	458
11.15.1 Detailed Description . . . . .	460
11.15.2 Constructor & Destructor Documentation . . . . .	461
11.15.2.1 Be_file_system . . . . .	461
11.15.2.2 ~Be_file_system . . . . .	461
11.15.3 Member Function Documentation . . . . .	461
11.15.3.1 type . . . . .	461
11.16cxx::Bitmap< BITS > Class Template Reference . . . . .	461
11.16.1 Detailed Description . . . . .	464
11.16.2 Constructor & Destructor Documentation . . . . .	464
11.16.2.1 Bitmap . . . . .	464
11.16.3 Member Function Documentation . . . . .	464
11.16.3.1 clear_all . . . . .	464
11.17cxx::Bitmap_base Class Reference . . . . .	464
11.17.1 Detailed Description . . . . .	466
11.17.2 Member Function Documentation . . . . .	466

11.17.2.1 words . . . . .	466
11.17.2.2 chars . . . . .	466
11.17.2.3 bit . . . . .	466
11.17.2.4 clear_bit . . . . .	467
11.17.2.5 set_bit . . . . .	467
11.17.2.6 operator[] . . . . .	468
11.17.2.7 scan_zero . . . . .	468
11.18L4::Bounds_error Class Reference . . . . .	469
11.18.1 Detailed Description . . . . .	472
11.19cxx::Bits::Bst< Node, Get_key, Compare > Class Template Reference . . . . .	472
11.19.1 Detailed Description . . . . .	476
11.19.2 Member Function Documentation . . . . .	476
11.19.2.1 dir . . . . .	476
11.19.2.2 dir . . . . .	476
11.19.2.3 begin . . . . .	477
11.19.2.4 end . . . . .	477
11.19.2.5 begin . . . . .	478
11.19.2.6 end . . . . .	478
11.19.2.7 rbegin . . . . .	478
11.19.2.8 rend . . . . .	479
11.19.2.9 rbegin . . . . .	479
11.19.2.10rend . . . . .	479
11.19.2.11find_node . . . . .	480
11.19.2.12lower_bound_node . . . . .	480
11.19.2.13find . . . . .	481
11.20cxx::Bits::Bst_node Class Reference . . . . .	482
11.20.1 Detailed Description . . . . .	485
11.21L4::Ipc::Buf_cp_in< T > Class Template Reference . . . . .	485
11.21.1 Detailed Description . . . . .	485
11.21.2 Constructor & Destructor Documentation . . . . .	485
11.21.2.1 Buf_cp_in . . . . .	485
11.22L4::Ipc::Buf_cp_out< T > Class Template Reference . . . . .	486
11.22.1 Detailed Description . . . . .	486
11.22.2 Constructor & Destructor Documentation . . . . .	486
11.22.2.1 Buf_cp_out . . . . .	486
11.22.3 Member Function Documentation . . . . .	487

---

11.22.3.1 size . . . . .	487
11.22.3.2 buf . . . . .	487
11.23L4::Ipc::Buf_in< T > Class Template Reference . . . . .	487
11.23.1 Detailed Description . . . . .	487
11.23.2 Constructor & Destructor Documentation . . . . .	488
11.23.2.1 Buf_in . . . . .	488
11.24L4::Cap< T > Class Template Reference . . . . .	488
11.24.1 Detailed Description . . . . .	490
11.24.2 Constructor & Destructor Documentation . . . . .	490
11.24.2.1 Cap . . . . .	490
11.24.2.2 Cap . . . . .	491
11.24.2.3 Cap . . . . .	491
11.24.3 Member Function Documentation . . . . .	491
11.24.3.1 move . . . . .	491
11.25L4Re::Cap_alloc Class Reference . . . . .	491
11.25.1 Detailed Description . . . . .	492
11.25.2 Member Function Documentation . . . . .	492
11.25.2.1 alloc . . . . .	492
11.25.2.2 alloc . . . . .	493
11.25.2.3 free . . . . .	493
11.25.2.4 get_cap_alloc . . . . .	494
11.26L4Re::Util::Cap_alloc_base Class Reference . . . . .	494
11.26.1 Detailed Description . . . . .	496
11.27L4::Cap_base Class Reference . . . . .	496
11.27.1 Detailed Description . . . . .	498
11.27.2 Member Enumeration Documentation . . . . .	498
11.27.2.1 No_init_type . . . . .	498
11.27.2.2 Cap_type . . . . .	498
11.27.3 Constructor & Destructor Documentation . . . . .	498
11.27.3.1 Cap_base . . . . .	498
11.27.3.2 Cap_base . . . . .	499
11.27.4 Member Function Documentation . . . . .	499
11.27.4.1 cap . . . . .	499
11.27.4.2 is_valid . . . . .	501
11.27.4.3 fpage . . . . .	501
11.27.4.4 snd_base . . . . .	502

11.27.4.5 validate . . . . .	503
11.27.4.6 validate . . . . .	503
11.27.5 Field Documentation . . . . .	503
11.27.5.1 _c . . . . .	503
11.28 cxx::Bitmap_base::Char< BITS > Class Template Reference . . . . .	504
11.28.1 Detailed Description . . . . .	504
11.29 L4Re::Video::Color_component Class Reference . . . . .	504
11.29.1 Detailed Description . . . . .	505
11.29.2 Constructor & Destructor Documentation . . . . .	505
11.29.2.1 Color_component . . . . .	505
11.29.3 Member Function Documentation . . . . .	505
11.29.3.1 size . . . . .	505
11.29.3.2 shift . . . . .	505
11.29.3.3 operator== . . . . .	506
11.29.3.4 get . . . . .	506
11.29.3.5 set . . . . .	506
11.29.3.6 dump . . . . .	506
11.30 L4::Com_error Class Reference . . . . .	507
11.30.1 Detailed Description . . . . .	510
11.30.2 Constructor & Destructor Documentation . . . . .	510
11.30.2.1 Com_error . . . . .	510
11.31 L4::Ipc_svr::Compound_reply Struct Reference . . . . .	510
11.31.1 Detailed Description . . . . .	511
11.32 L4Re::Console Class Reference . . . . .	511
11.32.1 Detailed Description . . . . .	513
11.33 L4Re::Util::Counting_cap_alloc< COUNTERTYPE > Class Template Reference . . . . .	513
11.33.1 Detailed Description . . . . .	514
11.34 L4Re::Dataspace Class Reference . . . . .	514
11.34.1 Detailed Description . . . . .	517
11.34.2 Member Enumeration Documentation . . . . .	517
11.34.2.1 Map_flags . . . . .	517
11.34.3 Member Function Documentation . . . . .	518
11.34.3.1 map . . . . .	518
11.34.3.2 map_region . . . . .	519
11.34.3.3 clear . . . . .	519
11.34.3.4 allocate . . . . .	520

11.34.3.5 <code>copy_in</code>	520
11.34.3.6 <code>phys</code>	521
11.34.3.7 <code>size</code>	522
11.34.3.8 <code>flags</code>	522
11.34.3.9 <code>info</code>	523
11.35L4Re::Util::Dataspace_svr Class Reference	524
11.35.1 Detailed Description	525
11.35.2 Member Function Documentation	525
11.35.2.1 <code>map</code>	525
11.35.2.2 <code>map_hook</code>	526
11.35.2.3 <code>phys</code>	526
11.35.2.4 <code>take</code>	526
11.35.2.5 <code>release</code>	526
11.35.2.6 <code>copy</code>	527
11.35.2.7 <code>clear</code>	527
11.35.2.8 <code>page_shift</code>	527
11.36L4Re::Debug_obj Class Reference	528
11.36.1 Detailed Description	529
11.36.2 Member Function Documentation	530
11.36.2.1 <code>debug</code>	530
11.37L4::Debugger Class Reference	530
11.37.1 Detailed Description	533
11.37.2 Member Function Documentation	533
11.37.2.1 <code>set_object_name</code>	533
11.37.2.2 <code>global_id</code>	533
11.37.2.3 <code>kobj_to_id</code>	533
11.37.2.4 <code>query_log_typeid</code>	534
11.37.2.5 <code>query_log_name</code>	534
11.37.2.6 <code>switch_log</code>	534
11.37.2.7 <code>get_object_name</code>	535
11.38L4::Ipc_svr::Default_loop_hooks Struct Reference	535
11.38.1 Detailed Description	535
11.39L4::Ipc_svr::Default_setup_wait Struct Reference	536
11.39.1 Detailed Description	536
11.40L4::Ipc_svr::Default_timeout Struct Reference	536
11.40.1 Detailed Description	537

11.41cxx::Bits::Direction Struct Reference . . . . .	537
11.41.1 Detailed Description . . . . .	538
11.41.2 Member Enumeration Documentation . . . . .	538
11.41.2.1 Direction_e . . . . .	538
11.41.3 Member Function Documentation . . . . .	538
11.41.3.1 operator! . . . . .	538
11.42L4Re::Vfs::Directory Class Reference . . . . .	539
11.42.1 Detailed Description . . . . .	541
11.42.2 Member Function Documentation . . . . .	541
11.42.2.1 faccessat . . . . .	541
11.42.2.2 mkdir . . . . .	542
11.42.2.3 unlink . . . . .	542
11.42.2.4 rename . . . . .	542
11.42.2.5 link . . . . .	543
11.42.2.6 symlink . . . . .	543
11.42.2.7 rmdir . . . . .	543
11.43L4::Element_already_exists Class Reference . . . . .	544
11.43.1 Detailed Description . . . . .	547
11.44L4::Element_not_found Class Reference . . . . .	547
11.44.1 Detailed Description . . . . .	550
11.45Elf32_Dyn Struct Reference . . . . .	550
11.45.1 Detailed Description . . . . .	550
11.45.2 Field Documentation . . . . .	550
11.45.2.1 d_val . . . . .	550
11.46Elf32_Ehdr Struct Reference . . . . .	550
11.46.1 Detailed Description . . . . .	551
11.46.2 Field Documentation . . . . .	552
11.46.2.1 e_phnum . . . . .	552
11.46.2.2 e_shnum . . . . .	552
11.47Elf32_Phdr Struct Reference . . . . .	552
11.47.1 Detailed Description . . . . .	553
11.48Elf32_Shdr Struct Reference . . . . .	553
11.48.1 Detailed Description . . . . .	554
11.49Elf32_Sym Struct Reference . . . . .	554
11.49.1 Detailed Description . . . . .	554
11.50Elf64_Dyn Struct Reference . . . . .	554

---

11.50.1 Detailed Description . . . . .	555
11.50.2 Field Documentation . . . . .	555
11.50.2.1 d_val . . . . .	555
11.51 Elf64_Ehdr Struct Reference . . . . .	555
11.51.1 Detailed Description . . . . .	556
11.51.2 Field Documentation . . . . .	556
11.51.2.1 e_phnum . . . . .	556
11.51.2.2 e_shnum . . . . .	556
11.52 Elf64_Phdr Struct Reference . . . . .	557
11.52.1 Detailed Description . . . . .	557
11.53 Elf64_Shdr Struct Reference . . . . .	557
11.53.1 Detailed Description . . . . .	558
11.54 Elf64_Sym Struct Reference . . . . .	558
11.54.1 Detailed Description . . . . .	559
11.55 L4Re::Env Class Reference . . . . .	559
11.55.1 Detailed Description . . . . .	563
11.55.2 Member Function Documentation . . . . .	563
11.55.2.1 env . . . . .	563
11.55.2.2 parent . . . . .	563
11.55.2.3 mem_alloc . . . . .	563
11.55.2.4 rm . . . . .	564
11.55.2.5 log . . . . .	564
11.55.2.6 main_thread . . . . .	564
11.55.2.7 task . . . . .	564
11.55.2.8 factory . . . . .	564
11.55.2.9 first_free_cap . . . . .	565
11.55.2.10 utcb_area . . . . .	565
11.55.2.11 first_free_utcb . . . . .	565
11.55.2.12 initial_caps . . . . .	565
11.55.2.13 get . . . . .	565
11.55.2.14 get_cap . . . . .	566
11.55.2.15 get_cap . . . . .	566
11.55.2.16 parent . . . . .	567
11.55.2.17 mem_alloc . . . . .	567
11.55.2.18 rm . . . . .	567
11.55.2.19 log . . . . .	567

11.55.2.20	main_thread . . . . .	567
11.55.2.21	factory . . . . .	568
11.55.2.22	first_free_cap . . . . .	568
11.55.2.23	utcb_area . . . . .	568
11.55.2.24	first_free_utcb . . . . .	568
11.55.2.25	scheduler . . . . .	568
11.55.2.26	scheduler . . . . .	569
11.55.2.27	initial_caps . . . . .	569
11.56	L4Re::Event_buffer_t< PAYLOAD >::Event Struct Reference . . . . .	569
11.56.1	Detailed Description . . . . .	569
11.57	L4Re::Event Class Reference . . . . .	570
11.57.1	Detailed Description . . . . .	573
11.57.2	Member Function Documentation . . . . .	573
11.57.2.1	get_buffer . . . . .	573
11.58	L4Re::Util::Event_buffer_consumer_t< PAYLOAD > Class Template Reference . . . . .	573
11.58.1	Detailed Description . . . . .	576
11.58.2	Member Function Documentation . . . . .	576
11.58.2.1	foreach_available_event . . . . .	576
11.58.2.2	process . . . . .	577
11.59	L4Re::Event_buffer_t< PAYLOAD > Class Template Reference . . . . .	577
11.59.1	Detailed Description . . . . .	580
11.59.2	Constructor & Destructor Documentation . . . . .	580
11.59.2.1	Event_buffer_t . . . . .	580
11.59.3	Member Function Documentation . . . . .	580
11.59.3.1	next . . . . .	580
11.59.3.2	put . . . . .	580
11.60	L4Re::Util::Event_buffer_t< PAYLOAD > Class Template Reference . . . . .	581
11.60.1	Detailed Description . . . . .	584
11.60.2	Member Function Documentation . . . . .	584
11.60.2.1	buf . . . . .	584
11.60.2.2	attach . . . . .	584
11.60.2.3	detach . . . . .	585
11.61	L4Re::Util::Event_t< PAYLOAD > Class Template Reference . . . . .	585
11.61.1	Detailed Description . . . . .	587
11.61.2	Member Enumeration Documentation . . . . .	587
11.61.2.1	Mode . . . . .	587

---

11.61.3 Member Function Documentation . . . . .	587
11.61.3.1 init . . . . .	587
11.61.3.2 buffer . . . . .	587
11.61.3.3 irq . . . . .	588
11.62L4::Exception_tracer Class Reference . . . . .	588
11.62.1 Detailed Description . . . . .	589
11.63L4::Factory Class Reference . . . . .	589
11.63.1 Detailed Description . . . . .	592
11.63.2 Member Function Documentation . . . . .	592
11.63.2.1 create . . . . .	592
11.63.2.2 create_task . . . . .	593
11.63.2.3 create_thread . . . . .	594
11.63.2.4 create_factory . . . . .	595
11.63.2.5 create_gate . . . . .	595
11.63.2.6 create_semaphore . . . . .	596
11.63.2.7 create_irq . . . . .	597
11.63.2.8 create_vm . . . . .	598
11.64L4Re::Vfs::File Class Reference . . . . .	599
11.64.1 Detailed Description . . . . .	601
11.65L4Re::Vfs::File_system Class Reference . . . . .	601
11.65.1 Detailed Description . . . . .	603
11.65.2 Member Function Documentation . . . . .	604
11.65.2.1 type . . . . .	604
11.65.2.2 mount . . . . .	604
11.66L4Re::Vfs::Fs Class Reference . . . . .	605
11.66.1 Detailed Description . . . . .	606
11.66.2 Member Function Documentation . . . . .	606
11.66.2.1 get_file . . . . .	606
11.66.2.2 alloc_fd . . . . .	607
11.66.2.3 set_fd . . . . .	607
11.66.2.4 free_fd . . . . .	607
11.66.2.5 mount . . . . .	607
11.67L4Re::Vfs::Generic_file Class Reference . . . . .	608
11.67.1 Detailed Description . . . . .	610
11.67.2 Member Function Documentation . . . . .	610
11.67.2.1 unlock_all_locks . . . . .	610

11.67.2.2 fstat64 . . . . .	611
11.67.2.3 fchmod . . . . .	611
11.67.2.4 get_status_flags . . . . .	611
11.67.2.5 set_status_flags . . . . .	611
11.68 gfxbitmap_offset Struct Reference . . . . .	612
11.68.1 Detailed Description . . . . .	612
11.69 L4Re::Video::Goos Class Reference . . . . .	612
11.69.1 Detailed Description . . . . .	615
11.69.2 Member Enumeration Documentation . . . . .	615
11.69.2.1 Flags . . . . .	615
11.69.3 Member Function Documentation . . . . .	616
11.69.3.1 info . . . . .	616
11.69.3.2 get_static_buffer . . . . .	616
11.69.3.3 create_buffer . . . . .	616
11.69.3.4 delete_buffer . . . . .	617
11.69.3.5 create_view . . . . .	617
11.69.3.6 delete_view . . . . .	617
11.69.3.7 view . . . . .	617
11.70 L4Re::Util::Video::Goos_svr Class Reference . . . . .	618
11.70.1 Detailed Description . . . . .	620
11.70.2 Member Function Documentation . . . . .	620
11.70.2.1 get_fb . . . . .	620
11.70.2.2 screen_info . . . . .	621
11.70.2.3 view_info . . . . .	621
11.70.2.4 refresh . . . . .	621
11.70.2.5 dispatch . . . . .	622
11.70.2.6 init_infos . . . . .	622
11.71 L4::Icu Class Reference . . . . .	622
11.71.1 Detailed Description . . . . .	625
11.71.2 Member Function Documentation . . . . .	625
11.71.2.1 bind . . . . .	625
11.71.2.2 unbind . . . . .	626
11.71.2.3 info . . . . .	627
11.71.2.4 msi_info . . . . .	627
11.71.2.5 mask . . . . .	628
11.71.2.6 unmask . . . . .	629

11.71.2.7 set_mode . . . . .	629
11.72L4::Ipc_svr::Ignore_errors Struct Reference . . . . .	630
11.72.1 Detailed Description . . . . .	631
11.73L4::Icu::Info Class Reference . . . . .	631
11.73.1 Detailed Description . . . . .	632
11.74L4Re::Video::Goos::Info Struct Reference . . . . .	632
11.74.1 Detailed Description . . . . .	634
11.74.2 Member Function Documentation . . . . .	634
11.74.2.1 auto_refresh . . . . .	634
11.75L4Re::Video::View::Info Struct Reference . . . . .	635
11.75.1 Detailed Description . . . . .	638
11.75.2 Field Documentation . . . . .	638
11.75.2.1 flags . . . . .	638
11.76L4::Invalid_capability Class Reference . . . . .	638
11.76.1 Detailed Description . . . . .	641
11.76.2 Constructor & Destructor Documentation . . . . .	641
11.76.2.1 Invalid_capability . . . . .	641
11.76.3 Member Function Documentation . . . . .	641
11.76.3.1 cap . . . . .	641
11.77L4::IOModifier Class Reference . . . . .	641
11.77.1 Detailed Description . . . . .	641
11.78L4::Ipc::Iostream Class Reference . . . . .	642
11.78.1 Detailed Description . . . . .	645
11.78.2 Constructor & Destructor Documentation . . . . .	645
11.78.2.1 Iostream . . . . .	645
11.78.3 Member Function Documentation . . . . .	646
11.78.3.1 reset . . . . .	646
11.78.3.2 call . . . . .	646
11.78.3.3 reply_and_wait . . . . .	647
11.78.3.4 reply_and_wait . . . . .	648
11.79L4::Ipc_gate Class Reference . . . . .	649
11.79.1 Detailed Description . . . . .	650
11.79.2 Member Function Documentation . . . . .	651
11.79.2.1 bind_thread . . . . .	651
11.79.2.2 get_infos . . . . .	651
11.80L4::Irq Class Reference . . . . .	651

11.80.1 Detailed Description . . . . .	654
11.80.2 Member Function Documentation . . . . .	654
11.80.2.1 attach . . . . .	654
11.80.2.2 chain . . . . .	655
11.80.2.3 detach . . . . .	656
11.80.2.4 receive . . . . .	656
11.80.2.5 wait . . . . .	657
11.80.2.6 unmask . . . . .	658
11.80.2.7 trigger . . . . .	658
11.81L4::Ipc::Istream Class Reference . . . . .	659
11.81.1 Detailed Description . . . . .	663
11.81.2 Constructor & Destructor Documentation . . . . .	663
11.81.2.1 Istream . . . . .	663
11.81.3 Member Function Documentation . . . . .	663
11.81.3.1 reset . . . . .	663
11.81.3.2 get . . . . .	663
11.81.3.3 skip . . . . .	664
11.81.3.4 get . . . . .	664
11.81.3.5 get . . . . .	664
11.81.3.6 tag . . . . .	665
11.81.3.7 tag . . . . .	665
11.81.3.8 wait . . . . .	665
11.81.3.9 wait . . . . .	666
11.81.3.10receive . . . . .	667
11.82L4Re::Util::Item_alloc_base Class Reference . . . . .	668
11.82.1 Detailed Description . . . . .	669
11.83cxx::List< D, Alloc >::Iter Class Reference . . . . .	670
11.83.1 Detailed Description . . . . .	672
11.84cxx::List_item::Iter Class Reference . . . . .	672
11.84.1 Detailed Description . . . . .	675
11.84.2 Member Function Documentation . . . . .	675
11.84.2.1 remove_me . . . . .	675
11.85L4::Kobject Class Reference . . . . .	675
11.85.1 Detailed Description . . . . .	676
11.85.2 Member Function Documentation . . . . .	676
11.85.2.1 cap . . . . .	676

---

11.85.2.2 dec_refcnt . . . . .	679
11.85.3 Friends And Related Function Documentation . . . . .	679
11.85.3.1 kobject_typeid . . . . .	679
11.86L4::Kobject_2t< Derived, Base1, Base2, PROTO > Class Template Reference . . . . .	680
11.86.1 Detailed Description . . . . .	681
11.86.2 Friends And Related Function Documentation . . . . .	681
11.86.2.1 kobject_typeid . . . . .	681
11.87L4::Kobject_t< Derived, Base, PROTO > Class Template Reference . . . . .	682
11.87.1 Detailed Description . . . . .	682
11.87.2 Friends And Related Function Documentation . . . . .	683
11.87.2.1 kobject_typeid . . . . .	683
11.88I4_buf_regs_t Struct Reference . . . . .	683
11.88.1 Detailed Description . . . . .	683
11.89I4_exc_regs_t Struct Reference . . . . .	684
11.89.1 Detailed Description . . . . .	686
11.89.2 Field Documentation . . . . .	686
11.89.2.1 flags . . . . .	686
11.90I4_fpage_t Union Reference . . . . .	686
11.90.1 Detailed Description . . . . .	687
11.91I4_icu_info_t Struct Reference . . . . .	687
11.91.1 Detailed Description . . . . .	688
11.91.2 Field Documentation . . . . .	688
11.91.2.1 features . . . . .	688
11.92I4_kernel_info_mem_desc_t Struct Reference . . . . .	688
11.92.1 Detailed Description . . . . .	688
11.93I4_kernel_info_t Struct Reference . . . . .	689
11.93.1 Detailed Description . . . . .	691
11.93.2 Field Documentation . . . . .	691
11.93.2.1 l4_config . . . . .	691
11.93.2.2 kdebug_config . . . . .	692
11.93.2.3 kdebug_permission . . . . .	692
11.94I4_msg_regs_t Struct Reference . . . . .	692
11.94.1 Detailed Description . . . . .	693
11.95I4_mshtag_t Struct Reference . . . . .	693
11.95.1 Detailed Description . . . . .	694
11.95.2 Member Function Documentation . . . . .	694

11.95.2.1 flags . . . . .	694
11.96l4_rt_preemption_t Union Reference . . . . .	695
11.96.1 Detailed Description . . . . .	695
11.97l4_rt_preemption_val32_t Struct Reference . . . . .	696
11.97.1 Detailed Description . . . . .	696
11.98l4_rt_preemption_val_t Struct Reference . . . . .	696
11.98.1 Detailed Description . . . . .	697
11.99l4_sched_cpu_set_t Struct Reference . . . . .	697
11.99.1 Detailed Description . . . . .	697
11.100l4_sched_param_t Struct Reference . . . . .	697
11.100.1 Detailed Description . . . . .	698
11.101l4_snd_fpage_t Struct Reference . . . . .	699
11.101.1 Detailed Description . . . . .	699
11.102l4_thread_regs_t Struct Reference . . . . .	700
11.102.1 Detailed Description . . . . .	701
11.103l4_timeout_s Struct Reference . . . . .	701
11.103.1 Detailed Description . . . . .	701
11.104l4_timeout_t Union Reference . . . . .	701
11.104.1 Detailed Description . . . . .	702
11.105l4_tracebuffer_status_t Struct Reference . . . . .	703
11.105.1 Detailed Description . . . . .	707
11.105.2 Field Documentation . . . . .	707
11.105.2.1 tracebuffer0 . . . . .	707
11.105.2.2 size0 . . . . .	707
11.105.2.3 version0 . . . . .	707
11.105.2.4 tracebuffer1 . . . . .	707
11.105.2.5 size1 . . . . .	707
11.105.2.6 version1 . . . . .	707
11.105.2.7 cnt_iobmap_tlb_flush . . . . .	708
11.105.3 tracebuffer_status_window_t Struct Reference . . . . .	708
11.105.4 Detailed Description . . . . .	708
11.106l4_vcon_attr_t Struct Reference . . . . .	708
11.106.1 Detailed Description . . . . .	709
11.107l4_vcpu_ipc_regs_t Struct Reference . . . . .	709
11.107.1 Detailed Description . . . . .	710
11.108l4_vcpu_regs_t Struct Reference . . . . .	711

11.109.1. Detailed Description . . . . .	712
11.109.2. Field Documentation . . . . .	713
11.109.2.1di . . . . .	713
11.109.2.2si . . . . .	713
11.109.2.3bp . . . . .	713
11.109.2.4bx . . . . .	713
11.109.2.5dx . . . . .	713
11.109.2.6cx . . . . .	713
11.109.2.7ax . . . . .	713
11.110. vcpu_state_t Struct Reference . . . . .	714
11.110.1. Detailed Description . . . . .	717
11.111. vhw_descriptor Struct Reference . . . . .	717
11.111.1. Detailed Description . . . . .	719
11.111.2. Field Documentation . . . . .	719
11.111.2.1magic . . . . .	719
11.111.2.2version . . . . .	719
11.111.2.3count . . . . .	719
11.111.2.4descs . . . . .	720
11.112. vhw_entry Struct Reference . . . . .	720
11.112.1. Detailed Description . . . . .	720
11.112.2. Field Documentation . . . . .	721
11.112.2.1type . . . . .	721
11.112.2.2provider_pid . . . . .	721
11.112.2.3mem_start . . . . .	721
11.112.2.4mem_size . . . . .	721
11.112.2.5irq_no . . . . .	721
11.112.2.6fd . . . . .	722
11.113. vm_state Struct Reference . . . . .	722
11.113.1. Detailed Description . . . . .	722
11.114. vm_svm_vmcb_control_area Struct Reference . . . . .	722
11.114.1. Detailed Description . . . . .	722
11.115. vm_svm_vmcb_state_save_area Struct Reference . . . . .	722
11.115.1. Detailed Description . . . . .	724
11.116. vm_svm_vmcb_state_save_area_seg Struct Reference . . . . .	724
11.116.1. Detailed Description . . . . .	724
11.117. vm_svm_vmcb_t Struct Reference . . . . .	724

11.117. <a href="#">Detailed Description</a>	726
11.118 <a href="#">re_aux_t Struct Reference</a>	726
11.118. <a href="#">Detailed Description</a>	726
11.119 <a href="#">re_ds_stats_t Struct Reference</a>	726
11.119. <a href="#">Detailed Description</a>	727
11.120 <a href="#">re_elf_aux_mword_t Struct Reference</a>	727
11.120. <a href="#">Detailed Description</a>	727
11.121 <a href="#">re_elf_aux_t Struct Reference</a>	727
11.121. <a href="#">Detailed Description</a>	727
11.122 <a href="#">re_elf_aux_vma_t Struct Reference</a>	727
11.122. <a href="#">Detailed Description</a>	728
11.123 <a href="#">re_env_cap_entry_t Struct Reference</a>	728
11.123. <a href="#">Detailed Description</a>	728
11.123. <a href="#">Constructor &amp; Destructor Documentation</a>	729
11.123.2. <a href="#">ll4re_env_cap_entry_t</a>	729
11.123. <a href="#">Field Documentation</a>	729
11.123.3. <a href="#">lflags</a>	729
11.124 <a href="#">re_env_t Struct Reference</a>	729
11.124. <a href="#">Detailed Description</a>	731
11.125 <a href="#">re_event_t Struct Reference</a>	731
11.125. <a href="#">Detailed Description</a>	732
11.126 <a href="#">re_video_color_component_t Struct Reference</a>	732
11.126. <a href="#">Detailed Description</a>	732
11.127 <a href="#">re_video_goops_info_t Struct Reference</a>	732
11.127. <a href="#">Detailed Description</a>	734
11.128 <a href="#">re_video_pixel_info_t Struct Reference</a>	734
11.128. <a href="#">Detailed Description</a>	735
11.129 <a href="#">re_video_view_info_t Struct Reference</a>	736
11.129. <a href="#">Detailed Description</a>	738
11.130 <a href="#">re_video_view_t Struct Reference</a>	738
11.130. <a href="#">Detailed Description</a>	738
11.131 <a href="#">util_idt_desc_t Struct Reference</a>	739
11.131. <a href="#">Detailed Description</a>	739
11.132 <a href="#">util_idt_header_t Struct Reference</a>	739
11.132. <a href="#">Detailed Description</a>	740
11.133 <a href="#">util_mb_addr_range_t Struct Reference</a>	741

11.133.1Detailed Description . . . . .	741
11.134util_mb_apm_t Struct Reference . . . . .	741
11.134.1Detailed Description . . . . .	741
11.135util_mb_drive_t Struct Reference . . . . .	742
11.135.1Detailed Description . . . . .	742
11.135.2Field Documentation . . . . .	742
11.135.2.1drive_number . . . . .	742
11.135.2.2drive_mode . . . . .	742
11.135.2.3drive_cylinders . . . . .	743
11.136util_mb_info_t Struct Reference . . . . .	743
11.136.1Detailed Description . . . . .	744
11.137util_mb_mod_t Struct Reference . . . . .	744
11.137.1Detailed Description . . . . .	745
11.137.2Field Documentation . . . . .	745
11.137.2.1mod_start . . . . .	745
11.137.2.2mod_end . . . . .	745
11.138util_mb_vbe_ctrl_t Struct Reference . . . . .	745
11.138.1Detailed Description . . . . .	745
11.139util_mb_vbe_mode_t Struct Reference . . . . .	746
11.139.1Detailed Description . . . . .	747
11.140xx::List< D, Alloc > Class Template Reference . . . . .	747
11.140.1Detailed Description . . . . .	748
11.140.2Member Function Documentation . . . . .	748
11.140.2.1push_back . . . . .	748
11.140.2.2push_front . . . . .	748
11.140.2.3remove . . . . .	749
11.140.2.4size . . . . .	749
11.140.2.5operator[] . . . . .	749
11.140.2.6operator[] . . . . .	750
11.140.2.7items . . . . .	750
11.141xx::List_alloc Class Reference . . . . .	750
11.141.1Detailed Description . . . . .	750
11.141.2Constructor & Destructor Documentation . . . . .	750
11.141.2.1List_alloc . . . . .	750
11.141.3Member Function Documentation . . . . .	751
11.141.3.1free . . . . .	751

11.141.3.2alloc . . . . .	751
11.141.3.3avail . . . . .	751
11.142xx::List_item Class Reference . . . . .	752
11.142.1Detailed Description . . . . .	753
11.142.2Member Function Documentation . . . . .	753
11.142.2.1get_prev_item . . . . .	753
11.142.2.2get_next_item . . . . .	753
11.142.2.3insert_prev_item . . . . .	753
11.142.2.4insert_next_item . . . . .	754
11.142.2.5remove_me . . . . .	754
11.142.2.6push_back . . . . .	754
11.142.2.7push_front . . . . .	755
11.142.2.8remove . . . . .	755
11.143xx::Log Class Reference . . . . .	756
11.143.1Detailed Description . . . . .	759
11.143.2Member Function Documentation . . . . .	759
11.143.2.1lprintf . . . . .	759
11.143.2.2print . . . . .	759
11.144L4::Factory::Lstr Struct Reference . . . . .	759
11.144.1Detailed Description . . . . .	760
11.145xx::Lt_functor< Obj > Struct Template Reference . . . . .	760
11.145.1Detailed Description . . . . .	760
11.146xx::Mem_alloc Class Reference . . . . .	760
11.146.1Detailed Description . . . . .	763
11.146.2Member Enumeration Documentation . . . . .	763
11.146.2.1Mem_alloc_flags . . . . .	763
11.146.3Member Function Documentation . . . . .	763
11.146.3.1alloc . . . . .	763
11.146.3.2free . . . . .	764
11.147L4::Kip::Mem_desc Class Reference . . . . .	764
11.147.1Detailed Description . . . . .	765
11.147.2Constructor & Destructor Documentation . . . . .	765
11.147.2.1Mem_desc . . . . .	765
11.147.3Member Function Documentation . . . . .	766
11.147.3.1first . . . . .	766
11.147.3.2count . . . . .	766

11.147.3.3count . . . . .	766
11.147.3.4start . . . . .	767
11.147.3.5end . . . . .	767
11.147.3.6size . . . . .	768
11.147.3.7type . . . . .	768
11.147.3.8sub_type . . . . .	768
11.147.3.9is_virtual . . . . .	768
11.147.3.10et . . . . .	769
11.148.4::Meta Class Reference . . . . .	769
11.148.1Detailed Description . . . . .	772
11.148.2Member Function Documentation . . . . .	772
11.148.2.1num_interfaces . . . . .	772
11.148.2.2interface . . . . .	772
11.148.2.3supports . . . . .	773
11.149.4Re::Vfs::Mman Class Reference . . . . .	774
11.149.1Detailed Description . . . . .	776
11.150.4::Thread::Modify_senders Class Reference . . . . .	776
11.150.1Detailed Description . . . . .	776
11.150.2Member Function Documentation . . . . .	776
11.150.2.1add . . . . .	776
11.151.4::Ipc::Msg_ptr< T > Class Template Reference . . . . .	777
11.151.1Detailed Description . . . . .	777
11.151.2Constructor & Destructor Documentation . . . . .	777
11.151.2.1Msg_ptr . . . . .	777
11.152.4Re::Util::Names::Name Class Reference . . . . .	778
11.152.1Detailed Description . . . . .	778
11.153.4Re::Namespace Class Reference . . . . .	778
11.153.1Detailed Description . . . . .	781
11.153.2Member Enumeration Documentation . . . . .	781
11.153.2.1Register_flags . . . . .	781
11.153.3Member Function Documentation . . . . .	781
11.153.3.1query . . . . .	781
11.153.3.2query . . . . .	782
11.153.3.3register_obj . . . . .	782
11.154xx::New_allocator< _Type > Class Template Reference . . . . .	783
11.154.1Detailed Description . . . . .	783

11.15 <del>L</del> 4::Factory::Nil Struct Reference . . . . .	783
11.155. Detailed Description . . . . .	783
11.156xx::Avl_set< Item, Compare, Alloc >::Node Class Reference . . . . .	783
11.156. Detailed Description . . . . .	784
11.156. Member Function Documentation . . . . .	784
11.156.2. lvalid . . . . .	784
11.157xx::Nothrow Class Reference . . . . .	784
11.157. Detailed Description . . . . .	784
11.158 <del>L</del> 4Re::Vfs::Ops Class Reference . . . . .	785
11.158. Detailed Description . . . . .	786
11.159 <del>L</del> 4::Ipc::Ostream Class Reference . . . . .	787
11.159. Detailed Description . . . . .	790
11.159. Member Function Documentation . . . . .	791
11.159.2. lput . . . . .	791
11.159.2.2put . . . . .	791
11.159.2.3tag . . . . .	791
11.159.2.4tag . . . . .	791
11.159.2.5send . . . . .	792
11.160 <del>L</del> 4::Out_of_memory Class Reference . . . . .	792
11.160. Detailed Description . . . . .	795
11.161xx::Pair< First, Second > Struct Template Reference . . . . .	795
11.161. Detailed Description . . . . .	796
11.161. Constructor & Destructor Documentation . . . . .	796
11.161.2. lPair . . . . .	796
11.162xx::Pair_first_compare< Cmp, Typ > Class Template Reference . . . . .	796
11.162. Detailed Description . . . . .	796
11.162. Constructor & Destructor Documentation . . . . .	797
11.162.2. lPair_first_compare . . . . .	797
11.162. Member Function Documentation . . . . .	797
11.162.3. loperator() . . . . .	797
11.163 <del>L</del> 4Re::Parent Class Reference . . . . .	797
11.163. Detailed Description . . . . .	799
11.163. Member Function Documentation . . . . .	800
11.163.2. lsignal . . . . .	800
11.164 <del>L</del> 4Re::Video::Pixel_info Class Reference . . . . .	800
11.164. Detailed Description . . . . .	803

---

11.164.1Constructor & Destructor Documentation . . . . .	803
11.164.2.1Pixel_info . . . . .	803
11.164.2.2Pixel_info . . . . .	803
11.164.3Member Function Documentation . . . . .	804
11.164.3.1r . . . . .	804
11.164.3.2g . . . . .	804
11.164.3.3b . . . . .	804
11.164.3.4a . . . . .	804
11.164.3.5bytes_per_pixel . . . . .	804
11.164.3.6bits_per_pixel . . . . .	805
11.164.3.7has_alpha . . . . .	805
11.164.3.8t . . . . .	805
11.164.3.9g . . . . .	805
11.164.3.10l . . . . .	805
11.164.3.11d . . . . .	806
11.164.3.12bytes_per_pixel . . . . .	806
11.164.3.13operator== . . . . .	806
11.164.3.14dump . . . . .	806
11.165.4Re::Util::Ref_cap< T > Struct Template Reference . . . . .	807
11.165.1Detailed Description . . . . .	807
11.166.4Re::Util::Ref_del_cap< T > Struct Template Reference . . . . .	807
11.166.1Detailed Description . . . . .	807
11.167.4Re::Vfs::Regular_file Class Reference . . . . .	808
11.167.1Detailed Description . . . . .	810
11.167.2Member Function Documentation . . . . .	810
11.167.2.1data_space . . . . .	810
11.167.2.2readv . . . . .	811
11.167.2.3writev . . . . .	811
11.167.2.4seek64 . . . . .	811
11.167.2.5truncate64 . . . . .	811
11.167.2.6fsync . . . . .	811
11.167.2.7fdatasync . . . . .	812
11.167.2.8get_lock . . . . .	812
11.167.2.9set_lock . . . . .	812
11.167.8Re::Rm Class Reference . . . . .	812
11.167.9Detailed Description . . . . .	816

11.168.2Member Enumeration Documentation . . . . .	816
11.168.2.1IDetach_result . . . . .	816
11.168.2.2Region_flags . . . . .	816
11.168.2.3Attach_flags . . . . .	816
11.168.2.4Detach_flags . . . . .	817
11.168.3Member Function Documentation . . . . .	817
11.168.3.1reserve_area . . . . .	817
11.168.3.2reserve_area . . . . .	818
11.168.3.3free_area . . . . .	819
11.168.3.4attach . . . . .	819
11.168.3.5attach . . . . .	820
11.168.3.6detach . . . . .	820
11.168.3.7detach . . . . .	821
11.168.3.8detach . . . . .	821
11.168.3.9find . . . . .	822
11.169L4::Runtime_error Class Reference . . . . .	823
11.169.1Detailed Description . . . . .	825
11.170L4::Factory::S Class Reference . . . . .	825
11.170.1Detailed Description . . . . .	827
11.170.2Constructor & Destructor Documentation . . . . .	827
11.170.2.1S . . . . .	827
11.170.3Member Function Documentation . . . . .	828
11.170.3.1operator l4_mshtag_t . . . . .	828
11.170.3.2operator<< . . . . .	828
11.170.3.3operator<< . . . . .	828
11.170.3.4operator<< . . . . .	828
11.170.3.5operator<< . . . . .	828
11.170.3.6operator<< . . . . .	829
11.171L4::Scheduler Class Reference . . . . .	829
11.171.1Detailed Description . . . . .	831
11.171.2Member Function Documentation . . . . .	832
11.171.2.1info . . . . .	832
11.171.2.2run_thread . . . . .	832
11.171.2.3idle_time . . . . .	833
11.171.2.4is_online . . . . .	834
11.172L4::Server< LOOP_HOOKS > Class Template Reference . . . . .	834

---

11.172.1	Detailed Description	836
11.172.2	Constructor & Destructor Documentation	836
11.172.2.1	IServer	836
11.172.3	Member Function Documentation	836
11.172.3.1	internal_loop	836
11.173	Class Reference	837
11.173.1	Detailed Description	839
11.173.2	Member Function Documentation	839
11.173.2.1	dispatch	839
11.174	xx::Slab< Type, Slab_size, Max_free, Alloc > Class Template Reference	839
11.174.1	Detailed Description	842
11.174.2	Member Function Documentation	842
11.174.2.1	lalloc	842
11.174.2.2	free	842
11.175	xx::Slab_static< Type, Slab_size, Max_free, Alloc > Class Template Reference	843
11.175.1	Detailed Description	845
11.175.2	Member Function Documentation	845
11.175.2.1	lalloc	845
11.176	4::Ipc::Small_buf Class Reference	845
11.176.1	Detailed Description	845
11.177	4::Smart_cap< T, SMART > Class Template Reference	846
11.177.1	Detailed Description	849
11.177.2	Constructor & Destructor Documentation	849
11.177.2.1	ISmart_cap	849
11.178	4Re::Smart_cap_auto< Unmap_flags > Class Template Reference	849
11.178.1	Detailed Description	850
11.179	4Re::Util::Smart_cap_auto< Unmap_flags > Class Template Reference	851
11.179.1	Detailed Description	851
11.180	4Re::Util::Smart_count_cap< Unmap_flags > Class Template Reference	851
11.180.1	Detailed Description	852
11.181	4Re::Vfs::Special_file Class Reference	852
11.181.1	Detailed Description	854
11.181.2	Member Function Documentation	854
11.181.2.1	ioctl	854
11.182	4vcpu::State Class Reference	854
11.182.1	Detailed Description	855

11.182.1Constructor & Destructor Documentation . . . . .	855
11.182.2.IState . . . . .	855
11.182.3Member Function Documentation . . . . .	855
11.182.3.1add . . . . .	855
11.182.3.2clear . . . . .	855
11.182.3.3set . . . . .	855
11.183.1Re::Dataspace::Stats Struct Reference . . . . .	856
11.183.1Detailed Description . . . . .	856
11.184.1L4::String Class Reference . . . . .	856
11.184.1Detailed Description . . . . .	856
11.185.1xx::List_item::T_iter< T, Poly > Class Template Reference . . . . .	857
11.185.1Detailed Description . . . . .	860
11.185.2Member Function Documentation . . . . .	860
11.185.2.1remove_me . . . . .	860
11.186.1L4::Task Class Reference . . . . .	860
11.186.1Detailed Description . . . . .	863
11.186.2Member Function Documentation . . . . .	863
11.186.2.1map . . . . .	863
11.186.2.2unmap . . . . .	864
11.186.2.3unmap_batch . . . . .	865
11.186.2.4cap_valid . . . . .	866
11.186.2.5cap_has_child . . . . .	867
11.186.2.6cap_equal . . . . .	867
11.186.2.7add_ku_mem . . . . .	868
11.187.1L4::Thread Class Reference . . . . .	869
11.187.1Detailed Description . . . . .	872
11.187.2Member Function Documentation . . . . .	872
11.187.2.1lex_regs . . . . .	872
11.187.2.2ex_regs . . . . .	873
11.187.2.3control . . . . .	874
11.187.2.4switch_to . . . . .	874
11.187.2.5stats_time . . . . .	875
11.187.2.6vcpu_resume_start . . . . .	875
11.187.2.7vcpu_resume_commit . . . . .	876
11.187.2.8vcpu_control . . . . .	876
11.187.2.9vcpu_control_ext . . . . .	877

11.187.2. <code>lregister_del_irq</code>	877
11.187.2. <code>lhodify_senders</code>	878
11.188 <del>4</del> :: <code>Type_info</code> Struct Reference	879
11.188.1.Detailed Description	879
11.189 <del>4</del> :: <code>Unknown_error</code> Class Reference	879
11.189.1.Detailed Description	881
11.190 <del>4</del> :: <code>Vcon</code> Class Reference	882
11.190.1.Detailed Description	885
11.190.2.Member Function Documentation	885
11.190.2.1 <code>send</code>	885
11.190.2.2 <code>write</code>	886
11.190.2.3 <code>read</code>	886
11.190.2.4 <code>set_attr</code>	887
11.190.2.5 <code>get_attr</code>	888
11.191 <del>4</del> Re::Util:: <code>Vcon_svr&lt; SVR &gt;</code> Class Template Reference	889
11.191.1.Detailed Description	889
11.191.2.Member Function Documentation	889
11.191.2.1 <code>dispatch</code>	889
11.192 <del>4</del> vcpu:: <code>Vcpu</code> Class Reference	890
11.192.1.Detailed Description	894
11.192.2.Member Function Documentation	894
11.192.2.1 <code>irq_disable_save</code>	894
11.192.2.2 <code>state</code>	895
11.192.2.3 <code>state</code>	895
11.192.2.4 <code>saved_state</code>	895
11.192.2.5 <code>saved_state</code>	895
11.192.2.6 <code>irq_enable</code>	895
11.192.2.7 <code>irq_restore</code>	896
11.192.2.8 <code>halt</code>	896
11.192.2.9 <code>task</code>	896
11.192.2.10 <code>page_fault_entry</code>	896
11.192.2.11 <code>irq_entry</code>	896
11.192.2.12 <code>l2</code>	897
11.192.2.13 <code>l3</code>	897
11.192.2.14 <code>l4</code>	897
11.192.2.15 <code>l5</code>	897

11.192.2.1entry_sp . . . . .	897
11.192.2.1entry_ip . . . . .	898
11.192.2.1ext_alloc . . . . .	898
11.192.2.1last . . . . .	898
11.192.2.2last . . . . .	898
11.192.4Re::Video::View Class Reference . . . . .	899
11.193.1Detailed Description . . . . .	902
11.193.2Member Enumeration Documentation . . . . .	902
11.193.2.1Flags . . . . .	902
11.193.2.2V_flags . . . . .	902
11.193.3Member Function Documentation . . . . .	902
11.193.3.1info . . . . .	902
11.193.3.2set_info . . . . .	903
11.193.3.3set_viewport . . . . .	903
11.193.3.4stack . . . . .	903
11.193.3.5refresh . . . . .	904
11.194L4::Vm Class Reference . . . . .	904
11.194.1Detailed Description . . . . .	906
11.194.2Member Function Documentation . . . . .	907
11.194.2.1run . . . . .	907
11.195xx::Bitmap_base::Word< BITS > Class Template Reference . . . . .	907
11.195.1Detailed Description . . . . .	908
<b>12 Example Documentation</b>	<b>909</b>
12.1 examples/clntsrv/client.cc . . . . .	909
12.2 examples/clntsrv/clntsrv.cfg . . . . .	910
12.3 examples/clntsrv/server.cc . . . . .	911
12.4 examples/libs/l4re/c++/mem_alloc/ma+rm.cc . . . . .	912
12.5 examples/libs/l4re/c++/shared_ds/ds_clnt.cc . . . . .	913
12.6 examples/libs/l4re/c++/shared_ds/ds_srv.cc . . . . .	915
12.7 examples/libs/l4re/c++/shared_ds/shared_ds.lua . . . . .	918
12.8 examples/libs/l4re/c/ma+rm.c . . . . .	918
12.9 examples/libs/l4re/streammap/client.cc . . . . .	920
12.10examples/libs/l4re/streammap/server.cc . . . . .	921
12.11examples/libs/l4re/streammap/streammap.cfg . . . . .	922
12.12examples/libs/libirq/async_isr.c . . . . .	923
12.13examples/libs/libirq/loop.c . . . . .	924

---

12.14examples/libs/shmc/prodcons.c . . . . .	924
12.15examples/sys/alien/main.c . . . . .	927
12.16examples/sys/ipc/ipc.cfg . . . . .	929
12.17examples/sys/ipc/ipc_example.c . . . . .	929
12.18examples/sys/isr/main.c . . . . .	931
12.19examples/sys/migrate/thread_migrate.cc . . . . .	932
12.20examples/sys/migrate/thread_migrate.cfg . . . . .	934
12.21examples/sys/singlestep/main.c . . . . .	935
12.22examples/sys/start-with-exc/main.c . . . . .	937
12.23examples/sys/utcb-ipc/main.c . . . . .	939
12.24examples/sys/ux-vhw/main.c . . . . .	941
12.25hello/server/src/main.c . . . . .	942
12.26tmpfs/lib/src/fs.cc . . . . .	942



# Chapter 1

## Fiasco.OC & L4 Runtime Environment (L4Re)

### 1.1 Preface

The intention of this document is to provide a birds eye overview about [L4Re](#) and about the environment in which typical applications and servers run. We highlight here the principled functionality of the servers in the environment but do not discuss their specific interfaces. Detailed documentation about these interface is available in the modules section.

The document is meant as a general overview repeating many design concepts of L4-based systems and capability systems in general. We do though assume familiarity with C++ and an idea on the general concepts and terms of [L4](#): threads --- as an abstraction for execution ---, tasks --- holding the capabilities to kernel objects that are accessible by the threads executing in this task ---, and [IPC](#) over [IPC-gates](#) to send messages and to transfer capabilities between tasks.

### 1.2 General System Structure

The system has a multi-tier architecture consisting of the following layers depicted in the figure below:

- **Microkernel** The microkernel is the component at the lowest level of the software stack. It is the only piece of software that is running in the privileged mode of the processor.
- **Tasks** Tasks are the basic containers (address spaces) in which system services and applications are executed. They run in the processor's deprivileged user mode.

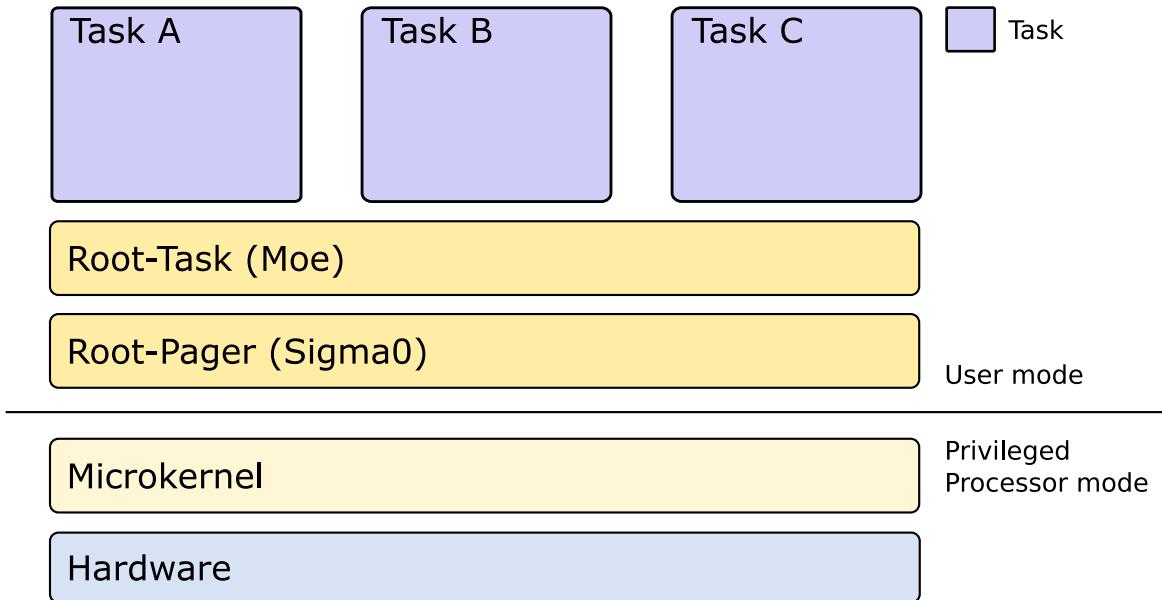


Figure 1.1: Basic Structure of an L4Re based system

In terms of functionality, the system is structured as follows:

- **Microkernel** The kernel provides primitives to execute programs in tasks, to enforce isolation among them, and to provide means of secure communication in order to let them cooperate. As the kernel is the most privileged, security-critical software component in the system, it is a general design goal to make it as small as possible in order to reduce its attack surface. It provides only a minimal set of mechanisms that are necessary to support applications.
- **Runtime Environment** The small kernel offers a concise set of interfaces, but these are not necessarily suited for building applications directly on top of it. The L4 Runtime Environment aims at providing more convenient abstractions for application development. It comprises low-level software components that interface directly with the microkernel. The root pager *sigma0* and the root task *Moe* are the most basic components of the runtime environment. Other services (e.g., for device enumeration) use interfaces provided by them.
- **Applications** Applications run on top of the system and use services provided by the Runtime Environment -- or by other applications. There may be several types of applications in the system and even virtual machine monitors and device drivers are considered applications in the terminology used in this document. They are running alongside other applications on the system.

Lending terminology from the distributed systems area, applications offering services to other applications are usually called *servers*, whereas applications using those services are named *clients*. Being in both roles

is also common, for instance, a file system server may be viewed as a server with respect to clients using the file system, while the server itself may also act as a client of a hard disk driver.

In the following sections, we discuss the basic concepts of our microkernel and its runtime environment in more depth.

## 1.3 The Fiasco.OC Microkernel

The Fiasco.OC microkernel is the lowest-level piece of software running in an L4-based system. The microkernel is the only program that runs in privileged processor mode. It does not include complex services such as program loading, device drivers, or file systems; those are implemented in user-level programs on top of it (a basic set these services and abstractions is provided by the [L4 Runtime Environment](#)).

Fiasco.OC kernel services are implemented in kernel objects. Tasks hold references to kernel objects in their respective "*object space*", which is a kernel-protected table. These references are called *capabilities*. Fiasco system calls are function invocations on kernel objects through the corresponding capabilities. These can be thought of as function invocations on object references in an object-oriented programming environment. Furthermore, if a task owns a capability, it may grant other tasks the same (or fewer) rights on this object by passing the capability from its own to the other task's object space.

From a design perspective, capabilities are a concept that enables flexibility in the system structure. A thread that invokes an object through a capability does not need to care about where this object is implemented. In fact, it is possible to implement all objects either in the kernel or in a user-level server and replace one implementation with the other transparently for clients.

### 1.3.1 Communication

The basic communication mechanism in L4-based systems is called "*Inter Process Communication (IPC)*". It is always synchronous, i.e. both communication partners need to actively rendezvous for IPC. In addition to transmitting arbitrary data between threads, IPC is also used to resolve hardware exceptions, faults and for virtual memory management.

### 1.3.2 Kernel Objects

The following list gives a short overview of the kernel objects provided by the Fiasco.OC microkernel:

- **Task** A task comprises a memory address space (represented by the task's page table), an object space (holding the kernel protected capabilities), and on X86 an IO-port address space.
- **Thread** A thread is bound to a task and executes code. Multiple threads can coexist in one task and are scheduled by the Fiasco scheduler.
- **Factory** A factory is used by applications to create new kernel objects. Access to a factory is required to create any new kernel object. Factories can control and restrict object creation.
- **IPC Gate** An IPC gate is used to create a secure communication channel between different tasks. It embeds a label (kernel protected payload) that securely identifies the gate through which a message is received. The gate label is not visible to and cannot be altered by the sender.
- **IRQ** IRQ objects provide access to hardware interrupts. Additionally, programs can create new virtual interrupt objects and trigger them. This allows to implement a signaling mechanism. The receiver cannot decide whether the interrupt is a physical or virtual one.

- **Vcon** Provides access to the in-kernel debugging console (input and output). There is only one such object in the kernel and it is only available, if the kernel is built with debugging enabled. This object is typically interposed through a user-level service or without debugging in the kernel can be completely based on user-level services.
- **Scheduler** Implements scheduling policy and assignment of threads to CPUs, including CPU statistics.

## 1.4 L4 Runtime Environment (L4Re)

The [L4 Runtime Environment \(L4Re\)](#) provides a basic set of services and abstractions, which are useful to implement and run user-level applications on top of the Fiasco.OC microkernel.

[L4Re](#) consists of a set of libraries and servers. Libraries as well as server interfaces are completely object oriented. They implement prototype implementations for the classes defined by the [L4Re](#) specification.

A minimal L4Re-based application needs 3 components to be booted beforehand: the Fiasco microkernel, the root pager (*Sigma0*), and the root task (*Moe*). The *Sigma0* root pager initially owns all system resources, but is usually used only to resolve page faults for the *Moe* root task. *Moe* provides the essential services to normal user applications such as an initial program loader, a region-map service for virtual memory management, and a memory (data space) allocator.

## 1.5 Introduction to L4Re's concepts

This section introduces basic concepts used by [L4Re](#). Understanding of these concepts is a fundamental requirement to understand the inner workings of L4Re's software components and can dramatically help developers in efficiently developing L4Re-based software.

## 1.6 Memory management - Data Spaces and the Region Map

### 1.6.1 User-level paging

Memory management in L4-based systems is done by user-level applications, the role is usually called *pager*. Tasks can give other tasks full or restricted access rights to parts of their own memory. The kernel offers means to grant the memory in a secure way, often referred to as *memory mapping*.

The described mechanism can be used to construct a memory hierarchy among tasks. The root of the hierarchy is *sigma0*, which initially gets all system resources and hands them out once on a first-come-first-served basis. Memory resources can be mapped between tasks at a page-size granularity. This size is predetermined by the CPU's memory management unit and is commonly set to 4 kB.

### 1.6.2 Data spaces

A data space is the [L4Re](#) abstraction for objects which may be accessed in a memory mapped fashion (i.e., using normal memory read and write instructions). Examples include the sections of a binary which the loader attaches to the application's address space, files in the ROM or on disk provided by a file server, the registers of memory-mapped devices and anonymous memory such as the heap or the stack.

Anonymous memory data spaces in particular (but in general all data spaces except memory mapped IO) can either be constructed entirely from a portion of the RAM or the current working set may be multiplexed

on some portion of the RAM. In the first case it is possible to eagerly insert all pages (more precisely page-frame capabilities) into the application's address space such that no further page faults occur when this data space is accessed. In general, however, only the pages for the some portion are provided and further pages are inserted by the pager as a result of page faults.

### 1.6.3 Virtual Memory Handling

The virtual memory of each task is constructed from data spaces, backing virtual memory regions (VMRs). The management of the VMRs is provided by an object called *region map*. A dedicated region-map object is associated with each task, it allows to attach and detach data spaces to an address space as well as to reserve areas of virtual memory. Since the region-map object possesses all knowledge about virtual memory layout of a task, it also serves as an application's default pager.

### 1.6.4 Memory Allocation

Operating systems commonly use anonymous memory for implementing dynamic memory allocation (e.g., using *malloc* or *new*). In an L4Re-based system, each task gets assigned a memory allocator providing anonymous memory using data spaces.

#### See also

[Data-Space API](#) and [Region map API](#).

## 1.7 Capabilities and Naming

The [L4Re](#) system is a capability based system which uses and offers capabilities to implement fine-grained access control.

Generally, owning a capability means to be allowed to communicate with the object the capability points to. All user-visible kernel objects, such as tasks, threads, and IRQs, can be accessed only through a capability. Please refer to the [Kernel Objects](#) documentation for details. Capabilities are stored in per-task capability tables (the object space) and are referenced by capability selectors or object flex pages. In a simplified view, a capability selector is a natural number indexing into the capability table of the current task.

As a matter of fact, a system designed solely based on capabilities, uses so-called 'local names', because each task can only access those objects made available to this task. Other objects are not visible to and accessible by the task.

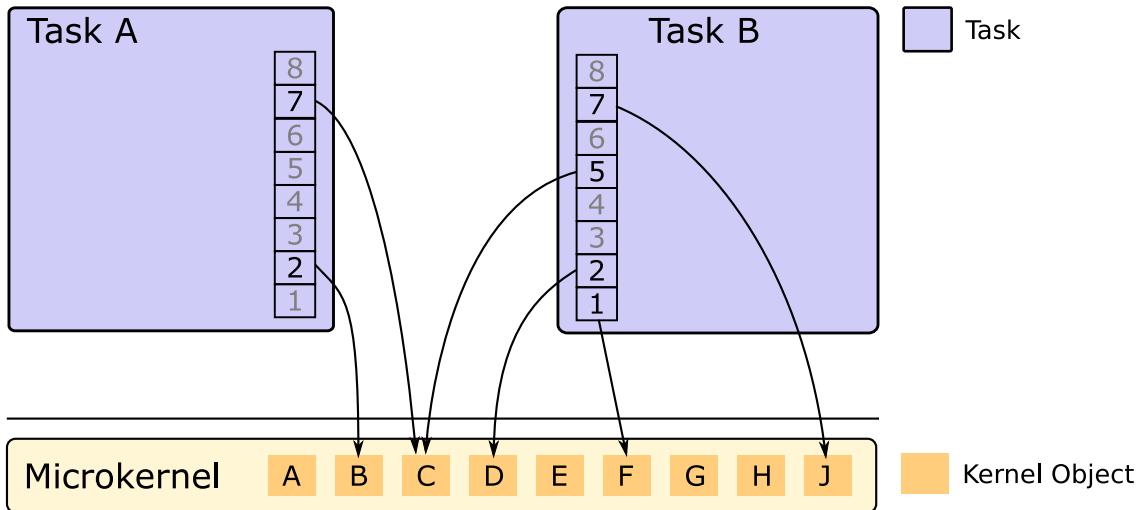


Figure 1.2: Capabilities and Local Naming in L4

So how does an application get access to service? In general all applications are started with an initial set of objects available. This set of objects is predetermined by the creator of a new application process and granted directly to into the new task before starting the first application thread. The application can then use these initial objects to request access to further objects or to transfer capabilities to own objects to other applications. A central [L4Re](#) object for exchanging capabilities at runtime is the name-space object, implementing a store of named capabilities.

From a security perspective, the set of initial capabilities (access rights to objects) completely define the execution environment of an application. Mandatory security policies can be defined by well known properties of the initial objects and carefully handled access rights to them.

## 1.8 Initial Environment and Application Bootstrapping

New applications that are started by a loader conforming to [L4Re](#) get provided an [initial environment](#). This environment comprises a set of capabilities to initial [L4Re](#) objects that are required to bootstrap and run this application. These capabilities include:

- A capability to an initial memory allocator for obtaining memory in the form of data spaces
- A capability to a factory which can be used to create additional kernel objects
- A capability to a Vcon object for debugging output and maybe input

- A set of named capabilities to application specific objects

During the bootstrapping of the application, the loader establishes data spaces for each individual region in the ELF binary. These include data spaces for the code and data sections, and a data space backed with RAM for the stack of the program's first thread.

One loader implementation is the *Moe* root task. Moe usually starts an *init* process that is responsible for coordinating the further boot process. The default *init* process is *Ned*, which implements a script-based configuration and startup of other processes. Ned uses Lua (<http://www.lua.org>) as its scripting language, see Ned Script example for more details.

### 1.8.1 Configuring an application before startup

The default L4Re init process (Ned) provides a Lua script based configuration of initial capabilities and application startup. Ned itself also has a set of initial objects available that can be used to create the environment for an application. The most important object is a kernel object factory that allows creation of kernel objects such as IPC gates (communication channels), tasks, threads, etc. Ned uses Lua tables (associative arrays) to represent sets of capabilities that shall be granted to application processes.

```
local caps = {
    name = some_capability
}
```

The 'L4' Lua package in Ned also has support functions to create application tasks, region-map objects, etc. to start an ELF binary in a new task. The package also contains Lua bindings for basic L4Re objects, for example, to generic factory objects, which are used to create kernel objects and also user-level objects provided by user-level servers.

```
L4.default_loader:start({ caps = { some_service = service } }, "rom/program --a
rg");
```

### 1.8.2 Connecting clients and servers

In general, a connection between a client and a server is represented by a communication channel (IPC gate). That is available to the client and the server. You can see the simplest connection between a client and a server in the following example.

```
local loader = L4.default_loader; -- which is Moe
local svc = loader:new_channel(); -- create an IPC gate
loader:start({ caps = { service = svc:full() } }, "rom/my_server");
loader:start({ caps = { service = svc:m("rw") } }, "rom/my_client");
```

As you can see in the snippet, the first action is to create a new channel (IPC gate) using `loader:new_channel()`. The capability to the gate is stored in the variable `svc`. Then the binary `my_server` is started in a new task, and full (`:full()`) access to the IPC gate is granted to the server as initial object. The gate is accessible to the server application as "service" in the set of its initial capabilities. Virtually in parallel a second task, running the client application, is started and also given access to the IPC gate with less rights (`:m("rw")`), note, this is essential). The server can now receive messages via the IPC gate and provide some service and the client can call operations on the IPC gate to communicate with the server.

Services that keep client specific state need to implement per-client server objects. Usually it is the responsibility of some authority (e.g., Ned) to request such an object from the service via a generic factory object that the service provides initially.

```

local loader = L4.default_loader; -- which is Moe
local svc = loader:new_channel():m("rws"); -- create an IPC gate with rws right
loader:start({ caps = { service = svc:full() } }, "rom/my-service");
loader:start({ caps = { foo_service = svc:create(object_to_create, "param") } },
             "rom/client");

```

This example is quite similar to the first one, however, the difference is that Ned itself calls the `create` method on the factory object provided by the server and passes the returned capability of that request as "`foo_service`" to the client process.

#### Note

The `svc:create(..)` call blocks on the server. This means the script execution blocks until the `my-service` application handles the `create` request.

## 1.9 Program Input and Output

The initial environment provides a Vcon capability used as the standard input/output stream. Output is usually connected to the parent of the program and displayed as debugging output. The standard output is also used as a back end to the C-style `printf` functions and the C++ streams.

Vcon services are implemented in Moe and the loader as well as by the Fiasco kernel and connected either to the serial line or to the screen if available.

#### See also

[Virtual Console](#)

## 1.10 Initial Memory Allocator and Factory

The purpose of the memory allocator and of the factory is to provide the application with the means to allocate memory (in the form of data spaces) and kernel objects respectively. An initial memory allocator and an initial factory are accessible via the allocation [L4Re](#) environment.

#### See also

[Memory allocator API](#)

The factory is a kernel object that provides the ability to create new kernel objects dynamically. A factory imposes a resource limit for kernel memory, and is thus a means to prevent denial of service attacks on kernel resources. A factory can also be used to create new factory objects.

#### See also

[Factory](#)

## 1.11 Application and Server Building Blocks

So far we have discussed the environment of applications in which a single thread runs and which may invoke services provided through their initial objects. In the following we describe some building blocks to extend the application in various dimensions and to eventually implement a server which implements user-level objects that may in turn be accessed by other applications and servers.

### 1.11.1 Creating Additional Application Threads

To create application threads, one must allocate a stack on which this thread may execute, create a thread kernel object and setup the information required at startup time (instruction pointer, stack pointer, etc.). In L4Re this functionality is encapsulated in the pthread library.

### 1.11.2 Providing a Service

In capability systems, services are typically provided by transferring a capability to those applications that are authorised to access the object to which the capability refers to.

Let us discuss an example to illustrate how two parties can communicate with each other: Assume a simple file server, which implements an interface for accessing individual files: `read(pos, buf, length)` and `write(pos, data, length)`.

L4Re provides support for building servers based on the class [L4::Server\\_object](#). [L4::Server\\_object](#) provides an abstract interface to be used with the [L4::Server](#) class. Specific server objects such as, in our case, files inherit from [L4::Server\\_object](#). Let us call this class `File_object`. When invoked upon receiving a message, the [L4::Server](#) will automatically identify the corresponding server object based on the capability that has been provided to its clients and invoke this object's *dispatch* function with the incoming message as a parameter. Based on this message, the server must then decide which of the protocols it implements was invoked (if any). Usually, it will evaluate a protocol specific opcode that clients are required to transmit as one of the first words in the message. For example, assume our server assigns the following opcodes: Read = 0 and Write = 1. The *dispatch* function calls the corresponding server function (i.e., `File_object::read()` or `File_object::write()`), which will in turn parse additional parameters given to the function. In our case, this would be the position and the amount of data to be read or written. In case the write function was called the server will now update the contents of the file with the data supplied. In case of a read it will store the requested part of the file in the message buffer. A reply to the client finishes the client request.



# Chapter 2

## Getting Started

Here you can find the first steps to boot a very simple setup.

The setup consists of the following components:

- Fiasco.OC --- Microkernel
- Sigma0 --- Root Pager
- Moe --- Root Task
- Ned --- Init Process
- hello --- Hello World Application

The guide assumes that you already compiled the base components and describes how to generate an ISO image, with GRUB 1 or GRUB 2 as a boot loader, that can for example be booted within QEMU.

First you need a `modules.list` file that contains an entry for the scenario.

```
modaddr 0x002000000

entry hello
  kernel fiasco -serial_esc
  roottask moe rom/hello.cfg
  module l4re
  module ned
  module hello.cfg
  module hello
```

This file describes all the binaries and scripts to put into the ISO image, and also describes the GRUB menu.lst contents. What you need to do is to set the make variable `MODULE_SEARCH_PATH` to contain the path to your Fiasco.OC build directory and the directory containing your `hello.cfg` script.

The `hello.cfg` script should look like the following. A ready to use version can be found in `l4/conf/examples`.

```
require("L4");
L4.default_loader:start({}, "rom/hello");
```

The first line of this script ensures that the `L4` package is available for the script. The second line uses the default loader object defined in that package and starts the binary `rom/hello`.

**Note**

All modules defined in `modules.list` are available as data spaces ([L4Re::Dataspace](#)) and registered in a name space ([L4Re::Namespace](#)). This name space is in turn available as 'rom' to the init process ([Ned](#)).

Now you can go to your [L4Re](#) build directory and run the following command.

**Note**

The example assumes that you have created the `modules.list` and `hello.cfg` files in the `/tmp` directory. Adapt if you created them somewhere else.

```
make grubiso E=hello MODULES_LIST=/tmp/modules.list MODULE_SEARCH_PATH=/tmp:<path_to_fiasco_builddir>
```

Or as an alternative use GRUB 2:

```
make grub2iso E=hello MODULES_LIST=/tmp/modules.list MODULE_SEARCH_PATH=/tmp:<path_to_fiasco_builddir>
```

Now you should be able to boot the image in QEMU by running:

```
qemu -cdrom images/hello.iso -serial stdio
```

If you press `<ESC>` in the terminal that shows you the serial output you enter the Fiasco.OC kernel debugger... Have fun.

**Customizations**

A basic set of bootable entries can be found in `l4/conf/modules.list`. This file is the default for any image creation as shown above. It is recommended that local modification regarding image creation are done in `conf/Makeconf.boot`. Initially you may copy `Makeconf.boot.example` to `Makeconf.boot`. You can overwrite `MODULES_LIST` to set your own modules-list file. Set `MODULE_SEARCH_PATH` to your setup according to the examples given in the file. When configured a make call is reduced to:

```
make grub2iso E=hello
```

All other local configuration can be done in a `Makeconf.local` file located in the `l4` directory.

# Chapter 3

## L4Re Servers

Here you shall find a tight overview over the standard services running on Fiasco.OC and L4Re.

### 3.1 Sigma0, the Root Pager

Sigma0 is a special server running on L4 because it is responsible of resolving page faults for the root task, the first useful task on L4Re. Sigma0 can be seen as part of the kernel, however it runs in unprivileged mode. To run something useful on Fiasco.OC you usually need to run Sigma0, nevertheless it is possible to replace Sigma0 by a different implementation.

### 3.2 Moe, the Root Task

Moe is our implementation of the L4 root task that is responsible for bootstrapping the system, and to provide basic resource management services to the applications on top. Therefore Moe provides L4Re resource management and multiplexing services:

- **Memory** in the form of memory allocators ([L4Re::Mem\\_alloc](#), [L4::Factory](#)) and data spaces ([L4Re::Dataspace](#))
- **Cpu** in the form of basic scheduler objects ([L4::Scheduler](#))
- **Vcon** multiplexing for debug output (output only)
- **Virtual memory management** for applications, [L4Re::Rm](#)

Moe further provides an implementation of L4Re name spaces ([L4Re::Namespace](#)), which are for example used to provide a read only directory of all multi-boot modules. In the case of a boot loader, like grub that enables a VESA frame buffer, there is also a single instance of an L4Re graphics session ([L4Re::Goos](#)).

To start the system Moe starts a single ELF program, this init process. The init process (usually Ned, see the next section) gets access to all resources managed by Moe and to the Sigma0 root pager interface.

For more details see [Moe, the Root-Task](#).

### 3.3 Ned, the Default Init Process

To keep the root task free from complicated scripting engines and to avoid circular dependencies in application startup (that could lead to dead locks) the configuration and startup of the real system is managed by an extra task, the init process.

Ned is such an init process that allows system configuration via Lua scripts.

For more information see [Ned](#).

### 3.4 Io, the Platform and Device Resource Manager

Because all peripheral management of Fiasco.OC is done in user-level applications, there is the need to have a centralized management of the resources belonging to the platform and to peripheral devices.

This is the job of Io. Io provides portable abstractions for iterating and accessing devices and their resources (IRQ's, IO Memory...), as well as delegating access to those resources to other applications (e.g., device drivers).

For more details see [Io, the Io Server](#).

### 3.5 Mag, the GUI Multiplexer

Our default multiplexer for the graphics hardware is Mag. Mag is a Nitpicker (TODO: ref) derivate that allows secure multiplexing of the graphics and input hardware among multiple applications and multiple complete windowing environments.

### 3.6 fb-drv, the Low-Level Graphics Driver

The fb-drv server provides low-level access and initialization of various graphics hardware. It has support for running VESA BIOS calls on Intel x86 platforms, as well as support for various ARM display controllers. *fb-drv*, provides a single instance of the L4Re::Goos interface and can serve as a back end for the Mag server, in particular, if there is no graphics support in the boot loader.

### 3.7 Rtc, the Real-Time Clock Server

Rtc is a simple multiplexer for real-time clock hardware on your platform.

### 3.8 Moe, the Root-Task

Moe is the default Root-Task implementation for L4Re-based systems.

*Moe* is the first task which is usually started in L4Re-based systems. The micro kernel starts *Moe* as the Root-Task.

Moe provides default implementation for the basic L4Re abstractions, such as data spaces ([L4Re::Dataspace](#)), region maps ([L4Re::Rm](#)), memory allocators ([L4Re::Mem\\_alloc](#), [L4::Factory](#)), name spaces ([L4Re::Namespace](#)) and so on (see [L4Re Interface](#)).

Moe consists of the following subsystems:

- [Name-Space Provider \(L4Re::Namespace\)](#) --- provides instances of name spaces
- [Boot FS](#) --- provides access to the files loaded during platform boot (e.g., linked into the boot image or loaded via GRUB boot loader)
- [Log Subsystem \(L4Re::Log\)](#) --- provides tagged log output for applications
- [l4re\\_moe\\_scheduler \(L4::Scheduler\)](#) --- provides simple scheduler objects for scheduling policy enforcement
- [Memory Allocator, Generic Factory \(L4Re::Mem\\_alloc, L4::Factory\)](#) --- provides allocation of physical RAM as data spaces, as well as allocation of the other [L4Re](#) objects provided by Moe

### 3.8.1 Memory Allocator, Generic Factory

The generic factory in Moe is responsible for all kinds of dynamic object allocation. The interface is a combination of [L4::Factory](#) and, for traditional reasons, [L4Re::Mem\\_alloc](#). The generic factory interface allows allocation of the following objects:

- [L4Re::Namespace](#)
- [L4Re::Dataspace](#), RAM allocation
- [L4Re::Rm](#), Virtual memory management for application tasks
- [L4::Vcon](#) (output only)
- [L4::Scheduler](#), to provide a restricted priority / CPU range for clients
- [L4::Factory](#), to provide a quota limited allocation for clients

The memory allocator in Moe is the alternative interface for allocating memory (RAM) in terms of [L4Re::Dataspace](#)-s (

#### See also

[L4Re::Mem\\_alloc](#)). The granularity for memory allocation is the machine page size ([L4\\_PAGESIZE](#)).

The provided data spaces can have different characteristics:

- Physically contiguous and pre allocated
- Non contiguous and on-demand allocated with possible copy on write (COW)

### 3.8.2 Name-Space Provider

Moe provides a name spaces conforming to the [L4Re::Namespace](#) interface (see [Name-space API](#)). Per default Moe creates a single name space for the [Boot FS](#). That is available as `rom` in the initial objects of the init process.

### 3.8.3 Boot FS

The Boot FS subsystem provides read only access to the files loaded during the platform boot (or available in ROM). These files are either linked into the boot image or loaded via a flexible boot loader, such as GRUB.

The subsystem provides an `L4Re::Namespace` object as directory and an `L4Re::Dataspace` object for each file.

### 3.8.4 Log Subsystem

The logging facility of Moe provides per application tagged and synchronized log output.

### 3.8.5 Command-Line Options

Moe command-line syntax is:

```
moe [--debug=<flags>] [--init=<binary>] [--l4re-dbg=<flags>] [--ldr-flags=<flags>] [-- <init options>]
```

#### 3.8.5.1 --debug=<debug flags>

This option enables debug messages from Moe itself, the <debug flags> values are a combination of info, warn, boot, server, loader, and ns (or all for full verbosity).

#### 3.8.5.2 --init=<init process>

This option allows to override the default init process binary, which is 'rom/ned'.

#### Note

command-line options to the init process are given after the -- special option.

#### 3.8.5.3 --l4re-dbg=<debug flags>

This option allows to set the debug options for the `L4Re` runtime environment of the init process. The flags are the same as for `--debug=`.

#### 3.8.5.4 --ldr-flags=<loader flags>

This option allows setting some loader options for the `L4Re` runtime environment. The flags are pre\_alloc, all\_segs\_cow, and pinned\_segs.

## 3.9 Ned, the Init Process

Ned's job is to bootstrap the system running on `L4Re`.

The main thing to do here is to coordinate the startup of services and applications as well as to provide the communication channels for them. The central facility in Ned is the Lua (<http://www.lua.org>) script interpreter with the `L4Re` and ELF-loader bindings.

The boot process is based on the execution of one or more Lua scripts that create communication channels (IPC gates), instantiate other [L4Re](#) objects, organize capabilities to these objects in sets, and start application processes with access to those objects (or based on those objects).

For starting applications, Ned depends on the services of [Moe, the Root-Task](#) or another *loader*, which must provide data spaces and region maps. Ned also uses the 'rom' capability as source for Lua scripts and at least the 'l4re' binary (the runtime environment core) running in each application.

Each application Ned starts is equipped with an [L4Re::Env](#) environment that provides information about all the initial objects made accessible to this application.

### 3.9.1 Lua Bindings for L4Re

Ned provides various bindings for [L4Re](#) abstractions. These bindings are located in the 'L4' package (`require "L4"`).

#### 3.9.1.1 Capabilities in Lua

Capabilities are handled as normal values in Lua. They can be stored in normal variables or Lua compound structures (tables). A capability in Lua possesses additional information about the access rights that shall be transferred to other tasks when the capability is transferred. To support implementation of the Principle of Least Privilege, minimal rights are assigned by default. Extended rights can be added using the method `mode("...")` (short `m("...")`) that returns a new reference to the capability with the given rights.

##### Note

It is generally impossible to elevate the real access rights to an object. This means that if Ned has only restricted rights to an object it is not possible to upgrade the access rights with the `mode` method.

The capabilities in Lua also carry dynamic type information about the referenced objects. They thereby provide type-specific operations on the objects, such as the `create` operation on a generic factory or the `query` and `register` operations on a name space.

#### 3.9.1.2 Access to L4Re::Env Capabilities

The initial objects provided to Ned itself are accessible via the table `L4.Env`. The default (usually unnamed) capabilities are accessible as `factory`, `log`, `mem_alloc`, `parent`, `rm`, and `scheduler` in the `L4.Env` table.

#### 3.9.1.3 Constants

##### Protocols

The protocol constants are defined by default in the `L4` package's table `L4.Proto`. The definition is not complete and only covers what is usually needed to configure and start applications. The protocols are for example used as first argument to the `Factory:create` method.

```
Proto = {
    Namespace = 0x4001,
    Goos      = 0x4003,
    Mem_alloc = 0x4004,
    Rm        = 0x4005,
    Irq       = -1,
    Sigma0    = -6,
```

```

Log      = -13,
Scheduler = -14,
Factory   = -15,
Ipc_gate  = 0,
}

```

## Debugging Flags

Debugging flags used for the applications L4Re core:

```

Dbg = {
    Info      = 1,
    Warn      = 2,
    Boot      = 4,
    Server    = 0x10,
    Exceptions = 0x20,
    Cmd_line  = 0x40,
    Loader    = 0x80,
    Name_space = 0x400,
    All       = 0xffffffff,
}

```

## Loader Flags

Flags for configuring the loading process of an application.

```

Ldr_flags = {
    eager_map    = 0x1, -- L4RE_AUX_LDR_FLAG_EAGER_MAP
    all_segs_cow = 0x2, -- L4RE_AUX_LDR_FLAG_ALL_SEGS_COW
    pinned_segs  = 0x4, -- L4RE_AUX_LDR_FLAG_PINNED_SEGS
}

```

### 3.9.1.4 Application Startup Details

The central facility for starting a new task with Ned is the class `L4.Loader`. This class provides interfaces for conveniently configuring and starting programs. It provides three operations:

- `new_channel()` Returns a new IPC gate that can be used to connect two applications
- `start()` and `startv()` Start a new application process and return a process object

The `new_channel()` call is used to provide a service application with a communication channel to bind its initial service to. The concrete behavior of the object and the number of IPC gates required by a server depends on the server implementation. The channel can be passed to client applications as well or can be used for operations within the script itself.

`start()` and `startv()` always require at least two arguments. The first one is a table that contains information about the initial objects an application shall get. The second argument is a string, which for `start()` is the program name plus a white-space-separated list of program arguments (`argv`). For `startv()` the second argument is just the program binary name -- which may contain spaces --, and the program arguments are provided as separate string arguments following the binary name (allowing spaces in arguments, too). The last optional argument is a table containing the POSIX environment variables for the program.

The Loader class uses reasonable defaults for most of the initial objects. However, you can override any initial object with some user-defined values. The main elements of the initial object table are:

- `factory` The factory used by the new process to create new kernel objects, such as threads etc. This must be a capability to an object implementing the [L4::Factory](#) protocol and defaults to the factory object provided to Ned.
- `mem` The memory allocator provided to the application and used by Ned allocates data spaces for the process. This defaults to Ned's memory allocator object (see [L4Re::Mem\\_alloc](#)).
- `rm_fab` The generic factory object used to allocate the region-map obejct for the process. (defaults to Ned's memory allocator).
- `log_fab` The generic factory to create the [L4Re::Log](#) object for the application's output (defaults to Ned's memory allocator). The `create` method of the `log_fab` object is called with `log_tag` and `log_color`, from this table, as arguments.
- `log_tag` The string used for tagging log output of this process (defaults to the program name) (see `log_fab`).
- `log_color` The color used for the log tag (defaults to "white").
- `scheduler` The scheduler object used for the process' threads (defaults to Ned's own scheduler).
- `caps` The table with application-specific named capabilities (default is an empty table). If the table does not contain a capability with the name 'rom', the 'rom' capability from Ned's initial caps is inserted into the table.

## 3.10 Io, the Io Server

The Io server handles all platform devices and resources such as I/O memory, ports (on x86) and interrupts, and grants access to those to clients.

Upon startup Io discovers all platform devices using available means on the system, e.g. on x86 the PCI bus is scanned and the ACPI subsystem initialised. Available I/O resource can also be configured statically.

Each Io server client is provided with its own virtual bus which it can iterate to find devices. A virtual PCI bus may be a part of this virtual bus.

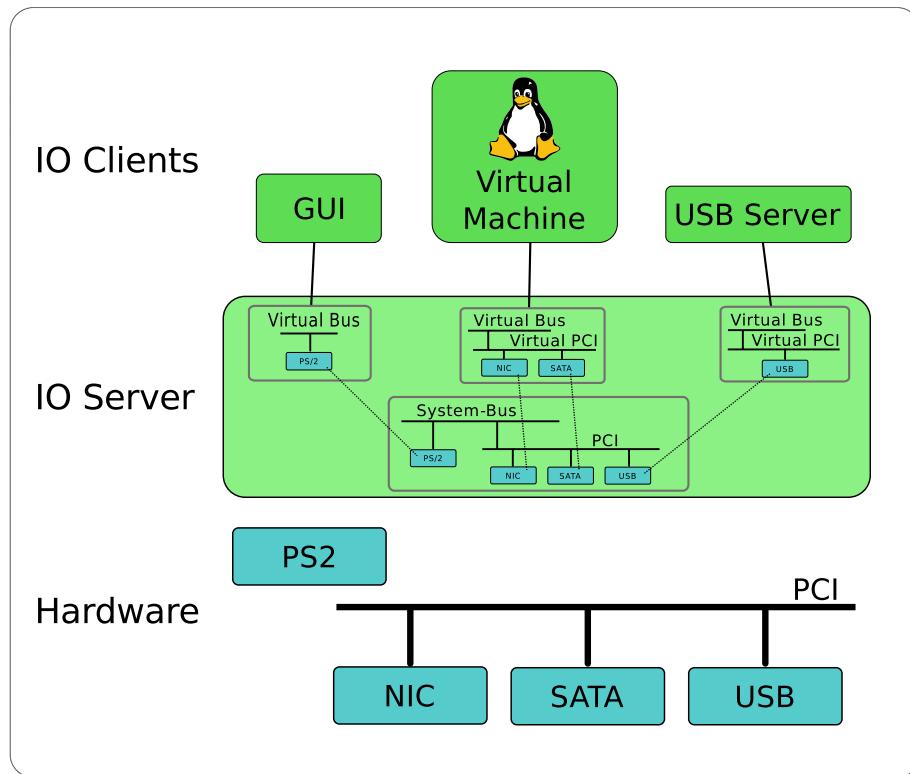


Figure 3.1: IO Service Architecture Overview

The Io server must be configured to create virtual buses for its clients. This is done with at least one configuration file specifying static resources as well as virtual buses for clients. The configuration may be split across several configuration files passed to Io through the command line.

The platform configuration is stored in the structure called `hw-root`. It lists devices that are available on the platform. For the x86 architecture a basic set of platform devices is defined in the file `x86-legacy.devs`. There are configuration files for various ARM platforms available, as well. If the system contains a PCI bus, it is scanned automatically and the devices found on it are added automatically to the pool of available devices.

To allow clients access to available devices, a virtual system bus needs to be created that lists the devices that should be available to that client. These buses have a name that Io uses to register a `vbus` object in its name space.

A very simple configuration for Io could look like this:

```

# Example configuration for io

# Configure 2 platform device to be known to io
hw-root
{
    FOODEVICE => new Device()
    {
        .hid = "FOODEVICE";
        new-res Irc(17);
        new-res Mmio(0x6f000000 .. 0x6f007fff);
    }

    BARDEVICE => new Device()
}

```

```
{  
    .hid = "BARDEVICE";  
    new-res Irq(19);  
    new-res Irq(20);  
    new-res Mmio(0x6f100000 .. 0x6f100fff);  
}  
}  
  
# Create a virtual bus for a client and give access to FOODEVICE  
client1 => new System_bus()  
{  
    dev => wrap(hw-root.FOODEVICE);  
}  
  
# Create a virtual bus for another client and give it access to  
# BARDEVICE  
client2 => new System_bus()  
{  
    dev => wrap(hw-root.BARDEVICE);  
}
```

Assigning clients PCI devices could look like this:

```
# This is a configuration snippet for PCI device selection  
  
pciclient => new System_bus()  
{  
    pci_storage[] => wrap(hw-root.match("PCI/CC_01"));  
    pci_net[] => wrap(hw-root.match("PCI/CC_02"));  
    pci_mm[] => wrap(hw-root.match("PCI/CC_04"))  
}
```

The CC numbers are PCI class codes. You can also use REV\_, VEN\_, DEV\_ and SUBSYS\_ to specify revision, vendor, device and subsystem with a hex number.



# Chapter 4

## Pthread Support

L4Re supports the standard pthread library functionality.

Therefore L4Re itself does not contain any documentation for pthreads itself. Please refer to the standard pthread documentation instead.

The L4Re specific parts will be described herein.

- Include pthread-l4.h header `file`:

```
#include <pthread-l4.h>
```

- Return the local thread capability of a pthread thread:

Use `pthread_getl4cap(pthread_t *t)` to get the capability index of the pthread t.

For example:

```
pthread_getl4cap(pthread_self());
```

- Setting the L4 priority of an L4 thread works with a special scheduling policy (other policies do not affect the L4 thread priority):

```
pthread_t t;
pthread_attr_t a;
struct sched_param sp;

pthread_attr_init(&a);
sp.sched_priority = l4_priority;
pthread_attr_setschedpolicy(&a, SCHED_L4);
pthread_attr_setschedparam(&a, &sp);
pthread_attr_setinheritsched(&a, PTHREAD_EXPLICIT_SCHED);

if (pthread_create(&t, &a, pthread_func, NULL))
    // failure...

pthread_attr_destroy(&a);
```



# Chapter 5

## Module Index

### 5.1 Modules

Here is a list of all modules:

Base API . . . . .	103
Basic Macros . . . . .	110
Cache Consistency . . . . .	152
Capabilities . . . . .	255
Error codes . . . . .	167
Fiasco extensions . . . . .	113
Fiasco real time scheduling extensions . . . . .	121
Kernel Debugger . . . . .	159
Flex pages . . . . .	130
Integer Types . . . . .	394
Kernel Interface Page . . . . .	208
Fiasco-UX Virtual devices . . . . .	277
Memory descriptors (C version) . . . . .	211
Kernel Objects . . . . .	204
Factory . . . . .	169
IPC-Gate API . . . . .	107
IRQs . . . . .	198
Interrupt controller . . . . .	175
Scheduler . . . . .	214
Task . . . . .	219
Thread . . . . .	226
Thread control . . . . .	239
vCPU API . . . . .	275
Virtual Console . . . . .	269
Virtual Machines . . . . .	174
VM API for SVM . . . . .	149
VM API for TZ . . . . .	280
VM API for VMX . . . . .	150
Memory operations. . . . .	278
Memory related . . . . .	154
Object Invocation . . . . .	182
Error Handling . . . . .	193
Message Items . . . . .	140

Message Tag . . . . .	245
Realtime API . . . . .	198
Timeouts . . . . .	142
Virtual Registers (UTCBs) . . . . .	261
ARM Virtual Registers (UTCB) . . . . .	280
Buffer Registers (BRs) . . . . .	265
Message Registers (MRs) . . . . .	265
Exception registers . . . . .	267
Thread Control Registers (TCRs) . . . . .	266
amd64 Virtual Registers (UTCB) . . . . .	282
x86 Virtual Registers (UTCB) . . . . .	282
IRQ handling library . . . . .	355
Interface for asynchronous ISR handlers. . . . .	359
Interface for asynchronous ISR handlers with a given IRQ capability. . . . .	362
Interface using direct functionality. . . . .	355
Interface using direct functionality. . . . .	360
L4Re C Interface . . . . .	100
Capability allocator . . . . .	73
Dataspace interface . . . . .	50
Debug interface . . . . .	52
Event interface . . . . .	53
Kumem allocator utility . . . . .	74
L4Re Util C Interface . . . . .	102
Log interface . . . . .	54
Memory allocator . . . . .	57
Namespace interface . . . . .	61
Region map interface . . . . .	62
Video API . . . . .	75
L4Re C++ Interface . . . . .	46
Auxiliary data . . . . .	92
C++ Exceptions . . . . .	43
Console API . . . . .	81
Data-Space API . . . . .	82
Debugging API . . . . .	83
Event API . . . . .	91
Goos video API . . . . .	100
Initial Environment . . . . .	86
L4Re ELF Auxiliary Information . . . . .	83
L4Re Protocol identifiers . . . . .	95
L4Re Util C++ Interface . . . . .	49
Kumem utilities . . . . .	99
L4Re Capability API . . . . .	97
Logging interface . . . . .	92
Memory allocator API . . . . .	93
Name-space API . . . . .	94
Parent API . . . . .	94
Region map API . . . . .	96
Shared Memory Library . . . . .	377
Chunks . . . . .	380
Consumer . . . . .	385
Producer . . . . .	383
Signals . . . . .	388

Consumer . . . . .	391
Producer . . . . .	391
Sigma0 API . . . . .	363
Internal constants . . . . .	367
Small C++ Template Lib . . . . .	44
Utility Functions . . . . .	340
Atomic Instructions . . . . .	294
Bit Manipulation . . . . .	301
Bitmap graphics and fonts . . . . .	347
Functions for rendering bitmap data in frame buffers . . . . .	348
Functions for rendering bitmap fonts to frame buffers . . . . .	351
CPU related functions . . . . .	283
Comfortable Command Line Parsing . . . . .	334
ELF binary format . . . . .	307
Functions to manipulate the local IDT . . . . .	286
IA32 Port I/O API . . . . .	344
Internal functions . . . . .	300
Kernel Interface Page API . . . . .	332
Low-Level Thread Functions . . . . .	338
Machine Restarting Function . . . . .	338
Priority related functions . . . . .	336
Random number support . . . . .	337
Timestamp Counter . . . . .	286
vCPU Support Library . . . . .	369
Extended vCPU support . . . . .	376



# Chapter 6

## Namespace Index

### 6.1 Namespace List

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

<code>cxx</code> (Various kinds of C++ utilities ) . . . . .	399
<code>cxx::Bits</code> (Internal helpers for the cxx package ) . . . . .	401
<code>L4</code> ( <code>L4</code> low-level kernel interface ) . . . . .	401
<code>L4::Ipc_svr</code> (Helper classes for <code>L4::Server</code> instantiation ) . . . . .	405
<code>L4Re</code> ( <code>L4</code> Runtime Environment ) . . . . .	406
<code>L4Re::Vfs</code> (Virtual file system for interfaces POSIX libc ) . . . . .	407



# Chapter 7

## Data Structure Index

### 7.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

L4::Alloc_list . . . . .	409
L4::Thread::Attr . . . . .	409
L4Re::Util::Auto_cap< T > . . . . .	413
L4Re::Util::Auto_del_cap< T > . . . . .	414
cxx::Auto_ptr< T > . . . . .	416
cxx::Avl_set< Item, Compare, Alloc > . . . . .	426
cxx::Avl_set< Pair< Key, Data >, Pair_first_compare< Compare< Key >, Pair< Key, Data > >, Alloc > . . . . .	426
cxx::Avl_map< Key, Data, Compare, Alloc > . . . . .	419
cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc > . . . . .	446
cxx::Base_slab< sizeof(Type), Slab_size, Max_free, Alloc > . . . . .	446
cxx::Slab< Type, Slab_size, Max_free, Alloc > . . . . .	839
cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc > . . . . .	449
cxx::Base_slab_static< sizeof(Type), Slab_size, Max_free, Alloc > . . . . .	449
cxx::Slab_static< Type, Slab_size, Max_free, Alloc > . . . . .	843
L4::Basic_registry . . . . .	454
cxx::Bitmap_base . . . . .	464
cxx::Bitmap< BITS > . . . . .	461
cxx::Bits::Bst< Node, Get_key, Compare > . . . . .	472
cxx::Avl_tree< Node, Get_key, Compare > . . . . .	435
cxx::Bits::Bst< _Node, Get_key, Pair_first_compare< Compare< Key >, Pair< Key, Data > > >	472
cxx::Avl_tree< _Node, Get_key, Pair_first_compare< Compare< Key >, Pair< Key, Data > >> . . . . .	435
cxx::Bits::Bst_node . . . . .	482
cxx::Avl_tree_node . . . . .	441
L4::Ipc::Buf_cp_in< T > . . . . .	485
L4::Ipc::Buf_cp_out< T > . . . . .	486
L4::Ipc::Buf_in< T > . . . . .	487
L4Re::Cap_alloc . . . . .	491
L4Re::Util::Cap_alloc_base . . . . .	494
L4::Cap_base . . . . .	496

L4::Cap< T > . . . . .	488
L4::Smart_cap< T, SMART > . . . . .	846
cxx::Bitmap_base::Char< BITS > . . . . .	504
L4Re::Video::Color_component . . . . .	504
L4::Ipc_svr::Compound_reply . . . . .	510
L4::Ipc_svr::Default_loop_hooks . . . . .	535
L4Re::Util::Counting_cap_alloc< COUNTERTYPE > . . . . .	513
L4Re::Util::Dataspace_svr . . . . .	524
L4::Ipc_svr::Default_setup_wait . . . . .	536
L4::Ipc_svr::Default_loop_hooks . . . . .	535
L4::Ipc_svr::Default_timeout . . . . .	536
L4::Ipc_svr::Default_loop_hooks . . . . .	535
cxx::Bits::Direction . . . . .	537
L4Re::Vfs::Directory . . . . .	539
L4Re::Vfs::File . . . . .	599
L4Re::Vfs::Be_file . . . . .	454
Elf32_Dyn . . . . .	550
Elf32_Ehdr . . . . .	550
Elf32_Phdr . . . . .	552
Elf32_Shdr . . . . .	553
Elf32_Sym . . . . .	554
Elf64_Dyn . . . . .	554
Elf64_Ehdr . . . . .	555
Elf64_Phdr . . . . .	557
Elf64_Shdr . . . . .	557
Elf64_Sym . . . . .	558
L4Re::Env . . . . .	559
L4Re::Event_buffer_t< PAYLOAD >::Event . . . . .	569
L4Re::Event_buffer_t< PAYLOAD > . . . . .	577
L4Re::Util::Event_buffer_t< PAYLOAD > . . . . .	581
L4Re::Util::Event_buffer_consumer_t< PAYLOAD > . . . . .	573
L4Re::Util::Event_t< PAYLOAD > . . . . .	585
L4::Exception_tracer . . . . .	588
L4::Base_exception . . . . .	444
L4::Invalid_capability . . . . .	638
L4::Runtime_error . . . . .	823
L4::Bounds_error . . . . .	469
L4::Com_error . . . . .	507
L4::Element_already_exists . . . . .	544
L4::Element_not_found . . . . .	547
L4::Out_of_memory . . . . .	792
L4::Unknown_error . . . . .	879
L4Re::Vfs::File_system . . . . .	601
L4Re::Vfs::Be_file_system . . . . .	458
L4Re::Vfs::Fs . . . . .	605
L4Re::Vfs::Ops . . . . .	785
L4Re::Vfs::Generic_file . . . . .	608
L4Re::Vfs::File . . . . .	599
gfxbitmap_offset . . . . .	612
L4Re::Util::Video::Goos_svr . . . . .	618

L4::Ipc_svr::Ignore_errors . . . . .	630
L4::Ipc_svr::Default_loop_hooks . . . . .	535
L4Re::Video::Goos::Info . . . . .	632
L4Re::Video::View::Info . . . . .	635
L4::IOModifier . . . . .	641
L4::Ipc::Istream . . . . .	659
L4::Ipc::Iostream . . . . .	642
L4Re::Util::Item_alloc_base . . . . .	668
cxx::List< D, Alloc >::Iter . . . . .	670
cxx::List_item::Iter . . . . .	672
cxx::List_item::T_iter< T, Poly > . . . . .	857
L4::Kobject . . . . .	675
L4::Kobject_t< Dataspace, L4::Kobject, L4Re::Protocol::Dataspace > . . . . .	682
L4Re::Dataspace . . . . .	514
L4::Kobject_t< Debug_obj, L4::Kobject, Protocol::Debug > . . . . .	682
L4Re::Debug_obj . . . . .	528
L4::Kobject_t< Debugger, Kobject, L4_PROTO_DEBUGGER > . . . . .	682
L4::Debugger . . . . .	530
L4::Kobject_t< Factory, Kobject, L4_PROTO_FACTORY > . . . . .	682
L4::Factory . . . . .	589
L4::Kobject_t< Goos, L4::Kobject, L4Re::Protocol::Goos > . . . . .	682
L4Re::Video::Goos . . . . .	612
L4::Kobject_t< Ipc_gate, Kobject, L4_PROTO_KOBJECT > . . . . .	682
L4::Ipc_gate . . . . .	649
L4::Kobject_t< Irq_eio, Kobject, L4_PROTO_IRQ > . . . . .	682
L4::Kobject_t< Mem_alloc, L4::Kobject, L4Re::Protocol::Mem_alloc > . . . . .	682
L4Re::Mem_alloc . . . . .	760
L4::Kobject_t< Meta, Kobject, L4_PROTO_META > . . . . .	682
L4::Meta . . . . .	769
L4::Kobject_t< Namespace, L4::Kobject, L4Re::Protocol::Namespace > . . . . .	682
L4Re::Namespace . . . . .	778
L4::Kobject_t< Parent, L4::Kobject, L4Re::Protocol::Parent > . . . . .	682
L4Re::Parent . . . . .	797
L4::Kobject_t< Rm, L4::Kobject, L4Re::Protocol::Rm > . . . . .	682
L4Re::Rm . . . . .	812
L4::Kobject_t< Scheduler, Kobject, L4_PROTO_SCHEDULER > . . . . .	682
L4::Scheduler . . . . .	829
L4::Kobject_t< Task, Kobject, L4_PROTO_TASK > . . . . .	682
L4::Task . . . . .	860
L4::Kobject_t< Vm, Task, L4_PROTO_VM > . . . . .	682
L4::Vm . . . . .	904
L4::Kobject_t< Thread, Kobject, L4_PROTO_THREAD > . . . . .	682
L4::Thread . . . . .	869
L4::Kobject_t< Vm, Kobject, L4_PROTO_VM > . . . . .	682
L4::Vm . . . . .	904
L4::Kobject_2t< Derived, Base1, Base2, PROTO > . . . . .	680
L4::Kobject_2t< Console, Video::Goos, Event > . . . . .	680
L4Re::Console . . . . .	511
L4::Kobject_t< Derived, Base, PROTO > . . . . .	682
L4::Kobject_t< Icu, Irq_eio, L4_PROTO_IRQ > . . . . .	682

L4::Icu . . . . .	622
L4::Kobject_t< Event, L4::Icu, L4Re::Protocol::Event > . . . . .	682
L4Re::Event . . . . .	570
L4::Kobject_t< Vcon, Icu, L4_PROTO_LOG > . . . . .	682
L4::Vcon . . . . .	882
L4::Kobject_t< Log, L4::Vcon, L4_PROTO_LOG > . . . . .	682
L4Re::Log . . . . .	756
L4::Kobject_t< Irq, Irq_eio, L4_PROTO_IRQ > . . . . .	682
L4::Irq . . . . .	651
14_buf_regs_t . . . . .	683
14_exc_regs_t . . . . .	684
14_fpage_t . . . . .	686
14_icu_info_t . . . . .	687
L4::Icu::Info . . . . .	631
14_kernel_info_mem_desc_t . . . . .	688
14_kernel_info_t . . . . .	689
14_msg_regs_t . . . . .	692
14_mshtag_t . . . . .	693
14_rt_preemption_t . . . . .	695
14_rt_preemption_val32_t . . . . .	696
14_rt_preemption_val_t . . . . .	696
14_sched_cpu_set_t . . . . .	697
14_sched_param_t . . . . .	697
14_snd_fpage_t . . . . .	699
14_thread_REGS_t . . . . .	700
14_timeout_s . . . . .	701
14_timeout_t . . . . .	701
14_tracebuffer_status_t . . . . .	703
14_tracebuffer_status_window_t . . . . .	708
14_vcon_attr_t . . . . .	708
14_vcpu_ipc_regs_t . . . . .	709
14_vcpu_REGS_t . . . . .	711
14_vcpu_state_t . . . . .	714
L4vcpu::Vcpu . . . . .	890
14_vhw_descriptor . . . . .	717
14_vhw_entry . . . . .	720
14_vm_state . . . . .	722
14_vm_svm_vmcb_control_area . . . . .	722
14_vm_svm_vmcb_state_save_area . . . . .	722
14_vm_svm_vmcb_state_save_area_seg . . . . .	724
14_vm_svm_vmcb_t . . . . .	724
14re_aux_t . . . . .	726
14re_ds_stats_t . . . . .	726
14re_elf_aux_mword_t . . . . .	727
14re_elf_aux_t . . . . .	727
14re_elf_aux_vma_t . . . . .	727
14re_env_cap_entry_t . . . . .	728
14re_env_t . . . . .	729
14re_event_t . . . . .	731
14re_video_color_component_t . . . . .	732
14re_video_goops_info_t . . . . .	732
14re_video_pixel_info_t . . . . .	734

l4re_video_view_info_t . . . . .	736
l4re_video_view_t . . . . .	738
l4util_idt_desc_t . . . . .	739
l4util_idt_header_t . . . . .	739
l4util_mb_addr_range_t . . . . .	741
l4util_mb_apm_t . . . . .	741
l4util_mb_drive_t . . . . .	742
l4util_mb_info_t . . . . .	743
l4util_mb_mod_t . . . . .	744
l4util_mb_vbe_ctrl_t . . . . .	745
l4util_mb_vbe_mode_t . . . . .	746
cxx::List< D, Alloc > . . . . .	747
cxx::List_alloc . . . . .	750
cxx::List_item . . . . .	752
L4::Factory::Lstr . . . . .	759
cxx::Lt_functor< Obj > . . . . .	760
L4::Kip::Mem_desc . . . . .	764
L4Re::Vfs::Mman . . . . .	774
L4Re::Vfs::Ops . . . . .	785
L4::Thread::Modify_senders . . . . .	776
L4::Ipc::Msg_ptr< T > . . . . .	777
L4Re::Util::Names::Name . . . . .	778
cxx::New_allocator< _Type > . . . . .	783
L4::Factory::Nil . . . . .	783
cxx::Avl_set< Item, Compare, Alloc >::Node . . . . .	783
cxx::Nothrow . . . . .	784
L4::Ipc::Ostream . . . . .	787
L4::Ipc::Iostream . . . . .	642
cxx::Pair< First, Second > . . . . .	795
cxx::Pair_first_compare< Cmp, Typ > . . . . .	796
L4Re::Video::Pixel_info . . . . .	800
L4Re::Util::Ref_cap< T > . . . . .	807
L4Re::Util::Ref_del_cap< T > . . . . .	807
L4Re::Vfs::Regular_file . . . . .	808
L4Re::Vfs::File . . . . .	599
L4::Factory::S . . . . .	825
L4::Server< LOOP_HOOKS > . . . . .	834
L4::Server_object . . . . .	837
L4::Ipc::Small_buf . . . . .	845
L4Re::Smart_cap_auto< Unmap_flags > . . . . .	849
L4Re::Util::Smart_cap_auto< Unmap_flags > . . . . .	851
L4Re::Util::Smart_count_cap< Unmap_flags > . . . . .	851
L4Re::Vfs::Special_file . . . . .	852
L4Re::Vfs::File . . . . .	599
L4vcpu::State . . . . .	854
L4Re::Dataspace::Stats . . . . .	856
L4::String . . . . .	856
L4::Type_info . . . . .	879
L4Re::Util::Vcon_svr< SVR > . . . . .	889
L4Re::Video::View . . . . .	899
cxx::Bitmap_base::Word< BITS > . . . . .	907



# Chapter 8

## Data Structure Index

### 8.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">L4::Alloc_list</a> (A simple list-based allocator ) . . . . .	409
<a href="#">L4::Thread::Attr</a> ( <a href="#">Thread</a> attributes used for control_commit() ) . . . . .	409
<a href="#">L4Re::Util::Auto_cap&lt; T &gt;</a> (Automatic capability that implements automatic free and unmap of the capability selector ) . . . . .	413
<a href="#">L4Re::Util::Auto_del_cap&lt; T &gt;</a> (Automatic capability that implements automatic free and unmap+delete of the capability selector ) . . . . .	414
<a href="#">cxx::Auto_ptr&lt; T &gt;</a> (Smart pointer with automatic deletion ) . . . . .	416
<a href="#">cxx::Avl_map&lt; Key, Data, Compare, Alloc &gt;</a> (AVL tree based associative container ) . . . . .	419
<a href="#">cxx::Avl_set&lt; Item, Compare, Alloc &gt;</a> (AVL Tree for simple comapreable items ) . . . . .	426
<a href="#">cxx::Avl_tree&lt; Node, Get_key, Compare &gt;</a> (A generic AVL tree ) . . . . .	435
<a href="#">cxx::Avl_tree_node</a> (Node of an AVL tree ) . . . . .	441
<a href="#">L4::Base_exception</a> (Base class for all exceptions, thrown by the <a href="#">L4Re</a> framework ) . . . . .	444
<a href="#">cxx::Base_slab&lt; Obj_size, Slab_size, Max_free, Alloc &gt;</a> (Basic slab allocator ) . . . . .	446
<a href="#">cxx::Base_slab_static&lt; Obj_size, Slab_size, Max_free, Alloc &gt;</a> (Merged slab allocator (allocators for objects of the same size are merged together) ) . . . . .	449
<a href="#">L4::Basic_registry</a> (This registry returns the corresponding server object based on the label ) . . . . .	454
<a href="#">L4Re::Vfs::Be_file</a> (Boiler plate class for implementing an open file for <a href="#">L4Re::Vfs</a> ) . . . . .	454
<a href="#">L4Re::Vfs::Be_file_system</a> (Boilerplate class for implementing a <a href="#">L4Re::Vfs::File_system</a> ) . . . . .	458
<a href="#">cxx::Bitmap&lt; BITS &gt;</a> (A static bit map ) . . . . .	461
<a href="#">cxx::Bitmap_base</a> (Basic bitmap abstraction ) . . . . .	464
<a href="#">L4::Bounds_error</a> (Access out of bounds ) . . . . .	469
<a href="#">cxx::Bits::Bst&lt; Node, Get_key, Compare &gt;</a> (Basic binary search tree (BST) ) . . . . .	472
<a href="#">cxx::Bits::Bst_node</a> (Basic type of a node in a binary search tree (BST) ) . . . . .	482
<a href="#">L4::Ipc::Buf_cp_in&lt; T &gt;</a> (Abstraction for extracting array from an <a href="#">Ipc::Istream</a> ) . . . . .	485
<a href="#">L4::Ipc::Buf_cp_out&lt; T &gt;</a> (Abstraction for inserting an array into an <a href="#">Ipc::Ostream</a> ) . . . . .	486
<a href="#">L4::Ipc::Buf_in&lt; T &gt;</a> (Abstraction to extract an array from an <a href="#">Ipc::Istream</a> ) . . . . .	487
<a href="#">L4::Cap&lt; T &gt;</a> (Capability Selector a la C++) . . . . .	488
<a href="#">L4Re::Cap_alloc</a> (Capability allocator interface ) . . . . .	491
<a href="#">L4Re::Util::Cap_alloc_base</a> (Capability allocator ) . . . . .	494
<a href="#">L4::Cap_base</a> (Base class for all kinds of capabilities ) . . . . .	496
<a href="#">cxx::Bitmap_base::Char&lt; BITS &gt;</a> (Helper abstraction for a byte contained in the bitmap ) . . . . .	504
<a href="#">L4Re::Video::Color_component</a> (A color component ) . . . . .	504
<a href="#">L4::Com_error</a> (Error conditions during IPC ) . . . . .	507

L4::Ipc_svr::Compound_reply (Mix in for LOOP_HOOKS to always use compound reply and wait ) . . . . .	510
L4Re::Console (Console class ) . . . . .	511
L4Re::Util::Counting_cap_alloc< COUNTERTYPE > (Reference-counting cap allocator ) . . . . .	513
L4Re::Dataspace (This class represents a data space ) . . . . .	514
L4Re::Util::Dataspace_svr (Dataspace server class ) . . . . .	524
L4Re::Debug_obj (Debug interface ) . . . . .	528
L4::Debugger (Debugger interface ) . . . . .	530
L4::Ipc_svr::Default_loop_hooks (Default LOOP_HOOKS ) . . . . .	535
L4::Ipc_svr::Default_setup_wait (Mix in for LOOP_HOOKS for setup_wait no op ) . . . . .	536
L4::Ipc_svr::Default_timeout (Mix in for LOOP_HOOKS to use a 0 send and a infinite receive timeout ) . . . . .	536
cxx::Bits::Direction (The direction to go in a binary search tree ) . . . . .	537
L4Re::Vfs::Directory (Interface for a POSIX file that is a directory ) . . . . .	539
L4::Element_already_exists (Exception for duplicate element insertions ) . . . . .	544
L4::Element_not_found (Exception for a failed lookup (element not found) ) . . . . .	547
Elf32_Dyn (ELF32 dynamic entry ) . . . . .	550
Elf32_Ehdr (ELF32 header ) . . . . .	550
Elf32_Phdr (ELF32 program header ) . . . . .	552
Elf32_Shdr (ELF32 section header - figure 1-9, page 1-9 ) . . . . .	553
Elf32_Sym (ELF32 symbol table entry ) . . . . .	554
Elf64_Dyn (ELF64 dynamic entry ) . . . . .	554
Elf64_Ehdr (ELF64 header ) . . . . .	555
Elf64_Phdr (ELF64 program header ) . . . . .	557
Elf64_Shdr (ELF64 section header ) . . . . .	557
Elf64_Sym (ELF64 symbol table entry ) . . . . .	558
L4Re::Env (Initial Environment (C++ version) ) . . . . .	559
L4Re::Event_buffer_t< PAYLOAD >::Event (Event structure used in buffer ) . . . . .	569
L4Re::Event (Event class ) . . . . .	570
L4Re::Util::Event_buffer_consumer_t< PAYLOAD > (An event buffer consumer ) . . . . .	573
L4Re::Event_buffer_t< PAYLOAD > (Event buffer class ) . . . . .	577
L4Re::Util::Event_buffer_t< PAYLOAD > (Event_buffer utility class ) . . . . .	581
L4Re::Util::Event_t< PAYLOAD > (Convenience wrapper for getting access to an event object ) . . . . .	585
L4::Exception_tracer (Back-trace support for exceptions ) . . . . .	588
L4::Factory (C++ L4 Factory, to create all kinds of kernel objects ) . . . . .	589
L4Re::Vfs::File (The basic interface for an open POSIX file ) . . . . .	599
L4Re::Vfs::File_system (Basic interface for an L4Re::Vfs file system ) . . . . .	601
L4Re::Vfs::Fs (POSIX File-system related functionality ) . . . . .	605
L4Re::Vfs::Generic_file (The common interface for an open POSIX file ) . . . . .	608
gfxbitmap_offset (Offsets in pmap[] and bmap[] ) . . . . .	612
L4Re::Video::Goos (A goos ) . . . . .	612
L4Re::Util::Video::Goos_svr (Goos server class ) . . . . .	618
L4::Icu (C++ version of an interrupt controller ) . . . . .	622
L4::Ipc_svr::Ignore_errors (Mix in for LOOP_HOOKS to ignore IPC errors ) . . . . .	630
L4::Icu::Info (Info for an ICU ) . . . . .	631
L4Re::Video::Goos::Info (Information structure of a goos ) . . . . .	632
L4Re::Video::View::Info (Information structure of a view ) . . . . .	635
L4::Invalid_capability (Indicates that an invalid object was invoked) . . . . .	638
L4::IOModifier (Modifier class for the IO stream ) . . . . .	641
L4::Ipc::Iostream (Input/Output stream for IPC [un]marshalling ) . . . . .	642
L4::Ipc_gate (L4 IPC gate ) . . . . .	649
L4::Irq (C++ version of an L4 IRQ ) . . . . .	651
L4::Ipc::Istream (Input stream for IPC unmarshalling ) . . . . .	659
L4Re::Util::Item_alloc_base (Item allocator ) . . . . .	668

cxx::List< D, Alloc >::Iter (Iterator ) . . . . .	670
cxx::List_item::Iter (Iterator for a list of ListItem-s ) . . . . .	672
L4::Kobject (Base class for all kinds of kernel objects, referred to by capabilities ) . . . . .	675
L4::Kobject_2t< Derived, Base1, Base2, PROTO > (Helper class to create an L4Re interface class that is derived from two base classes ) . . . . .	680
L4::Kobject_t< Derived, Base, PROTO > (Helper class to create an L4Re interface class that is derived from a single base class ) . . . . .	682
l4_buf_regs_t (Encapsulation of the buffer-registers block in the UTCB ) . . . . .	683
l4_exc_regs_t (UTCB structure for exceptions ) . . . . .	684
l4_fpage_t (L4 flexpage type ) . . . . .	686
l4_icu_info_t (Info structure for an ICU ) . . . . .	687
l4_kernel_info_mem_desc_t (Memory descriptor data structure ) . . . . .	688
l4_kernel_info_t (L4 Kernel Interface Page ) . . . . .	689
l4_msg_regs_t (Encapsulation of the message-register block in the UTCB ) . . . . .	692
l4_mshtag_t (Message tag data structure ) . . . . .	693
l4_rt_preemption_t (Struct ) . . . . .	695
l4_rt_preemption_val32_t (Struct ) . . . . .	696
l4_rt_preemption_val_t (Struct ) . . . . .	696
l4_sched_cpu_set_t (CPU sets ) . . . . .	697
l4_sched_param_t (Scheduler parameter set ) . . . . .	697
l4_snd_fpage_t (Send-flex-page types ) . . . . .	699
l4_thread_REGS_t (Encapsulation of the thread-control-register block of the UTCB ) . . . . .	700
l4_timeout_s (Basic timeout specification ) . . . . .	701
l4_timeout_t (Timeout pair ) . . . . .	701
l4_tracebuffer_status_t (Trace buffer status ) . . . . .	703
l4_tracebuffer_status_window_t (Trace-buffer status window descriptor ) . . . . .	708
l4_vcon_attr_t (Vcon attribute structure ) . . . . .	708
l4_vcpu_ipc_regs_t (VCPU message registers ) . . . . .	709
l4_vcpu_regs_t (VCPU registers ) . . . . .	711
l4_vcpu_state_t (State of a vCPU ) . . . . .	714
l4_vhw_descriptor (Virtual hardware devices description ) . . . . .	717
l4_vhw_entry (Description of a device ) . . . . .	720
l4_vm_state (State structure for TrustZone VMs ) . . . . .	722
l4_vm_svm_vmcb_control_area (VMCB structure for SVM VMs ) . . . . .	722
l4_vm_svm_vmcb_state_save_area (State save area structure for SVM VMs ) . . . . .	722
l4_vm_svm_vmcb_state_save_area_seg (State save area segment selector struct ) . . . . .	724
l4_vm_svm_vmcb_t (Control structure for SVM VMs ) . . . . .	724
l4re_aux_t (Auxiliary descriptor ) . . . . .	726
l4re_ds_stats_t (Information about the data space ) . . . . .	726
l4re_elf_aux_mword_t (Auxiliary vector element for a single unsigned data word ) . . . . .	727
l4re_elf_aux_t (Generic header for each auxiliary vector element ) . . . . .	727
l4re_elf_aux_vma_t (Auxiliary vector element for a reserved virtual memory area ) . . . . .	727
l4re_env_cap_entry_t (Entry in the L4Re environment array for the named initial objects ) . . . . .	728
l4re_env_t (Initial Environment structure (C version) ) . . . . .	729
l4re_event_t (Event structure used in buffer ) . . . . .	731
l4re_video_color_component_t (Color component structure ) . . . . .	732
l4re_video_goops_info_t (Goops information structure ) . . . . .	732
l4re_video_pixel_info_t (Pixel_info structure ) . . . . .	734
l4re_video_view_info_t (View information structure ) . . . . .	736
l4re_video_view_t (C representation of a goops view ) . . . . .	738
l4util_idt_desc_t (IDT entry ) . . . . .	739
l4util_idt_header_t (Header of an IDT table ) . . . . .	739

<a href="#">l4util_mb_addr_range_t</a> (INT-15, AX=E820 style "AddressRangeDescriptor" ...with a "size" parameter on the front which is the structure size - 4, pointing to the next one, up until the full buffer length of the memory map has been reached ) . . . . .	741
<a href="#">l4util_mb_apm_t</a> (APM BIOS info ) . . . . .	741
<a href="#">l4util_mb_drive_t</a> (Drive Info structure ) . . . . .	742
<a href="#">l4util_mb_info_t</a> . . . . .	743
<a href="#">l4util_mb_mod_t</a> . . . . .	744
<a href="#">l4util_mb_vbe_ctrl_t</a> (VBE controller information ) . . . . .	745
<a href="#">l4util_mb_vbe_mode_t</a> (VBE mode information ) . . . . .	746
<a href="#">cxx::List&lt; D, Alloc &gt;</a> (Doubly linked list, with internal allocation ) . . . . .	747
<a href="#">cxx::List_alloc</a> (Standard list-based allocator ) . . . . .	750
<a href="#">cxx::List_item</a> (Basic list item ) . . . . .	752
<a href="#">L4Re::Log</a> (Log interface class ) . . . . .	756
<a href="#">L4::Factory::Lstr</a> (Special type to add a pascal string into the factory create stream ) . . . . .	759
<a href="#">cxx::Lt_functor&lt; Obj &gt;</a> (Generic comparator class that defaults to the less-than operator ) . . . . .	760
<a href="#">L4Re::Mem_alloc</a> (Memory allocator ) . . . . .	760
<a href="#">L4::Kip::Mem_desc</a> (Memory descriptors stored in the kernel interface page ) . . . . .	764
<a href="#">L4::Meta</a> (Meta interface that shall be implemented by each <a href="#">L4Re</a> object and gives access to the dynamic type information for <a href="#">L4Re</a> objects ) . . . . .	769
<a href="#">L4Re::Vfs::Mman</a> (Interface for the POSIX memory management ) . . . . .	774
<a href="#">L4::Thread::Modify_senders</a> (Wrapper class for modifying senders ) . . . . .	776
<a href="#">L4::Ipc::Msg_ptr&lt; T &gt;</a> (Pointer to an element of type T in an <a href="#">Ipc::Istream</a> ) . . . . .	777
<a href="#">L4Re::Util::Names::Name</a> ( <a href="#">Name</a> class ) . . . . .	778
<a href="#">L4Re::Namespace</a> (Name-space interface ) . . . . .	778
<a href="#">cxx::New_allocator&lt; _Type &gt;</a> (Standard allocator based on operator new () ) . . . . .	783
<a href="#">L4::Factory::Nil</a> (Special type to add a void argument into the factory create stream ) . . . . .	783
<a href="#">cxx::Avl_set&lt; Item, Compare, Alloc &gt;::Node</a> (A smart pointer to a tree item ) . . . . .	783
<a href="#">cxx::Nothrow</a> (Helper type to distinguish the oeprator new version that does not throw exceptions ) . . . . .	784
<a href="#">L4Re::Vfs::Ops</a> (Interface for the POSIX backends for an application ) . . . . .	785
<a href="#">L4::Ipc::Ostream</a> (Output stream for IPC marshalling ) . . . . .	787
<a href="#">L4::Out_of_memory</a> (Exception signalling insufficient memory ) . . . . .	792
<a href="#">cxx::Pair&lt; First, Second &gt;</a> ( <a href="#">Pair</a> of two values ) . . . . .	795
<a href="#">cxx::Pair_first_compare&lt; Cmp, Typ &gt;</a> (Comparison functor for <a href="#">Pair</a> ) . . . . .	796
<a href="#">L4Re::Parent</a> ( <a href="#">Parent</a> interface ) . . . . .	797
<a href="#">L4Re::Video::Pixel_info</a> (Pixel information ) . . . . .	800
<a href="#">L4Re::Util::Ref_cap&lt; T &gt;</a> (Automatic capability that implements automatic free and unmap of the capability selector ) . . . . .	807
<a href="#">L4Re::Util::Ref_del_cap&lt; T &gt;</a> (Automatic capability that implements automatic free and unmap+delete of the capability selector ) . . . . .	807
<a href="#">L4Re::Vfs::Regular_file</a> (Interface for a POSIX file that provides regular file semantics ) . . . . .	808
<a href="#">L4Re::Rm</a> (Region map ) . . . . .	812
<a href="#">L4::Runtime_error</a> (Exception for an abstract runtime error ) . . . . .	823
<a href="#">L4::Factory::S</a> (Stream class for the <a href="#">create()</a> argument stream ) . . . . .	825
<a href="#">L4::Scheduler</a> (Scheduler object ) . . . . .	829
<a href="#">L4::Server&lt; LOOP_HOOKS &gt;</a> (Basic server loop for handling client requests ) . . . . .	834
<a href="#">L4::Server_object</a> (Abstract server object to be used with the L4::Registry_dispatcher ) . . . . .	837
<a href="#">cxx::Slab&lt; Type, Slab_size, Max_free, Alloc &gt;</a> ( <a href="#">Slab</a> allocator for object of type <a href="#">Type</a> ) . . . . .	839
<a href="#">cxx::Slab_static&lt; Type, Slab_size, Max_free, Alloc &gt;</a> (Merged slab allocator (allocators for objects of the same size are merged together) ) . . . . .	843
<a href="#">L4::Ipc::Small_buf</a> (A receive item for receiving a single capability ) . . . . .	845
<a href="#">L4::Smart_cap&lt; T, SMART &gt;</a> (Smart capability class ) . . . . .	846
<a href="#">L4Re::Smart_cap_auto&lt; Unmap_flags &gt;</a> (Helper for Auto_cap and Auto_del_cap ) . . . . .	849
<a href="#">L4Re::Util::Smart_cap_auto&lt; Unmap_flags &gt;</a> (Helper for <a href="#">Auto_cap</a> and <a href="#">Auto_del_cap</a> ) . . . . .	851

L4Re::Util::Smart_count_cap< Unmap_flags > (Helper for <a href="#">Ref_cap</a> and <a href="#">Ref_del_cap</a> ) . . . . .	851
L4Re::Vfs::Special_file (Interface for a POSIX file that provides special file semantics) . . . . .	852
L4vcpu::State (C++ implementation of state word in the vCPU area) . . . . .	854
L4Re::Dataspace::Stats (Information about the data space) . . . . .	856
L4::String (A null-terminated string container class) . . . . .	856
cxx::List_item::T_iter< T, Poly > (Iterator for derived classes from ListItem) . . . . .	857
L4::Task (An L4 Task) . . . . .	860
L4::Thread (L4 kernel thread) . . . . .	869
L4::Type_info (Dynamic Type Information for <a href="#">L4Re</a> Interfaces) . . . . .	879
L4::Unknown_error (Exception for an unknown condition) . . . . .	879
L4::Vcon (C++ L4 Vcon) . . . . .	882
L4Re::Util::Vcon_svr< SVR > (Console server template class) . . . . .	889
L4vcpu::Vcpu (C++ implementation of the vCPU save state area) . . . . .	890
L4Re::Video::View ( <a href="#">View</a> ) . . . . .	899
L4::Vm (Virtual machine) . . . . .	904
cxx::Bitmap_base::Word< BITS > (Helper abstraction for a word contained in the bitmap) . . . . .	907

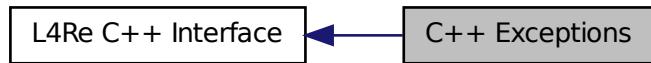


# Chapter 9

## Module Documentation

### 9.1 C++ Exceptions

Collaboration diagram for C++ Exceptions:



### Data Structures

- class [L4::Exception\\_tracer](#)  
*Back-trace support for exceptions.*
- class [L4::Base\\_exception](#)  
*Base class for all exceptions, thrown by the [L4Re](#) framework.*
- class [L4::Runtime\\_error](#)  
*Exception for an abstract runtime error.*
- class [L4::Out\\_of\\_memory](#)  
*Exception signalling insufficient memory.*
- class [L4::Element\\_already\\_exists](#)  
*Exception for duplicate element insertions.*
- class [L4::Unknown\\_error](#)  
*Exception for an unknown condition.*

- class [L4::Element\\_not\\_found](#)  
*Exception for a failed lookup (element not found).*
- class [L4::Invalid\\_capability](#)  
*Indicates that an invalid object was invoked.*
- class [L4::Com\\_error](#)  
*Error conditions during IPC.*
- class [L4::Bounds\\_error](#)  
*Access out of bounds.*

## Files

- file [exceptions](#)  
*Base exceptions.*
- file [std\\_exc\\_io](#)  
*Base exceptions std stream operator.*

## 9.2 Small C++ Template Lib

### Data Structures

- class [L4::Alloc\\_list](#)  
*A simple list-based allocator.*
- class [cxx::Auto\\_ptr< T >](#)  
*Smart pointer with automatic deletion.*
- class [cxx::Avl\\_map< Key, Data, Compare, Alloc >](#)  
*AVL tree based associative container.*
- class [cxx::Avl\\_set< Item, Compare, Alloc >](#)  
*AVL Tree for simple comapreable items.*
- class [cxx::Bitmap\\_base](#)  
*Basic bitmap abstraction.*
- class [cxx::Bitmap< BITS >](#)  
*A static bit map.*
- class [cxx::List\\_item](#)  
*Basic list item.*
- struct [cxx::Pair< First, Second >](#)

*Pair of two values.*

- class `cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >`  
*Basic slab allocator.*
- class `cxx::Slab< Type, Slab_size, Max_free, Alloc >`  
*Slab allocator for object of type Type.*
- class `cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >`  
*Merged slab allocator (allocators for objects of the same size are merged together).*
- class `cxx::Slab_static< Type, Slab_size, Max_free, Alloc >`  
*Merged slab allocator (allocators for objects of the same size are merged together).*
- class `cxx::Nothrow`  
*Helper type to distinguish the operator new version that does not throw exceptions.*
- class `cxx::New_allocator< _Type >`  
*Standard allocator based on operator new () .*
- class `L4::String`  
*A null-terminated string container class.*

## Namespaces

- namespace `cxx`  
*Various kinds of C++ utilities.*

## Functions

- template<typename T1 >  
`T1 cxx::min (T1 a, T1 b)`  
*Get the minimum of a and b.*
- template<typename T1 >  
`T1 cxx::max (T1 a, T1 b)`  
*Get the maximum of a and b.*
- `void * operator new (size_t, void *mem, cxx::Nothrow const &) throw ()`  
*Simple placement new operator.*
- `void * operator new (size_t, cxx::Nothrow const &) throw ()`  
*New operator that does not throw exceptions.*

## 9.2.1 Function Documentation

### 9.2.1.1 template<typename T1> T1 cxx::min ( T1 a, T1 b ) [inline]

Get the minimum of *a* and *b*.

#### Parameters

- a* the first value.
- b* the second value.

#### Examples:

[tmpfs/lib/src/fs.cc](#).

Definition at line 35 of file [minmax](#).

### 9.2.1.2 template<typename T1> T1 cxx::max ( T1 a, T1 b ) [inline]

Get the maximum of *a* and *b*.

#### Parameters

- a* the first value.
- b* the second value.

Definition at line 45 of file [minmax](#).

### 9.2.1.3 void\* operator new ( size\_t, void \* mem, cxx::nothrow const & ) throw () [inline]

Simple placement new operator.

#### Parameters

- mem* the address of the memory block to place the new object.

#### Returns

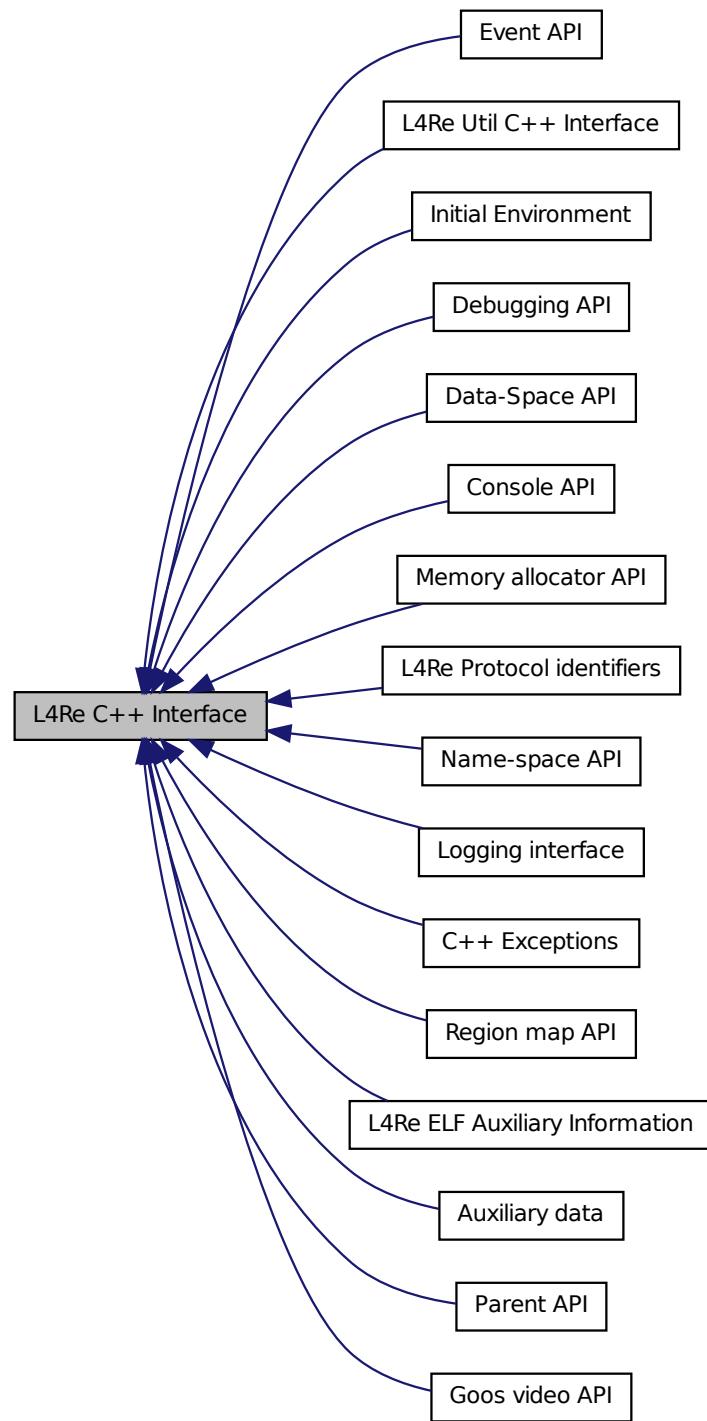
the address given by *mem*.

Definition at line 39 of file [std\\_alloc](#).

## 9.3 L4Re C++ Interface

Documentation of the [L4](#) Runtime Environment C++ API.

Collaboration diagram for L4Re C++ Interface:



## Modules

- C++ Exceptions
- L4Re Util C++ Interface

*Documentation of the L4 Runtime Environment utility functionality in C++.*

- Console API

*Console interface.*

- Data-Space API

*Data-Space API.*

- Debugging API

*Debugging Interface.*

- L4Re ELF Auxiliary Information

*API for embedding auxiliary information into binary programs.*

- Initial Environment

*Environment that is initially provided to an L4 task.*

- Event API

*Event interface.*

- Auxiliary data

- Logging interface

*Interface for log output.*

- Memory allocator API

*Memory-allocator interface.*

- Name-space API

*API for name spaces that store capabilities.*

- Parent API

*Parent interface.*

- L4Re Protocol identifiers

*Basic protocol identifiers used for L4Re.*

- Region map API

*Virtual address-space management.*

- Goos video API

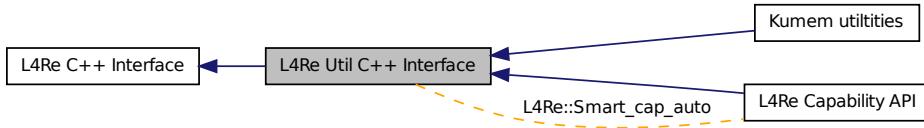
### 9.3.1 Detailed Description

Documentation of the L4 Runtime Environment C++ API.

## 9.4 L4Re Util C++ Interface

Documentation of the L4 Runtime Environment utility functionality in C++.

Collaboration diagram for L4Re Util C++ Interface:



## Data Structures

- class [L4Re::Smart\\_cap\\_auto< Unmap\\_flags >](#)  
*Helper for Auto\_cap and Auto\_del\_cap.*
- class [L4Re::Util::Cap\\_alloc\\_base](#)  
*Capability allocator.*
- class [L4Re::Util::Counting\\_cap\\_alloc< COUNTERTYPE >](#)  
*Reference-counting cap allocator.*
- class [L4Re::Util::Dataspace\\_svr](#)  
*Dataspace server class.*
- class [L4Re::Util::Event\\_buffer\\_t< PAYLOAD >](#)  
*Event\_buffer utility class.*
- class [L4Re::Util::Event\\_buffer\\_consumer\\_t< PAYLOAD >](#)  
*An event buffer consumer.*
- class [L4Re::Util::Vcon\\_svr< SVR >](#)  
*Console server template class.*
- class [L4Re::Util::Video::Goos\\_svr](#)  
*Goos server class.*

## Modules

- [L4Re Capability API](#)
- [Kumem utilities](#)

### 9.4.1 Detailed Description

Documentation of the L4 Runtime Environment utility functionality in C++.

## 9.5 Dataspace interface

Dataspace C interface.

Collaboration diagram for Dataspace interface:



### Data Structures

- struct `l4re_ds_stats_t`  
*Information about the data space.*

### Typedefs

- typedef `l4_cap_idx_t l4re_ds_t`  
*Dataspace type.*
- typedef `l4_cap_idx_t l4re_namespace_t`  
*Dataspace type.*

### Functions

- long `l4re_ds_clear` (const `l4re_ds_t` ds, `l4_addr_t` offset, unsigned long size) L4\_NOTHROW
  - long `l4re_ds_allocate` (const `l4re_ds_t` ds, `l4_addr_t` offset, `l4_size_t` size) L4\_NOTHROW
  - int `l4re_ds_copy_in` (const `l4re_ds_t` ds, `l4_addr_t` dst\_offs, const `l4re_ds_t` src, `l4_addr_t` src\_offs, unsigned long size) L4\_NOTHROW
  - long `l4re_ds_size` (const `l4re_ds_t` ds) L4\_NOTHROW
  - long `l4re_ds_flags` (const `l4re_ds_t` ds) L4\_NOTHROW
  - int `l4re_ds_info` (const `l4re_ds_t` ds, `l4re_ds_stats_t` \*stats) L4\_NOTHROW
  - int `l4re_ds_phys` (const `l4re_ds_t` ds, `l4_addr_t` offset, `l4_addr_t` \*phys\_addr, `l4_size_t` \*phys\_size) L4\_NOTHROW
- Return physical address.*

### 9.5.1 Detailed Description

Dataspace C interface.

### 9.5.2 Function Documentation

#### 9.5.2.1 long l4re\_ds\_clear ( *const l4re\_ds\_t ds, l4\_addr\_t offset, unsigned long size* )

##### Returns

0 on success, <0 on errors

##### See also

[L4Re::Dataspace::clear](#)

#### 9.5.2.2 long l4re\_ds\_allocate ( *const l4re\_ds\_t ds, l4\_addr\_t offset, l4\_size\_t size* )

##### Returns

0 on success, <0 on errors

##### See also

[L4Re::Dataspace::allocate](#)

#### 9.5.2.3 int l4re\_ds\_copy\_in ( *const l4re\_ds\_t ds, l4\_addr\_t dst\_offs, const l4re\_ds\_t src, l4\_addr\_t src\_offs, unsigned long size* )

##### Returns

0 on success, <0 on errors

##### See also

[L4Re::Dataspace::copy\\_in](#)

#### 9.5.2.4 long l4re\_ds\_size ( *const l4re\_ds\_t ds* )

##### Returns

size of dataspace, <0 on errors

##### See also

[L4Re::Dataspace::size](#)

#### 9.5.2.5 long l4re\_ds\_flags ( *const l4re\_ds\_t ds* )

##### See also

[L4Re::Dataspace::flags](#)

### 9.5.2.6 int l4re\_ds\_info ( const l4re\_ds\_t *ds*, l4re\_ds\_stats\_t \* *stats* )

See also

[L4Re::Dataspace::info](#)

### 9.5.2.7 int l4re\_ds\_phys ( const l4re\_ds\_t *ds*, l4\_addr\_t *offset*, l4\_addr\_t \* *phys\_addr*, l4\_size\_t \* *phys\_size* )

Return physical address.

#### Parameters

*ds* Dataspace

*offset* Offset in bytes in dataspace

#### Return values

*phys\_addr* Physical address

*phys\_size* Size of physically contiguous region starting from *phys\_addr* (in bytes).

#### Returns

0 for success, <0 on error

The function returns the physical address of an offset in a dataspace. Use multiple calls of the function to get all physical regions in case of physically non-contiguous dataspaces.

See also

[L4Re::Dataspace::phys](#)

## 9.6 Debug interface

Collaboration diagram for Debug interface:



## Functions

- void [l4re\\_debug\\_obj\\_debug \(l4\\_cap\\_idx\\_t srv, unsigned long function\) L4\\_NOTHROW](#)  
*Call debug function of L4Re service.*

### 9.6.1 Function Documentation

#### 9.6.1.1 void l4re\_debug\_obj\_debug ( l4\_cap\_idx\_t *srv*, unsigned long *function* )

Call debug function of L4Re service.

##### Parameters

*srv* Object to call.

*function* Function to call.

##### See also

[L4Re::Debug\\_obj::debug](#)

## 9.7 Event interface

Event C interface.

Collaboration diagram for Event interface:



## Functions

- long [l4re\\_event\\_get](#) (const l4\_cap\_idx\_t *server*, const l4re\_ds\_t *ds*) L4\_NOTHROW  
*Get an event object.*

### 9.7.1 Detailed Description

Event C interface.

### 9.7.2 Function Documentation

#### 9.7.2.1 long l4re\_event\_get ( const l4\_cap\_idx\_t *server*, const l4re\_ds\_t *ds* )

Get an event object.

##### Parameters

*server* Server to talk to.

*ds* Buffer to event data.

*irq* Event signal.

#### Returns

0 for success, <0 on error

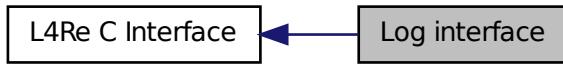
#### See also

[L4Re::Event::get](#)

## 9.8 Log interface

Log C interface.

Collaboration diagram for Log interface:



## Functions

- void [l4re\\_log\\_print](#) (char const \*string) throw ()
   
*Write a null terminated string to the default log.*
- void [l4re\\_log\\_printn](#) (char const \*string, int len) throw ()
   
*Write a string of a given length to the default log.*
- void [l4re\\_log\\_print\\_srv](#) (const [l4\\_cap\\_idx\\_t](#) logcap, char const \*string) throw ()
   
*Write a null terminated string to a log.*
- void [l4re\\_log\\_printn\\_srv](#) (const [l4\\_cap\\_idx\\_t](#) logcap, char const \*string, int len) throw ()
   
*Write a string of a given length to a log.*

### 9.8.1 Detailed Description

Log C interface.

### 9.8.2 Function Documentation

#### 9.8.2.1 void [l4re\\_log\\_print](#) ( char const \* *string* ) throw () [inline]

Write a null terminated string to the default log.

**Parameters**

*string* Text to print, null terminated.

**Returns**

0 for success, <0 on error

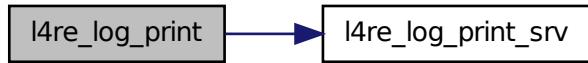
**See also**

[L4Re::Log::print](#)

Definition at line 99 of file [log.h](#).

References [l4re\\_log\\_print\\_srv\(\)](#), and [l4re\\_env\\_t::log](#).

Here is the call graph for this function:



### 9.8.2.2 void l4re\_log\_printn ( char const \* string, int len ) throw () [inline]

Write a string of a given length to the default log.

**Parameters**

*string* Text to print, null terminated.

*len* Length of string in bytes.

**Returns**

0 for success, <0 on error

**See also**

[L4Re::Log::printn](#)

Definition at line 105 of file [log.h](#).

References [l4re\\_log\\_printn\\_srv\(\)](#), and [l4re\\_env\\_t::log](#).

Here is the call graph for this function:



### 9.8.2.3 void l4re\_log\_print\_srv ( const l4\_cap\_idx\_t logcap, char const \* string ) throw ()

Write a null terminated string to a log.

#### Parameters

- logcap* Log capability (service).
- string* Text to print, null terminated.

#### Returns

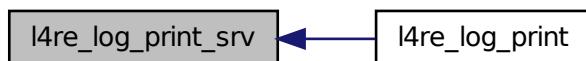
0 for success, <0 on error

#### See also

[L4Re::Log::print](#)

Referenced by [l4re\\_log\\_print\(\)](#).

Here is the caller graph for this function:



### 9.8.2.4 void l4re\_log\_printn\_srv ( const l4\_cap\_idx\_t logcap, char const \* string, int len ) throw (0)

Write a string of a given length to a log.

#### Parameters

- logcap* Log capability (service).

**string** Text to print, null terminated.

**len** Length of string in bytes.

### Returns

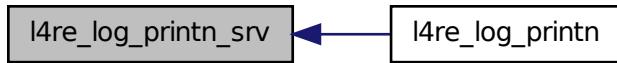
0 for success, <0 on error

### See also

[L4Re::Log::println](#)

Referenced by [l4re\\_log\\_printn\(\)](#).

Here is the caller graph for this function:



## 9.9 Memory allocator

Memory allocator C interface.

Collaboration diagram for Memory allocator:



### Enumerations

- enum [l4re\\_ma\\_flags](#)

*Flags for requesting memory at the memory allocator.*

### Functions

- long [l4re\\_ma\\_alloc](#) (unsigned long size, [l4re\\_ds\\_t](#) const mem, unsigned long flags) throw ()

*Allocate memory.*

- long [l4re\\_ma\\_free](#) ([l4re\\_ds\\_t](#) const mem) throw ()  
*Free memory.*
- long [l4re\\_ma\\_alloc\\_srv](#) ([l4\\_cap\\_idx\\_t](#) srv, unsigned long size, [l4re\\_ds\\_t](#) const mem, unsigned long flags) throw ()  
*Allocate memory.*
- long [l4re\\_ma\\_free\\_srv](#) ([l4\\_cap\\_idx\\_t](#) srv, [l4re\\_ds\\_t](#) const mem) throw ()  
*Free memory.*

### 9.9.1 Detailed Description

Memory allocator C interface.

### 9.9.2 Enumeration Type Documentation

#### 9.9.2.1 enum [l4re\\_ma\\_flags](#)

Flags for requesting memory at the memory allocator.

See also

[L4Re::Mem\\_alloc::Mem\\_alloc\\_flags](#)

Definition at line 42 of file [mem\\_alloc.h](#).

### 9.9.3 Function Documentation

#### 9.9.3.1 long [l4re\\_ma\\_alloc](#) ( unsigned long *size*, [l4re\\_ds\\_t](#) const *mem*, unsigned long *flags* ) throw () [inline]

Allocate memory.

##### Parameters

*size* Size to be requested in bytes (granularity is (super)pages and the size is rounded up to this granularity).

*mem* Capability slot to put the requested dataspace in

*flags* Flags, see [l4re\\_ma\\_flags](#)

##### Returns

0 on success, <0 on error

##### See also

[L4Re::Mem\\_alloc::alloc](#)

The memory allocator returns a dataspace.

#### Note

This function is using the [L4Re::Env::env\(\)->mem\\_alloc\(\)](#) service.

#### Examples:

[examples/libs/l4re/c/ma+rm.c](#).

Definition at line [123](#) of file [mem\\_alloc.h](#).

References [l4re\\_ma\\_alloc\\_srv\(\)](#), and [l4re\\_env\\_t::mem\\_alloc](#).

Here is the call graph for this function:



### 9.9.3.2 long l4re\_ma\_free ( l4re\_ds\_t const mem ) throw () [inline]

Free memory.

#### Parameters

*mem* Dataspace to free.

#### Returns

0 on success, <0 on error

#### See also

[L4Re::Mem\\_alloc::free](#)

#### Note

This function is using the [L4Re::Env::env\(\)->mem\\_alloc\(\)](#) service.

#### Examples:

[examples/libs/l4re/c/ma+rm.c](#).

Definition at line [130](#) of file [mem\\_alloc.h](#).

References [l4re\\_ma\\_free\\_srv\(\)](#), and [l4re\\_env\\_t::mem\\_alloc](#).

Here is the call graph for this function:



### 9.9.3.3 long l4re\_ma\_alloc\_srv ( l4\_cap\_idx\_t *srv*, unsigned long *size*, l4re\_ds\_t const *mem*, unsigned long *flags* ) throw ()

Allocate memory.

#### Parameters

*srv* Memory allocator service.  
*size* Size to be requested.  
*mem* Capability slot to put the requested dataspace in  
*flags* Flags, see [l4re\\_ma\\_flags](#)

#### Returns

0 on success, <0 on error

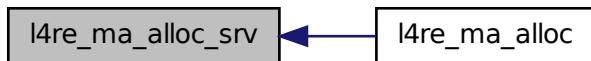
#### See also

[L4Re::Mem\\_alloc::alloc](#)

The memory allocator returns a dataspace.

Referenced by [l4re\\_ma\\_alloc\(\)](#).

Here is the caller graph for this function:



### 9.9.3.4 long l4re\_ma\_free\_srv ( l4\_cap\_idx\_t *srv*, l4re\_ds\_t const *mem* ) throw ()

Free memory.

### Parameters

***srv*** Memory allocator service.

***mem*** Dataspace to free.

### Returns

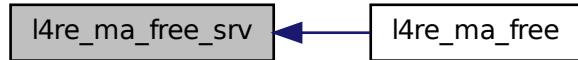
0 on success, <0 on error

### See also

[L4Re::Mem\\_alloc::free](#)

Referenced by [l4re\\_ma\\_free\(\)](#).

Here is the caller graph for this function:



## 9.10 Namespace interface

Namespace C interface.

Collaboration diagram for Namespace interface:



### Enumerations

- enum [l4re\\_ns\\_register\\_flags](#)

*Namespace register flags.*

### Functions

- long [l4re\\_ns\\_query\\_to\\_srv](#) ([l4re\\_namespace\\_t](#) srv, char const \*name, [l4\\_cap\\_idx\\_t](#) const cap, int timeout) throw ()

- long `l4re_ns_register_obj_srv` (`l4re_namespace_t` `srv`, `char const *``name`, `l4_cap_idx_t` `const obj`, `unsigned flags`) `throw ()`

### 9.10.1 Detailed Description

Namespace C interface.

### 9.10.2 Enumeration Type Documentation

#### 9.10.2.1 enum `l4re_ns_register_flags`

Namespace register flags.

##### See also

[L4Re::Namespace::Register\\_flags](#)

Definition at line 39 of file `namespace.h`.

### 9.10.3 Function Documentation

#### 9.10.3.1 long `l4re_ns_query_to_srv` ( `l4re_namespace_t` `srv`, `char const *``name`, `l4_cap_idx_t` `const cap`, `int timeout` ) `throw ()`

##### Returns

0 on success, <0 on error

##### See also

[L4Re::Namespace::query](#)

#### 9.10.3.2 long `l4re_ns_register_obj_srv` ( `l4re_namespace_t` `srv`, `char const *``name`, `l4_cap_idx_t` `const obj`, `unsigned flags` ) `throw ()`

##### Returns

0 on success, <0 on error

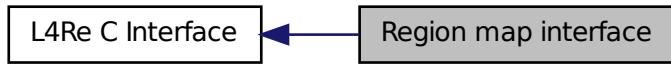
##### See also

[L4Re::Namespace::register\\_obj](#)

## 9.11 Region map interface

Region map C interface.

Collaboration diagram for Region map interface:



## Enumerations

- enum l4re\_rm\_flags\_t {
 L4RE\_RM\_READ\_ONLY = 0x01, L4RE\_RM\_NO\_ALIAS = 0x02, L4RE\_RM\_PAGER = 0x04,
 L4RE\_RM\_RESERVED = 0x08,
 L4RE\_RM\_REGION\_FLAGS = 0x0f, L4RE\_RM\_OVERMAP = 0x10, L4RE\_RM\_SEARCH\_-
 ADDR = 0x20, L4RE\_RM\_IN\_AREA = 0x40,
 L4RE\_RM\_EAGER\_MAP = 0x80, L4RE\_RM\_ATTACH\_FLAGS = 0xf0
 }
- Flags for region operations.*

## Functions

- int l4re\_rm\_reserve\_area (l4\_addr\_t \*start, unsigned long size, unsigned flags, unsigned char align) throw ()
- int l4re\_rm\_free\_area (l4\_addr\_t addr) throw ()
- int l4re\_rm\_attach (void \*\*start, unsigned long size, unsigned long flags, l4re\_ds\_t const mem, l4\_-
 addr\_t offs, unsigned char align) throw ()
- int l4re\_rm\_detach (void \*addr) throw ()  
  
*Detach and unmap in current task.*
- int l4re\_rm\_detach\_ds (void \*addr, l4re\_ds\_t \*ds) throw ()  
  
*Detach, unmap and return affected dataspace in current task.*
- int l4re\_rm\_detach\_unmap (l4\_addr\_t addr, l4\_cap\_idx\_t task) throw ()  
  
*Detach and unmap in specified task.*
- int l4re\_rm\_detach\_ds\_unmap (void \*addr, l4re\_ds\_t \*ds, l4\_cap\_idx\_t task) throw ()  
  
*Detach and unmap in specified task.*
- int l4re\_rm\_find (l4\_addr\_t \*addr, unsigned long \*size, l4\_addr\_t \*offset, unsigned \*flags, l4re\_ds\_t
 \*m) throw ()
- void l4re\_rm\_show\_lists (void) throw ()  
  
*Dump region map internal data structures.*
- int l4re\_rm\_reserve\_area\_srv (l4\_cap\_idx\_t rm, l4\_addr\_t \*start, unsigned long size, unsigned flags,
 unsigned char align) throw ()

- int `l4re_rm_free_area_srv` (`l4_cap_idx_t` rm, `l4_addr_t` addr) throw ()
- int `l4re_rm_attach_srv` (`l4_cap_idx_t` rm, void \*\*start, unsigned long size, unsigned long flags, `l4re_ds_t` const mem, `l4_addr_t` offs, unsigned char align) throw ()
- int `l4re_rm_detach_srv` (`l4_cap_idx_t` rm, `l4_addr_t` addr, `l4re_ds_t` \*ds, `l4_cap_idx_t` task) throw ()
- int `l4re_rm_find_srv` (`l4_cap_idx_t` rm, `l4_addr_t` \*addr, unsigned long \*size, `l4_addr_t` \*offset, unsigned \*flags, `l4re_ds_t` \*m) throw ()
- void `l4re_rm_show_lists_srv` (`l4_cap_idx_t` rm) throw ()

*Dump region map internal data structures.*

### 9.11.1 Detailed Description

Region map C interface.

### 9.11.2 Enumeration Type Documentation

#### 9.11.2.1 enum `l4re_rm_flags_t`

Flags for region operations.

Enumerator:

- `L4RE_RM_READ_ONLY` Region is read-only.
- `L4RE_RM_NO_ALIAS` The region contains exclusive memory that is not mapped anywhere else.
- `L4RE_RM_PAGER` Region has a pager.
- `L4RE_RM_RESERVED` Region is reserved (blocked).
- `L4RE_RM_REGION_FLAGS` Mask of all region flags.
- `L4RE_RM_OVERMAP` Unmap memory already mapped in the region.
- `L4RE_RM_SEARCH_ADDR` Search for a suitable address range.
- `L4RE_RM_IN_AREA` Search only in area, or map into area.
- `L4RE_RM_EAGER_MAP` Eagerly map the attached data space in.
- `L4RE_RM_ATTACH_FLAGS` Mask of all attach flags.

Definition at line 40 of file `rm.h`.

### 9.11.3 Function Documentation

#### 9.11.3.1 int `l4re_rm_reserve_area` ( `l4_addr_t` \* `start`, `unsigned long` `size`, `unsigned` `flags`, `unsigned char` `align` ) throw () [inline]

##### Returns

0 on success, <0 on error

##### See also

[L4Re::Rm::reserve\\_area](#)

This function is using the L4::Env::env()->rm() service.

Definition at line 229 of file [rm.h](#).

References [l4re\\_rm\\_reserve\\_area\\_srv\(\)](#), and [l4re\\_env\\_t::rm](#).

Here is the call graph for this function:



### 9.11.3.2 int l4re\_rm\_free\_area ( l4\_addr\_t addr ) throw () [inline]

#### Returns

0 on success, <0 on error

#### See also

[L4Re::Rm::free\\_area](#)

This function is using the L4::Env::env()->rm() service.

Definition at line 237 of file [rm.h](#).

References [l4re\\_rm\\_free\\_area\\_srv\(\)](#), and [l4re\\_env\\_t::rm](#).

Here is the call graph for this function:



### 9.11.3.3 int l4re\_rm\_attach ( void \*\* start, unsigned long size, unsigned long flags, l4re\_ds\_t const mem, l4\_addr\_t offs, unsigned char align ) throw () [inline]

#### Parameters

*start* Virtual start address

*size* Size of the data space to attach (in bytes)

**flags** Flags, see [Attach\\_flags](#) and [Region\\_flags](#)

**mem** Data space

**offs** Offset into the data space to use

**align** Alignment of the virtual region, log2-size, default: a page ([L4\\_PAGESHIFT](#)), Only meaningful if the [Search\\_addr](#) flag is used.

### Return values

**start** Start of region if [Search\\_addr](#) was used.

### Returns

0 on success, <0 on error

- [-L4\\_ENOENT](#)
- [-L4\\_EPERM](#)
- [-L4\\_EINVAL](#)
- [-L4\\_EADDRNOTAVAIL](#)
- IPC errors

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

### Note

When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is searched.

There is no region object created, instead the region is defined by a virtual address within this range (see [find](#)).

### Returns

0 on success, <0 on error

### See also

[L4Re::Rm::attach](#)

This function is using the L4::Env::env()->rm() service.

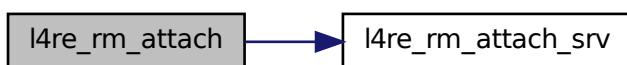
### Examples:

[examples/libs/l4re/c/ma+rm.c](#).

Definition at line 243 of file [rm.h](#).

References [l4re\\_rm\\_attach\\_srv\(\)](#), and [l4re\\_env\\_t::rm](#).

Here is the call graph for this function:



**9.11.3.4 int l4re\_rm\_detach ( void \* *addr* ) throw () [inline]**

Detach and unmap in current task.

**Parameters**

*addr* Address of the region to detach.

**Returns**

0 on success, <0 on error

Also

**See also**

[L4Re::Rm::detach](#)

This function is using the L4::Env::env()->rm() service.

Definition at line [253](#) of file [rm.h](#).

References [L4\\_BASE\\_TASK\\_CAP](#), [l4re\\_rm\\_detach\\_srv\(\)](#), and [l4re\\_env\\_t::rm](#).

Here is the call graph for this function:

**9.11.3.5 int l4re\_rm\_detach\_ds ( void \* *addr*, l4re\_ds\_t \* *ds* ) throw () [inline]**

Detach, unmap and return affected dataspace in current task.

**Parameters**

*addr* Address of the region to detach.

**Return values**

*ds* Returns dataspace that is affected.

**Returns**

0 on success, <0 on error

Also

**See also**

[L4Re::Rm::detach](#)

This function is using the L4::Env::env()->rm() service.

### Examples:

[examples/libs/l4re/c/ma+rm.c](#).

Definition at line [266](#) of file [rm.h](#).

References [L4\\_BASE\\_TASK\\_CAP](#), [l4re\\_rm\\_detach\\_srv\(\)](#), and [l4re\\_env\\_t::rm](#).

Here is the call graph for this function:



### 9.11.3.6 int l4re\_rm\_detach\_unmap ( l4\_addr\_t *addr*, l4\_cap\_idx\_t *task* ) throw () [inline]

Detach and unmap in specified task.

#### Parameters

*addr* Address of the region to detach.

*task* Task to unmap pages from, specify L4\_INVALID\_CAP to not unmap

#### Returns

0 on success, <0 on error

Also

#### See also

[L4Re::Rm::detach](#)

This function is using the L4::Env::env()->rm() service.

Definition at line [260](#) of file [rm.h](#).

References [l4re\\_rm\\_detach\\_srv\(\)](#), and [l4re\\_env\\_t::rm](#).

Here is the call graph for this function:



**9.11.3.7** `int l4re_rm_detach_ds_unmap( void * addr, l4re_ds_t * ds, l4_cap_idx_t task ) throw( ) [inline]`

Detach and unmap in specified task.

#### Parameters

*addr* Address of the region to detach.

#### Return values

*ds* Returns dataspace that is affected.

#### Parameters

*task* Task to unmap pages from, specify L4\_INVALID\_CAP to not unmap

#### Returns

0 on success, <0 on error

Also

#### See also

[L4Re::Rm::detach](#)

This function is using the L4::Env::env()->rm() service.

Definition at line 273 of file [rm.h](#).

References [l4re\\_rm\\_detach\\_srv\(\)](#), and [l4re\\_env\\_t::rm](#).

Here is the call graph for this function:



**9.11.3.8 int l4re\_rm\_find ( l4\_addr\_t \* addr, unsigned long \* size, l4\_addr\_t \* offset, unsigned \* flags, l4re\_ds\_t \* m ) throw () [inline]**

#### Returns

0 on success, <0 on error

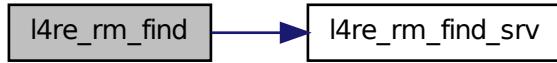
#### See also

[L4Re::Rm::find](#)

Definition at line 280 of file [rm.h](#).

References [l4re\\_rm\\_find\\_srv\(\)](#), and [l4re\\_env\\_t::rm](#).

Here is the call graph for this function:



**9.11.3.9 void l4re\_rm\_show\_lists ( void ) throw () [inline]**

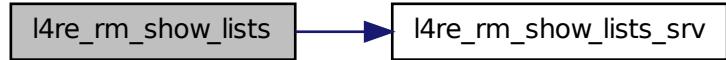
Dump region map internal data structures.

This function is using the L4::Env::env()->rm() service.

Definition at line 287 of file [rm.h](#).

References [l4re\\_rm\\_show\\_lists\\_srv\(\)](#), and [l4re\\_env\\_t::rm](#).

Here is the call graph for this function:



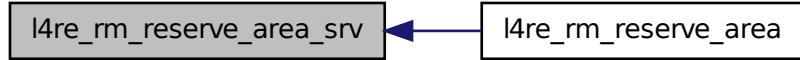
**9.11.3.10 int l4re\_rm\_reserve\_area\_srv ( l4\_cap\_idx\_t rm, l4\_addr\_t \* start, unsigned long size, unsigned flags, unsigned char align ) throw ()**

#### See also

[L4Re::Rm::reserve\\_area](#)

Referenced by [l4re\\_rm\\_reserve\\_area\(\)](#).

Here is the caller graph for this function:



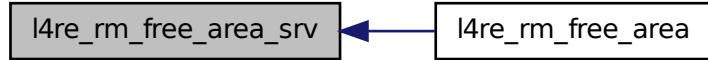
#### 9.11.3.11 int l4re\_rm\_free\_area\_srv ( l4\_cap\_idx\_t rm, l4\_addr\_t addr ) throw ()

See also

[L4Re::Rm::free\\_area](#)

Referenced by [l4re\\_rm\\_free\\_area\(\)](#).

Here is the caller graph for this function:



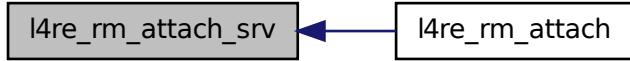
#### 9.11.3.12 int l4re\_rm\_attach\_srv ( l4\_cap\_idx\_t rm, void \*\* start, unsigned long size, unsigned long flags, l4re\_ds\_t const mem, l4\_addr\_t offs, unsigned char align ) throw ()

See also

[L4Re::Rm::attach](#)

Referenced by [l4re\\_rm\\_attach\(\)](#).

Here is the caller graph for this function:



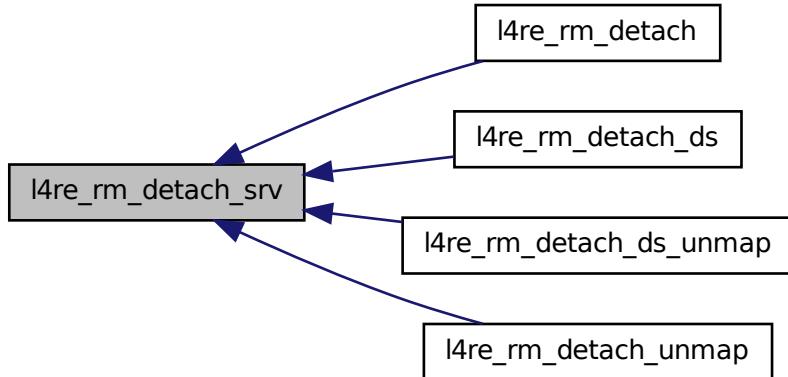
**9.11.3.13** `int l4re_rm_detach_srv ( l4_cap_idx_t rm, l4_addr_t addr, l4re_ds_t * ds, l4_cap_idx_t task ) throw ()`

See also

[L4Re::Rm::detach](#)

Referenced by [l4re\\_rm\\_detach\(\)](#), [l4re\\_rm\\_detach\\_ds\(\)](#), [l4re\\_rm\\_detach\\_ds\\_unmap\(\)](#), and [l4re\\_rm\\_detach\\_unmap\(\)](#).

Here is the caller graph for this function:



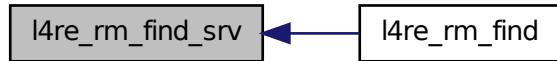
**9.11.3.14** `int l4re_rm_find_srv ( l4_cap_idx_t rm, l4_addr_t * addr, unsigned long * size, l4_addr_t * offset, unsigned * flags, l4re_ds_t * m ) throw ()`

See also

[L4Re::Rm::find](#)

Referenced by [l4re\\_rm\\_find\(\)](#).

Here is the caller graph for this function:



## 9.12 Capability allocator

Capability allocator C interface.

Collaboration diagram for Capability allocator:



### Functions

- [l4\\_cap\\_idx\\_t l4re\\_util\\_cap\\_alloc \(void\) L4\\_NOTHROW](#)  
*Get free capability index at capability allocator.*
- [void l4re\\_util\\_cap\\_free \(l4\\_cap\\_idx\\_t cap\) L4\\_NOTHROW](#)  
*Return capability index to capability allocator.*
- [void l4re\\_util\\_cap\\_free\\_um \(l4\\_cap\\_idx\\_t cap\) L4\\_NOTHROW](#)  
*Return capability index to capability allocator, and unmaps the object.*
- [long l4re\\_util\\_cap\\_last \(void\) L4\\_NOTHROW](#)  
*Return last capability index the allocator can return.*

#### 9.12.1 Detailed Description

Capability allocator C interface.

### 9.12.2 Function Documentation

#### 9.12.2.1 long l4re\_util\_cap\_last ( void )

Return last capability index the allocator can return.

##### Returns

last/biggest capability index the allocator can return

## 9.13 Kumem allocator utility

Kumem allocator utility C interface.

Collaboration diagram for Kumem allocator utility:



## Functions

- int l4re\_util\_kumem\_alloc (l4\_addr\_t \*mem, unsigned pages\_order, l4\_cap\_idx\_t task, l4\_cap\_idx\_t regmgr) L4\_NO\_THROW

*Get free capability index at capability allocator.*

### 9.13.1 Detailed Description

Kumem allocator utility C interface.

### 9.13.2 Function Documentation

#### 9.13.2.1 int l4re\_util\_kumem\_alloc ( l4\_addr\_t \* mem, unsigned pages\_order, l4\_cap\_idx\_t task, l4\_cap\_idx\_t regmgr )

Get free capability index at capability allocator.

Allocate state area.

##### Return values

*mem* Pointer to memory that has been allocated.

*pages\_order* Size to allocate, in log2 pages.

### Parameters

- **task** Task to use for allocation.
- **regmgr** Region manager to use for allocation.

### Returns

0 for success, error code otherwise

## 9.14 Video API

Collaboration diagram for Video API:



## Data Structures

- struct [l4re\\_video\\_color\\_component\\_t](#)  
*Color component structure.*
- struct [l4re\\_video\\_pixel\\_info\\_t](#)  
*Pixel\_info structure.*
- struct [l4re\\_video\\_goos\\_info\\_t](#)  
*Goos information structure.*
- struct [l4re\\_video\\_view\\_info\\_t](#)  
*View information structure.*
- struct [l4re\\_video\\_view\\_t](#)  
*C representation of a goos view.*

## Typedefs

- typedef struct [l4re\\_video\\_color\\_component\\_t](#) l4re\_video\_color\_component\_t  
*Color component structure.*
- typedef struct [l4re\\_video\\_pixel\\_info\\_t](#) l4re\_video\_pixel\_info\_t  
*Pixel\_info structure.*

- **typedef struct l4re\_video\_view\_info\_t l4re\_video\_view\_info\_t**  
*View information structure.*
- **typedef struct l4re\_video\_view\_t l4re\_video\_view\_t**  
*C representation of a goos view.*

## Enumerations

- **enum l4re\_video\_goos\_info\_flags\_t { F\_l4re\_video\_goos\_auto\_refresh = 0x01, F\_l4re\_video\_goos\_pointer = 0x02, F\_l4re\_video\_goos\_dynamic\_views = 0x04, F\_l4re\_video\_goos\_dynamic\_buffers = 0x08 }**  
*Flags of information on the goos.*
- **enum l4re\_video\_view\_info\_flags\_t {**  
**F\_l4re\_video\_view\_none = 0x00, F\_l4re\_video\_view\_set\_buffer = 0x01, F\_l4re\_video\_view\_set\_buffer\_offset = 0x02, F\_l4re\_video\_view\_set\_bytes\_per\_line = 0x04,**  
**F\_l4re\_video\_view\_set\_pixel = 0x08, F\_l4re\_video\_view\_set\_position = 0x10, F\_l4re\_video\_view\_dyn\_allocated = 0x20, F\_l4re\_video\_view\_set\_background = 0x40,**  
**F\_l4re\_video\_view\_set\_flags = 0x80, F\_l4re\_video\_view\_above = 0x01000, F\_l4re\_video\_view\_flags\_mask = 0xff000 }**  
*Flags of information on a view.*

## Functions

- **int l4re\_video\_goos\_info (l4re\_video\_goos\_t goos, l4re\_video\_goos\_info\_t \*ginfo) L4\_NOTHROW**  
*Get information on a goos.*
- **int l4re\_video\_goos\_refresh (l4re\_video\_goos\_t goos, int x, int y, int w, int h) L4\_NOTHROW**  
*Flush a rectangle of pixels of the goos screen.*
- **int l4re\_video\_goos\_create\_buffer (l4re\_video\_goos\_t goos, unsigned long size, l4\_cap\_idx\_t buffer) L4\_NOTHROW**  
*Create a new buffer (memory buffer) for pixel data.*
- **int l4re\_video\_goos\_delete\_buffer (l4re\_video\_goos\_t goos, unsigned idx) L4\_NOTHROW**  
*Delete a pixel buffer.*
- **int l4re\_video\_goos\_get\_static\_buffer (l4re\_video\_goos\_t goos, unsigned idx, l4\_cap\_idx\_t buffer) L4\_NOTHROW**  
*Get the data-space capability of the static pixel buffer.*
- **int l4re\_video\_goos\_create\_view (l4re\_video\_goos\_t goos, l4re\_video\_view\_t \*view) L4\_NOTHROW**  
*Create a new view (.*

- int `l4re_video_goops_delete_view` (`l4re_video_goops_t` `goos`, `l4re_video_view_t` \*`view`) L4\_-NOTHROW  
*Delete a view.*
- int `l4re_video_goops_get_view` (`l4re_video_goops_t` `goos`, unsigned `idx`, `l4re_video_view_t` \*`view`) L4\_-NOTHROW  
*Get a view for the given index.*
- int `l4re_video_view_refresh` (`l4re_video_view_t` \*`view`, int `x`, int `y`, int `w`, int `h`) L4\_-NOTHROW  
*Flush the given rectangle of pixels of the given view.*
- int `l4re_video_view_get_info` (`l4re_video_view_t` \*`view`, `l4re_video_view_info_t` \*`info`) L4\_-NOTHROW  
*Retrieve information about the given view.*
- int `l4re_video_view_set_info` (`l4re_video_view_t` \*`view`, `l4re_video_view_info_t` \*`info`) L4\_-NOTHROW  
*Set properties of the view.*
- int `l4re_video_view_set_viewport` (`l4re_video_view_t` \*`view`, int `x`, int `y`, int `w`, int `h`, unsigned long `bofs`) L4\_-NOTHROW  
*Set the viewport parameters of a view.*
- int `l4re_video_view_stack` (`l4re_video_view_t` \*`view`, `l4re_video_view_t` \*`pivot`, int `behind`) L4\_-NOTHROW  
*Change the stacking order in the stack of visible views.*

### 9.14.1 Typedef Documentation

#### 9.14.1.1 `typedef struct l4re_video_view_t l4re_video_view_t`

C representation of a goos view.

A view is a visible rectangle that provides a view to the contents of a buffer (frame buffer) memory object and is placed on a real screen.

### 9.14.2 Enumeration Type Documentation

#### 9.14.2.1 `enum l4re_video_goops_info_flags_t`

Flags of information on the goos.

**Enumerator:**

- F\_l4re\_video\_goops\_auto\_refresh* The graphics display is automatically refreshed.
- F\_l4re\_video\_goops\_pointer* We have a mouse pointer.
- F\_l4re\_video\_goops\_dynamic\_views* Supports dynamically allocated views.
- F\_l4re\_video\_goops\_dynamic\_buffers* Supports dynamically allocated buffers.

Definition at line 39 of file `goos.h`.

### 9.14.2.2 enum l4re\_video\_view\_info\_flags\_t

Flags of information on a view.

**Enumerator:**

- F\_l4re\_video\_view\_none* everything for this view is static (the VESA-FB case)
- F\_l4re\_video\_view\_set\_buffer* buffer object for this view can be changed
- F\_l4re\_video\_view\_set\_buffer\_offset* buffer offset can be set
- F\_l4re\_video\_view\_set\_bytes\_per\_line* bytes per line can be set
- F\_l4re\_video\_view\_set\_pixel* pixel type can be set
- F\_l4re\_video\_view\_set\_position* position on screen can be set
- F\_l4re\_video\_view\_dyn\_allocated* View is dynamically allocated.
- F\_l4re\_video\_view\_set\_background* Set view as background for session.
- F\_l4re\_video\_view\_set\_flags* Set view property flags.
- F\_l4re\_video\_view\_above* Flag the view as stay on top.
- F\_l4re\_video\_view\_flags\_mask* Mask containing all possible property flags.

Definition at line 33 of file [view.h](#).

### 9.14.3 Function Documentation

#### 9.14.3.1 int l4re\_video\_goops\_info ( l4re\_video\_goops\_t *goos*, l4re\_video\_goops\_info\_t \* *ginfo* )

Get information on a goos.

**Parameters**

*goos* Goos object

**Return values**

*ginfo* Pointer to goos information structure.

**Returns**

0 for success, <0 on error

- [-L4\\_ENODEV](#)
- IPC errors

#### 9.14.3.2 int l4re\_video\_goops\_refresh ( l4re\_video\_goops\_t *goos*, int *x*, int *y*, int *w*, int *h* )

Flush a rectangle of pixels of the goos screen.

**Parameters**

*goos* the target object of the operation.

*x* the x-coordinate of the upper left corner of the rectangle

*y* the y-coordinate of the upper left corner of the rectangle

*w* the width of the rectangle to be flushed

*h* the height of the rectangle

**9.14.3.3 int l4re\_video\_goops\_create\_buffer ( l4re\_video\_goops\_t goos, unsigned long size, l4\_cap\_idx\_t buffer )**

Create a new buffer (memory buffer) for pixel data.

**Parameters**

*goos* the target object for the operation.

*size* the size in bytes for the pixel buffer.

*buffer* a capability index to receive the data-space capability for the buffer.

**Returns**

>=0: The index of the created buffer (used to assign views and for deletion). <0: on error

**9.14.3.4 int l4re\_video\_goops\_delete\_buffer ( l4re\_video\_goops\_t goos, unsigned idx )**

Delete a pixel buffer.

**Parameters**

*goos* the target goos object.

*idx* the buffer index of the buffer to delete (the return value of [l4re\\_video\\_goops\\_create\\_buffer\(\)](#))

**9.14.3.5 int l4re\_video\_goops\_get\_static\_buffer ( l4re\_video\_goops\_t goos, unsigned idx, l4\_cap\_idx\_t buffer )**

Get the data-space capability of the static pixel buffer.

**Parameters**

*goos* the target goos object.

*buffer* a capability index to receive the data-space capability.

This function allows access to static, preexisting pixel buffers. Such static buffers exist for static configurations, such as the VESA framebuffer.

**9.14.3.6 int l4re\_video\_goops\_create\_view ( l4re\_video\_goops\_t goos, l4re\_video\_view\_t \* view )**

Create a new view (.

**See also**

[l4re\\_video\\_view\\_t](#)

**Parameters**

*goos* the goos session to use.

**Return values**

*view* the structure will be initialized for the new view.

### 9.14.3.7 int l4re\_video\_goops\_delete\_view ( l4re\_video\_goops\_t *goos*, l4re\_video\_view\_t \* *view* )

Delete a view.

#### Parameters

*goos* the goos session to use.

*view* the view to delete, the given data-structure is invalid afterwards.

### 9.14.3.8 int l4re\_video\_goops\_get\_view ( l4re\_video\_goops\_t *goos*, unsigned *idx*, l4re\_video\_view\_t \* *view* )

Get a view for the given index.

#### Parameters

*goos* the target goos session.

*idx* the index of the view to retrieve.

#### Return values

*view* the structure will be initialized to the view with the given index.

This function allows to access static views as provided by the VESA framebuffer (the monitor). However, it also allows to access dynamic views created with [l4re\\_video\\_goops\\_create\\_view\(\)](#).

### 9.14.3.9 int l4re\_video\_view\_refresh ( l4re\_video\_view\_t \* *view*, int *x*, int *y*, int *w*, int *h* )

Flush the given rectangle of pixels of the given *view*.

#### Parameters

*view* the target view of the operation.

*x* x-coordinate of the upper left corner

*y* y-coordinate of the upper left corner

*w* the width of the rectangle

*h* the height of the rectangle

### 9.14.3.10 int l4re\_video\_view\_get\_info ( l4re\_video\_view\_t \* *view*, l4re\_video\_view\_info\_t \* *info* )

Retrieve information about the given *view*.

#### Parameters

*view* the target view for the operation.

#### Return values

*info* a buffer receiving the information about the view.

**9.14.3.11 int l4re\_video\_view\_set\_info ( l4re\_video\_view\_t \* *view*, l4re\_video\_view\_info\_t \* *info* )**

Set properties of the view.

#### Parameters

*view* the target view of the operation.

*info* the parameters to be set on the view.

Which parameters can be manipulated on a given view can be figured out with [l4re\\_video\\_view\\_get\\_info\(\)](#) and this depends on the concrete instance the view object.

**9.14.3.12 int l4re\_video\_view\_set\_viewport ( l4re\_video\_view\_t \* *view*, int *x*, int *y*, int *w*, int *h*, unsigned long *bofs* )**

Set the viewport parameters of a view.

#### Parameters

*view* the target view of the operation.

*x* the x-coordinate of the upper left corner on the screen.

*y* the y-coordinate of the upper left corner on the screen.

*w* the width of the view.

*h* the height of the view.

*bofs* the offset (in bytes) of the upper left pixel in the memory buffer

This function is a convenience wrapper for [l4re\\_video\\_view\\_set\\_info\(\)](#), just setting the often changed parameters of a dynamic view. With this function a view can be placed on the real screen and at the same time on its backing buffer.

**9.14.3.13 int l4re\_video\_view\_stack ( l4re\_video\_view\_t \* *view*, l4re\_video\_view\_t \* *pivot*, int *behind* )**

Change the stacking order in the stack of visible views.

#### Parameters

*view* the target view for the operation.

*pivot* the neighbor view, relative to which *view* shall be stacked. a NULL value allows top (*behind* = 1) and bottom (*behind* = 0) placement of the view.

*behind* describes the placement of the view relative to the *pivot* view.

## 9.15 Console API

[Console interface](#).

Collaboration diagram for Console API:



## Data Structures

- class [L4Re::Console](#)

*Console class.*

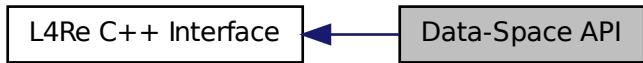
### 9.15.1 Detailed Description

[Console interface.](#)

## 9.16 Data-Space API

Data-Space API.

Collaboration diagram for Data-Space API:



## Data Structures

- class [L4Re::Dataspace](#)

*This class represents a data space.*

- struct [L4Re::Dataspace::Stats](#)

*Information about the data space.*

### 9.16.1 Detailed Description

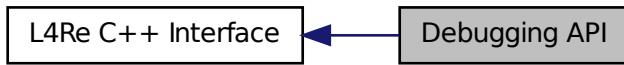
Data-Space API. Data spaces are a central abstraction provided by L4Re. A data space is an abstraction for any thing that is available via usual memory access instructions. A data space can be a file, as well as the memory-mapped registers of a device, or anonymous memory, such as a heap.

The data space interface defines a set of methods that allow any kind of data space to be attached (mapped) to the virtual address space of an L4 task and then be accessed via memory-access instructions. The [region-map interface \(L4Re::Rm\)](#) can be used to attach a data space to a virtual address space of a task paged by a certain instance of a region map ([L4Re::Rm](#)).

## 9.17 Debugging API

Debugging Interface.

Collaboration diagram for Debugging API:



### Data Structures

- class [L4Re::Debug\\_obj](#)

*Debug interface.*

### 9.17.1 Detailed Description

Debugging Interface. The debugging interface can be provided to retrieve, or log debugging information for an object. Each class may realize the debug interface to provide debugging functionality. For example, the region-map objects provide a facility to dump the currently established memory regions.

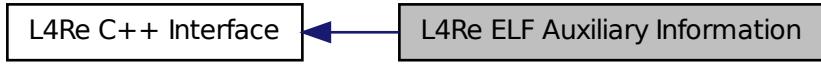
#### See also

[L4::Debug\\_obj](#) for more information.

## 9.18 L4Re ELF Auxiliary Information

API for embedding auxiliary information into binary programs.

Collaboration diagram for L4Re ELF Auxiliary Information:



## Data Structures

- struct [l4re\\_elf\\_aux\\_t](#)  
*Generic header for each auxiliary vector element.*
- struct [l4re\\_elf\\_aux\\_vma\\_t](#)  
*Auxiliary vector element for a reserved virtual memory area.*
- struct [l4re\\_elf\\_aux\\_mword\\_t](#)  
*Auxiliary vector element for a single unsigned data word.*

## Defines

- #define [L4RE\\_ELF\\_AUX\\_ELEM](#) const \_\_attribute\_\_((used, section(".rol4re\_elf\_aux")))
   
*Define an auxiliary vector element.*
- #define [L4RE\\_ELF\\_AUX\\_ELEM\\_T](#)(type, id, tag, val...)
   
*Define an auxiliary vector element.*

## Typedefs

- typedef struct [l4re\\_elf\\_aux\\_t](#) [l4re\\_elf\\_aux\\_t](#)  
*Generic header for each auxiliary vector element.*
- typedef struct [l4re\\_elf\\_aux\\_vma\\_t](#) [l4re\\_elf\\_aux\\_vma\\_t](#)  
*Auxiliary vector element for a reserved virtual memory area.*
- typedef struct [l4re\\_elf\\_aux\\_mword\\_t](#) [l4re\\_elf\\_aux\\_mword\\_t](#)  
*Auxiliary vector element for a single unsigned data word.*

## Enumerations

- enum {
   
L4RE\_ELF\_AUX\_T\_NONE = 0, L4RE\_ELF\_AUX\_T\_VMA, L4RE\_ELF\_AUX\_T\_STACK\_SIZE,
 L4RE\_ELF\_AUX\_T\_STACK\_ADDR,
   
L4RE\_ELF\_AUX\_T\_KIP\_ADDR }

### 9.18.1 Detailed Description

API for embedding auxiliary information into binary programs. This API allows information for the binary loader to be embedded into a binary application. This information can be reserved areas in the virtual memory of an application and things such as the stack size to be allocated for the first application thread.

### 9.18.2 Define Documentation

#### 9.18.2.1 #define L4RE\_ELF\_AUX\_ELEM const \_\_attribute\_\_((used, section(".rol4re\_elf\_aux")))

Define an auxiliary vector element.

This is the generic method for defining auxiliary vector elements. A more convenient way is to use L4RE\_ELF\_AUX\_ELEM\_T.

Usage:

```
L4RE_ELF_AUX_ELEM l4re_elf_aux_vma_t decl_name =
{ L4RE_ELF_AUX_T_VMA, sizeof(l4re_elf_aux_vma_t), 0x2000, 0x4000 };
```

Definition at line 52 of file [elf\\_aux.h](#).

#### 9.18.2.2 #define L4RE\_ELF\_AUX\_ELEM\_T( type, id, tag, val... )

**Value:**

```
static const __attribute__((used, section(".rol4re_elf_aux"))) \
type id = {tag, sizeof(type), val}
```

Define an auxiliary vector element.

**Parameters**

*type* is the data type for the element (e.g., [l4re\\_elf\\_aux\\_vma\\_t](#))

*id* is the identifier (variable name) for the declaration (the variable is defined with static storage class)

*tag* is the tag value for the element e.g., [L4RE\\_ELF\\_AUX\\_T\\_VMA](#)

*val* are the values to be set in the descriptor

Usage:

```
L4RE_ELF_AUX_ELEM_T(l4re_elf_aux_vma_t, decl_name, L4RE_ELF_AUX_T_VMA, 0x2000, 0
x4000 );
```

Definition at line 67 of file [elf\\_aux.h](#).

### 9.18.3 Enumeration Type Documentation

#### 9.18.3.1 anonymous enum

Enumerator:

`L4RE_ELF_AUX_T_NONE` Tag for an invalid element in the auxiliary vector.

`L4RE_ELF_AUX_T_VMA` Tag for descriptor for a reserved virtual memory area.

`L4RE_ELF_AUX_T_STACK_SIZE` Tag for descriptor that defines the stack size for the first application thread.

`L4RE_ELF_AUX_T_STACK_ADDR` Tag for descriptor that defines the stack address for the first application thread.

`L4RE_ELF_AUX_T_KIP_ADDR` Tag for descriptor that defines the KIP address for the binaries address space.

Definition at line 71 of file [elf\\_aux.h](#).

## 9.19 Initial Environment

Environment that is initially provided to an L4 task.

Collaboration diagram for Initial Environment:



## Data Structures

- class [L4Re::Env](#)  
*Initial Environment (C++ version).*
- struct [l4re\\_env\\_cap\\_entry\\_t](#)  
*Entry in the L4Re environment array for the named initial objects.*
- struct [l4re\\_env\\_t](#)  
*Initial Environment structure (C version).*

## Typedefs

- typedef struct [l4re\\_env\\_cap\\_entry\\_t](#) [l4re\\_env\\_cap\\_entry\\_t](#)  
*Entry in the L4Re environment array for the named initial objects.*

- `typedef struct l4re_env_t l4re_env_t`  
*Initial Environment structure (C version).*

## Functions

- `l4re_env_t * l4re_env (void) throw ()`  
*Get L4Re initial environment (C version).*
- `l4_kernel_info_t * l4re_kip (void) throw ()`  
*Get Kernel Info Page.*
- `l4_cap_idx_t l4re_get_env_cap (char const *name) throw ()`  
*Get the capability selector for the object named name.*
- `l4_cap_idx_t l4re_get_env_cap_e (char const *name, l4re_env_t const *e) throw ()`  
*Get the capability selector for the object named name.*
- `l4re_env_cap_entry_t const * l4re_get_env_cap_l (char const *name, unsigned l, l4re_env_t const *e) throw ()`  
*Get the full l4re\_env\_cap\_entry\_t for the object named name.*

### 9.19.1 Detailed Description

Environment that is initially provided to an [L4](#) task. The initial environment is provided to each [L4](#) task that is started by an [L4Re](#) conform loader, such as the Moe root task. The initial environment provides access to a set of initial capabilities and some additional information about the available resources, such as free UTCBs (see [Virtual Registers](#)) and available entries in capability table (provided by the micro kernel).

The initial set of capabilities is:

- C[parent:L4Re::Parent] --- parent object
- C[mem\_alloc:L4Re::Mem\_alloc] --- initial memory allocator
- C[log:L4Re::Log] --- logging facility
- C[main\_thread:L4::Thread] --- first application thread
- C[rm:L4::Rm] --- region manager
- C[factory:L4::Factory] --- factory to create kernel objects
- C[task:L4::Task] --- the task itself

Additional information is:

- First free entry in capability table
- The [UTCB](#) area (as flex page)
- First free UTCB (address in the UTCB area)

**See also**

[L4Re::Env](#), [l4re\\_env\\_t](#) for more information.

### 9.19.2 Typedef Documentation

#### 9.19.2.1 `typedef struct l4re_env_t l4re_env_t`

Initial Environment structure (C version).

**See also**

[Initial environment](#)

### 9.19.3 Function Documentation

#### 9.19.3.1 `l4re_env_t * l4re_env( void ) throw() [inline]`

Get [L4Re](#) initial environment (C version).

**Returns**

Pointer to [L4Re](#) initial environment (C version).

**Examples:**

[examples/sys/aliens/main.c](#), [examples/sys/isr/main.c](#), [examples/sys/singlestep/main.c](#),  
[examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 164 of file [env.h](#).

Referenced by [l4re\\_get\\_env\\_cap\(\)](#).

Here is the caller graph for this function:



#### 9.19.3.2 `l4_kernel_info_t * l4re_kip( void ) throw() [inline]`

Get Kernel Info Page.

**Returns**

Pointer to Kernel Info Page (KIP) structure.

**Examples:**

[examples/sys/aliens/main.c](#), and [examples/sys/ux-vhw/main.c](#).

Definition at line 168 of file [env.h](#).

**9.19.3.3 l4\_cap\_idx\_t l4re\_get\_env\_cap ( char const \* name ) throw () [inline]**

Get the capability selector for the object named *name*.

**Parameters**

*name* is the name of the object to lookup in the initial objects.

**Returns**

A valid capability selector if the object exists or an invalid capability selector if not ([l4\\_is\\_invalid\\_cap\(\)](#)).

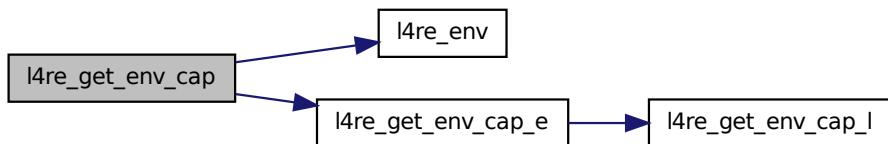
**Examples:**

[examples/sys/isr/main.c](#).

Definition at line 205 of file [env.h](#).

References [l4re\\_env\(\)](#), and [l4re\\_get\\_env\\_cap\\_e\(\)](#).

Here is the call graph for this function:

**9.19.3.4 l4\_cap\_idx\_t l4re\_get\_env\_cap\_e ( char const \* name, l4re\_env\_t const \* e ) throw () [inline]**

Get the capability selector for the object named *name*.

**Parameters**

*name* is the name of the object to lookup in the initial objects.

*e* is the environment structure to use for the operation.

**Returns**

A valid capability selector if the object exists or an invalid capability selector if not ([l4\\_is\\_invalid\\_cap\(\)](#)).

Definition at line 192 of file `env.h`.

References `l4re_env_cap_entry_t::cap`, and `l4re_get_env_cap_l()`.

Referenced by `l4re_get_env_cap()`.

Here is the call graph for this function:



Here is the caller graph for this function:



### 9.19.3.5 `l4re_env_cap_entry_t const * l4re_get_env_cap_l( char const * name, unsigned l, l4re_env_t const * e ) throw() [inline]`

Get the full `l4re_env_cap_entry_t` for the object named `name`.

#### Parameters

`name` is the name of the object to lookup in the initial objects.

`l` is the length of the name string, thus `name` might not be zero terminated.

`e` is the environment structure to use for the operation.

#### Returns

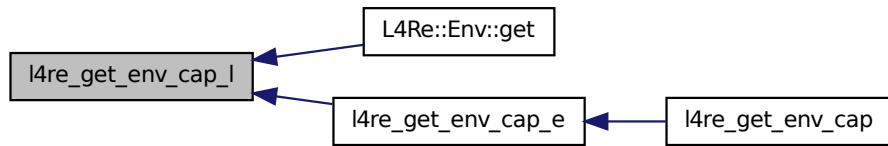
A pointer to an `l4re_env_cap_entry_t` if the object exists or NULL if not.

Definition at line 175 of file `env.h`.

References `l4re_env_cap_entry_t::flags`, and `l4re_env_cap_entry_t::name`.

Referenced by `L4Re::Env::get()`, and `l4re_get_env_cap_e()`.

Here is the caller graph for this function:



## 9.20 Event API

[Event](#) interface.

Collaboration diagram for Event API:



## Data Structures

- class [L4Re::Event](#)

*Event class.*

- class [L4Re::Event\\_buffer\\_t< PAYLOAD >](#)

*Event buffer class.*

### 9.20.1 Detailed Description

[Event](#) interface.

## 9.21 Auxiliary data

Collaboration diagram for Auxiliary data:



### Data Structures

- struct [l4re\\_aux\\_t](#)  
*Auxiliary descriptor.*

### Typedefs

- typedef struct [l4re\\_aux\\_t](#) [l4re\\_aux\\_t](#)  
*Auxiliary descriptor.*

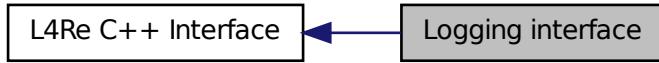
### Enumerations

- enum [l4re\\_aux\\_ldr\\_flags\\_t](#)  
*Flags for program loading.*

## 9.22 Logging interface

Interface for log output.

Collaboration diagram for Logging interface:



## Data Structures

- class [L4Re::Log](#)

*Log interface class.*

### 9.22.1 Detailed Description

Interface for log output. The logging interface provides a facility sending log output. One purpose of the interface is to serialize the output and provide the possibility to tag output sent to a specific log object.

## 9.23 Memory allocator API

Memory-allocator interface.

Collaboration diagram for Memory allocator API:



## Data Structures

- class [L4Re::Mem\\_alloc](#)

*Memory allocator.*

### 9.23.1 Detailed Description

Memory-allocator interface. The memory-allocator API is the basic API to allocate memory from the [L4Re](#) subsystem. The memory is allocated in terms of data spaces (see [L4Re::Dataspace](#)). The provided data spaces have at least the property that data written to such a data space is available as long as the data space is not freed or the data is not overwritten. In particular, the memory backing a data space from an allocator need not be allocated instantly, but may be allocated lazily on demand.

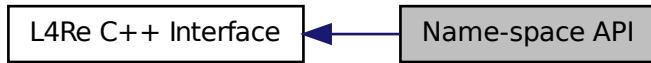
A memory allocator can provide data spaces with additional properties, such as physically contiguous memory, pre-allocated memory, or pinned memory. To request memory with an additional property the [L4Re::Mem\\_alloc::alloc\(\)](#) method provides a flags parameter. If the concrete implementation of a memory allocator does not support or allow allocation of memory with a certain property, the allocation may be refused.

The main interface is defined by the class [L4Re::Mem\\_alloc](#).

## 9.24 Name-space API

API for name spaces that store capabilities.

Collaboration diagram for Name-space API:



### Data Structures

- class [L4Re::Namespace](#)  
*Name-space interface.*

#### 9.24.1 Detailed Description

API for name spaces that store capabilities. This is a basic abstraction for managing a mapping from human-readable names to capabilities. In particular, a name can also be mapped to a capability that refers to another name space object. By this means name spaces can be constructed hierarchically.

Name spaces play a central role in [L4Re](#), because the implementation of the name space objects determine the policy which capabilities (which objects) are accessible to a client of a name space.

## 9.25 Parent API

[Parent](#) interface.

Collaboration diagram for Parent API:



### Data Structures

- class [L4Re::Parent](#)

*Parent* interface.

### 9.25.1 Detailed Description

[Parent](#) interface. The parent interface provides means for an [L4](#) task to signal changes in its execution state. The main purpose is to signal program termination.

#### See also

[L4Re::Parent](#) for information about the concrete interface.

## 9.26 L4Re Protocol identifiers

Basic protocol identifiers used for [L4Re](#).

Collaboration diagram for L4Re Protocol identifiers:



## Enumerations

- enum [L4Re::Dataspace\\_::Opcodes](#)  
*Data-space communication-protocol opcodes.*
- enum [L4Re::Event\\_::Opcodes](#)  
*Event communication-protocol opcodes.*
- enum [L4Re::Log\\_::Opcodes](#)  
*Logging-service communication-protocol opcodes.*
- enum [L4Re::Mem\\_alloc\\_::Opcodes](#)  
*Memory-allocator communication-protocol opcodes.*
- enum [L4Re::Namespace\\_::Opcodes](#)  
*Name-space communication-protocol opcodes.*
- enum [L4Re::Parent\\_::Opcodes](#)  
*Parent communication-protocol opcodes.*

- enum L4Re::Protocol::Protocols {
   
    L4Re::Protocol::Default = 0,   L4Re::Protocol::Dataspace,   L4Re::Protocol::Namespace,  
    L4Re::Protocol::Parent,  
    L4Re::Protocol::Goos, L4Re::Protocol::Mem\_alloc, L4Re::Protocol::Rm, L4Re::Protocol::Event,  
    L4Re::Protocol::Debug = ~0x7ffffUL }
   
*Protocols*  
*These protocol IDs are used to distinguish requests for the different L4Re interfaces.*
- enum L4Re::Rm\_::Opcodes  
*Region-map communication-protocol opcodes.*
- enum L4Re::Video::Goos\_::Opcodes  
*Frame buffer communication-protocol opcodes.*

### 9.26.1 Detailed Description

Basic protocol identifiers used for L4Re.

### 9.26.2 Enumeration Type Documentation

#### 9.26.2.1 enum L4Re::Protocol::Protocols

Protocols

These protocol IDs are used to distinguish requests for the different L4Re interfaces.

The interfaces use different protocol IDs to enable objects that realize a set of those interfaces at once.

**Enumerator:**

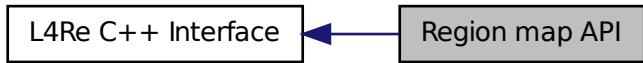
- Default** Default protocol, used in message tag.
- Dataspace** ID for data space objects.
- Namespace** ID for name space objects.
- Parent** ID for parent objects.
- Goos** ID for goos objects.
- Mem\_alloc** ID for memory allocator objects.
- Rm** ID for region map objects.
- Event** ID for event channel objects.
- Debug** ID for debug objects.

Definition at line 44 of file [protocols](#).

## 9.27 Region map API

Virtual address-space management.

Collaboration diagram for Region map API:



## Data Structures

- class [L4Re::Rm](#)

*Region map.*

### 9.27.1 Detailed Description

Virtual address-space management. The central purpose of the region-map API is to provide means to manage the virtual memory address space of an [L4](#) task. A region-map object implements two protocols. The first protocol is the kernel page-fault protocol, to resolve page faults for threads running in an [L4](#) task. The second protocol is the region-map protocol itself, that allows to attach a data-space object to a region of the virtual address space.

There are two basic concepts provided by a region-map abstraction:

- Regions provide a means to create a view to a data space (or parts of a data space).
- Areas provide a means to reserve areas in a virtual memory address space for special purposes. A reserved area is skipped when searching for an available range of virtual memory, or may be explicitly used to search only within that area.

#### See also

[Data-Space API.](#) , [L4Re::Dataspace](#), [L4Re::Rm](#)

## 9.28 L4Re Capability API

Collaboration diagram for L4Re Capability API:



## Data Structures

- class [L4Re::Cap\\_alloc](#)  
*Capability allocator interface.*
- class [L4Re::Smart\\_cap\\_auto< Unmap\\_flags >](#)  
*Helper for Auto\_cap and Auto\_del\_cap.*
- class [L4Re::Util::Smart\\_cap\\_auto< Unmap\\_flags >](#)  
*Helper for Auto\_cap and Auto\_del\_cap.*
- class [L4Re::Util::Smart\\_count\\_cap< Unmap\\_flags >](#)  
*Helper for Ref\_cap and Ref\_del\_cap.*
- struct [L4Re::Util::Auto\\_cap< T >](#)  
*Automatic capability that implements automatic free and unmap of the capability selector.*
- struct [L4Re::Util::Auto\\_del\\_cap< T >](#)  
*Automatic capability that implements automatic free and unmap+delete of the capability selector.*
- struct [L4Re::Util::Ref\\_cap< T >](#)  
*Automatic capability that implements automatic free and unmap of the capability selector.*
- struct [L4Re::Util::Ref\\_del\\_cap< T >](#)  
*Automatic capability that implements automatic free and unmap+delete of the capability selector.*

## Functions

- virtual [L4Re::Cap\\_alloc::~Cap\\_alloc \(\)=0](#)  
*Destructor.*

## Variables

- [\\_Cap\\_alloc](#) & [L4Re::Util::cap\\_alloc](#)  
*Capability allocator.*

### 9.28.1 Variable Documentation

#### 9.28.1.1 [\\_Cap\\_alloc& L4Re::Util::cap\\_alloc](#)

Capability allocator.

This is the instance of the capability allocator that is used by usual applications. The actual implementation of the allocator depends on the configuration of the system.

Per default we use a reference count capability allocator, that keeps a reference counter for each managed capability selector.

**Note**

This capability allocator is not thread-safe.

**Examples:**

[examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#),  
[examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#), and [examples/libs/l4re/streammap/client.cc](#).

Referenced by [L4Re::Util::Smart\\_count\\_cap< Unmap\\_flags >::copy\(\)](#), and [L4Re::Util::Smart\\_cap\\_-auto< Unmap\\_flags >::free\(\)](#).

## 9.29 Kumem utilties

Collaboration diagram for Kumem utilties:



## Functions

- int [L4Re::Util::kumem\\_alloc](#) (*l4\_addr\_t* \**mem*, *unsigned pages\_order*, *L4::Cap< L4::Task >* *task*=*L4Re::Env::env()->task()*, *L4::Cap< L4Re::Rm >* *rm*=*L4Re::Env::env()->rm()*) *throw ()*  
*Allocate state area.*

### 9.29.1 Function Documentation

#### 9.29.1.1 int L4Re::Util::kumem\_alloc ( *l4\_addr\_t* \* *mem*, *unsigned pages\_order*, *L4::Cap< L4::Task >* *task* = *L4Re::Env::env()->task()*, *L4::Cap< L4Re::Rm >* *rm* = *L4Re::Env::env()->rm()* ) *throw ()*

*Allocate state area.*

**Return values**

- mem* Pointer to memory that has been allocated.  
*pages\_order* Size to allocate, in log2 pages.

**Parameters**

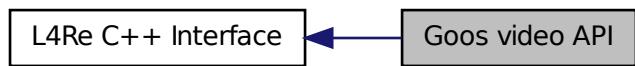
- task* Task to use for allocation.  
*rm* Region manager to use for allocation.

## Returns

0 for success, error code otherwise

## 9.30 Goos video API

Collaboration diagram for Goos video API:



## Data Structures

- class [L4Re::Video::Color\\_component](#)

*A color component.*

- class [L4Re::Video::Pixel\\_info](#)

*Pixel information.*

- class [L4Re::Video::Goos](#)

*A goos.*

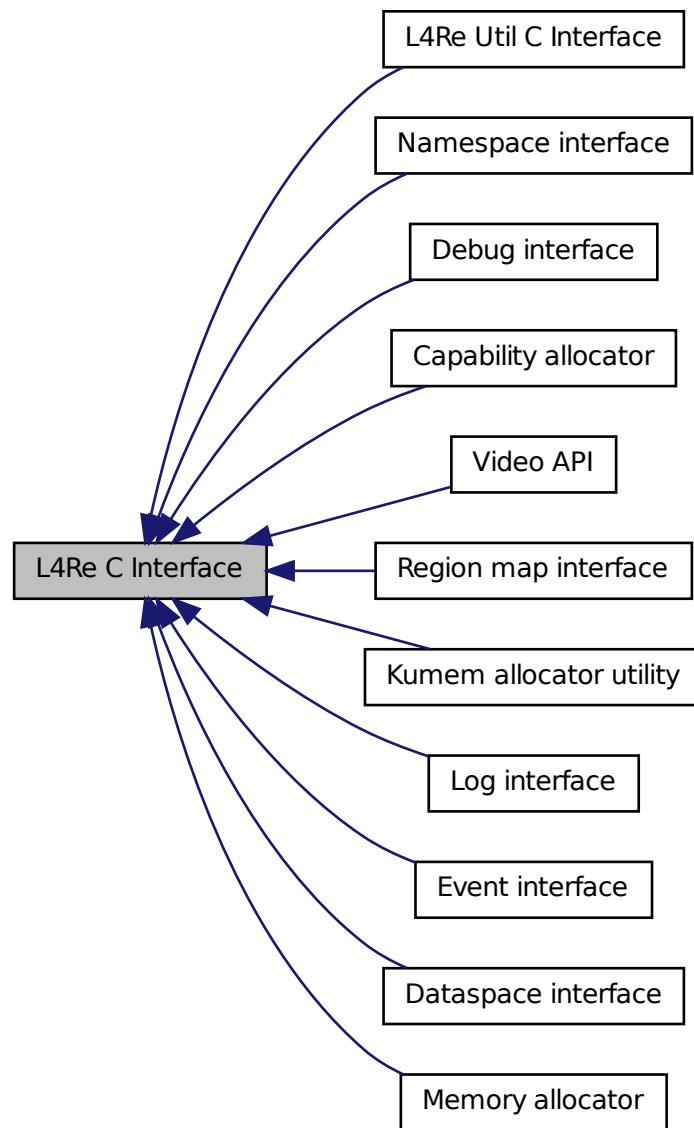
- class [L4Re::Video::View](#)

*View.*

## 9.31 L4Re C Interface

Documentation for the [L4Re](#) C Interface.

Collaboration diagram for L4Re C Interface:



## Modules

- [Dataspace interface](#)

*Dataspace C interface.*

- [Debug interface](#)
- [Event interface](#)

*Event C interface.*

- [Log interface](#)

*Log C interface.*

- [Memory allocator](#)

*Memory allocator C interface.*

- [Namespace interface](#)

*Namespace C interface.*

- [Region map interface](#)

*Region map C interface.*

- [Capability allocator](#)

*Capability allocator C interface.*

- [Kumem allocator utility](#)

*Kumem allocator utility C interface.*

- [Video API](#)

- [L4Re Util C Interface](#)

*Documentation of the L4 Runtime Environment utility functionality in C.*

### 9.31.1 Detailed Description

Documentation for the [L4Re](#) C Interface. The interface functions closely align with the C++ functions and add no further functionalities.

For new programs it is advised to use the C++ interface.

## 9.32 L4Re Util C Interface

Documentation of the [L4](#) Runtime Environment utility functionality in C.

Collaboration diagram for L4Re Util C Interface:



## Functions

- `l4_mshtag_t l4re_util_cap_release (l4_cap_idx_t cap)`

*Release a capability from a task (unmap).*

### 9.32.1 Detailed Description

Documentation of the L4 Runtime Environment utility functionality in C. The interface functions closely align with the C++ functions and add no further functionalities.

For new programs it is advised to use the C++ interface.

### 9.32.2 Function Documentation

#### 9.32.2.1 `l4_mshtag_t l4re_util_cap_release ( l4_cap_idx_t cap ) [inline]`

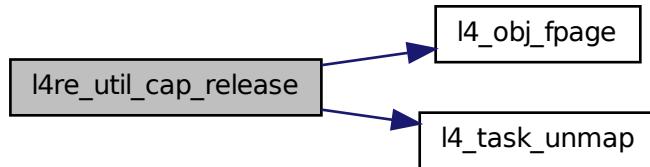
Release a capability from a task (unmap).

Note that the given capability does not need to be handled by the capability allocator in any way.

Definition at line 51 of file `cap.h`.

References `L4_BASE_TASK_CAP`, `L4_FP_ALL_SPACES`, `l4_obj_fpage()`, and `l4_task_unmap()`.

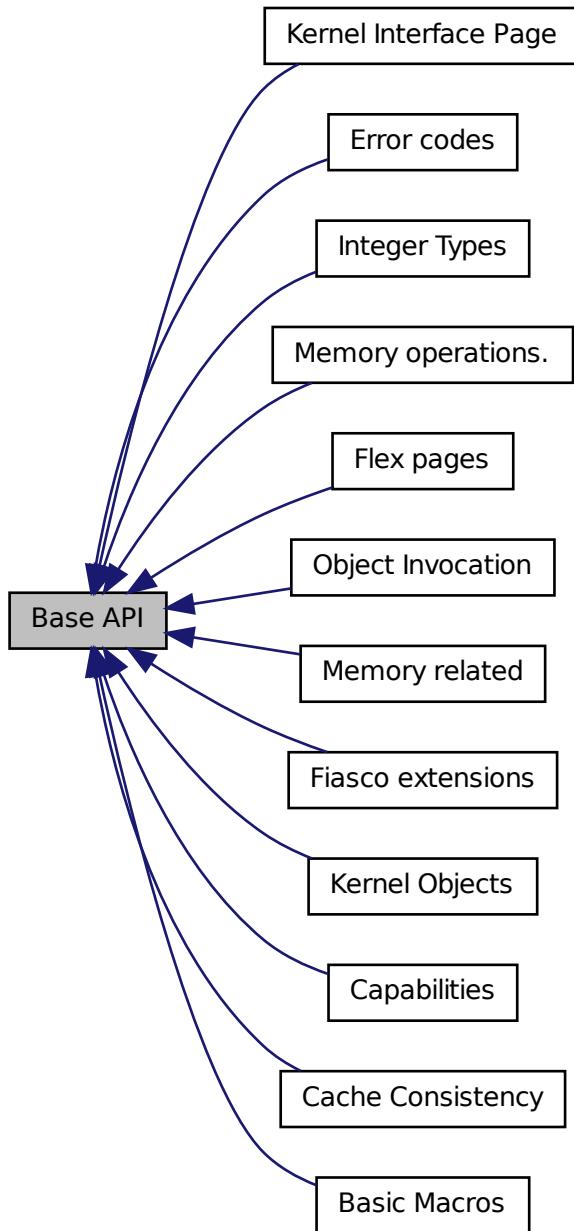
Here is the call graph for this function:



## 9.33 Base API

Interfaces for all kinds of base functionality.

Collaboration diagram for Base API:



## Modules

- [Basic Macros](#)

*L4 standard macros for header files, function definitions, and public APIs etc.*

- **Fiasco extensions**

*Kernel debugger extensions of the Fiasco L4 implementation.*

- **Flex pages**

*Flex-page related API.*

- **Cache Consistency**

*Various functions for cache consistency.*

- **Memory related**

*Memory related constants, data types and functions.*

- **Error codes**

*Common error codes.*

- **Object Invocation**

*API for L4 object invocation.*

- **Kernel Objects**

*API of kernel objects.*

- **Kernel Interface Page**

*Kernel Interface Page.*

- **Capabilities**

*Functions and definitions related to capabilities.*

- **Memory operations.**

*Operations for memory access.*

- **Integer Types**

*#include<l4/sys/l4int.h>*

## Files

- file **cache.h**

*Cache-consistency functions.*

- file **compiler.h**

*L4 compiler related defines.*

- file **consts.h**

*Common constants.*

- file **debugger.h**

*Debugger related definitions.*

- file [factory](#)

*Common factory related definitions.*

- file [factory.h](#)

*Common factory related definitions.*

- file [icu](#)

*Interrupt controller.*

- file [icu.h](#)

*Interrupt controller.*

- file [ipc.h](#)

*Common IPC interface.*

- file [irq](#)

*Interrupt functionality.*

- file [irq.h](#)

*Interrupt functionality.*

- file [kip](#)

*L4::Kip class, memory descriptors.*

- file [kip.h](#)

*Kernel Info Page access functions.*

- file [memdesc.h](#)

*Memory description functions.*

- file [meta](#)

*Meta interface for getting dynamic type information about objects behind capabilities.*

- file [types.h](#)

*Common L4 ABI Data Types.*

- file [vhw.h](#)

*Descriptors for virtual hardware (under UX).*

- file [consts.h](#)

*Common L4 constants, arm version.*

- file [types.h](#)

*L4 kernel API type definitions.*

- file [consts.h](#)

*Common L4 constants, amd64 version.*

- file [ipc.h](#)

*L4 IPC System Calls, x86.*

- file `types.h`

*L4 kernel API type definitions.*

- file `consts.h`

*Common L4 constants, x86 version.*

### 9.33.1 Detailed Description

Interfaces for all kinds of base functionality. Some notes on Inter Process Communication (IPC)

IPC in L4 is always synchronous and unbuffered: a message is transferred from the sender to the recipient if and only if the recipient has invoked a corresponding IPC operation. The sender blocks until this happens or a timeout specified by the sender elapses without the destination becoming ready to receive.

## 9.34 IPC-Gate API

Secure communication object.

Collaboration diagram for IPC-Gate API:



### Data Structures

- class `L4::Ipc_gate`

*L4 IPC gate.*

### Enumerations

- enum `L4_ipc_gate_ops` { `L4_IPC_GATE_OP_BIND` = 0x10, `L4_IPC_GATE_OP_GET_INFO` = 0x11 }

*Operations on the IPC-gate.*

### Functions

- `l4_msgtag_t l4_ipc_gate_bind_thread (l4_cap_idx_t gate, l4_cap_idx_t thread, l4_umword_t label)`

*Bind the IPC-gate to the thread.*

- `l4_mshtag_t l4_ipc_gate_get_infos (l4_cap_idx_t gate, l4_umword_t *label)`

*Get information on the IPC-gate.*

### 9.34.1 Detailed Description

Secure communication object. IPC-Gate objects provide a means to establish secure communication channels to L4 Threads (Thread). An IPC-Gate object can be created using a Factory (`l4_factory_create_gate()`) and get assigned a specific L4 thread and a *label* as protected payload. The *label* has the size of one machine word and can only be seen by the Task running the thread that is assigned of the IPC-gate. The *label* is received as part of the IPC message. The *label* can thus be used to securely identify the IPC-gate that was used to send a message.

An IPC-gate is usually used to represent an user-level object and may be the address of the data structure for the object in the server task.

With client privileges an IPC-gate does not provide any direct API and thus an IPC-gate kernel object cannot be modified by invocations. Each invocation of an IPC-gate kernel object is translated into an IPC message to the assigned thread.

#### See also

[Object Invocation](#)

### 9.34.2 Enumeration Type Documentation

#### 9.34.2.1 enum L4\_ipc\_gate\_ops

Operations on the IPC-gate.

Enumerator:

`L4_IPC_GATE_OP_BIND` Bind operation.

`L4_IPC_GATE_OP_GET_INFO` Info operation.

Definition at line 75 of file [ipc\\_gate.h](#).

### 9.34.3 Function Documentation

#### 9.34.3.1 `l4_mshtag_t l4_ipc_gate_bind_thread ( l4_cap_idx_t gate, l4_cap_idx_t thread, l4_umword_t label ) [inline]`

Bind the IPC-gate to the thread.

#### Parameters

*t* Thread to bind the IPC-gate to

*label* Label to use

*utc* UTCB to use.

### Returns

System call return tag.

Definition at line 117 of file [ipc\\_gate.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.34.3.2 `l4_mshtag_t l4_ipc_gate_get_infos ( l4_cap_idx_t gate, l4_umword_t * label ) [inline]`

Get information on the IPC-gate.

#### Return values

*label* Label of the gate.

#### Parameters

*utcb* UTCb to use.

#### Returns

System call return tag.

Definition at line 124 of file [ipc\\_gate.h](#).

References [l4\\_utcb\(\)](#).

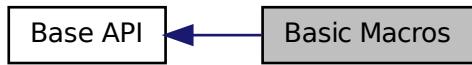
Here is the call graph for this function:



## 9.35 Basic Macros

[L4](#) standard macros for header files, function definitions, and public APIs etc.

Collaboration diagram for Basic Macros:



### Defines

- `#define L4_DECLARE_CONSTRUCTOR(func, prio)`  
*L4 Inline function attribute.*
- `#define __END_DECLS`  
*End section with C types and functions.*
- `#define EXTERN_C_BEGIN`  
*Start section with C types and functions.*
- `#define EXTERN_C_END`  
*End section with C types and functions.*
- `#define EXTERN_C`  
*Mark C types and functions.*
- `#define L4_NOTHROW`  
*Mark a function declaration and definition as never throwing an exception.*
- `#define L4_EXPORT`  
*Attribute to mark functions, variables, and data types as being exported from a library.*
- `#define L4_HIDDEN`  
*Attribute to mark functions, variables, and data types as being explicitly hidden from users of a library.*
- `#define L4_NORETURN`  
*Noreturn function attribute.*
- `#define L4_NOINSTRUMENT`  
*No instrumentation function attribute.*
- `#define EXPECT_TRUE(x)`  
*Expression is likely to execute.*

- `#define EXPECT_FALSE(x)`  
*Expression is unlikely to execute.*
- `#define L4_STICKY(x)`  
*Mark symbol sticky (even not there).*
- `#define L4_DEPRECATED(s)`  
*Mark symbol deprecated.*
- `#define L4_stringify_helper(x)`  
*stringify helper.*
- `#define L4_stringify(x)`  
*stringify.*
- `#define L4_CV`  
*Define calling convention.*
- `#define L4_CV`  
*Define calling convention.*
- `#define L4_CV __attribute__((regparm(0)))`  
*Define calling convention.*

## Functions

- `void l4_barrier(void)`  
*Memory barrier.*
- `void l4_mb(void)`  
*Memory barrier.*
- `void l4_wmb(void)`  
*Write memory barrier.*

### 9.35.1 Detailed Description

L4 standard macros for header files, function definitions, and public APIs etc. `#include <l4/sys/compiler.h>`

### 9.35.2 Define Documentation

#### 9.35.2.1 `#define L4_DECLARE_CONSTRUCTOR( func, prio )`

L4 Inline function attribute.

Handcoded version of `__attribute__((constructor(xx)))`.

## Parameters

*func* function declaration (prototype)  
*prio* the prio must be 65535 - *gcc\_prio*

Definition at line 79 of file [compiler.h](#).

### 9.35.2.2 #define L4\_NOTHROW

Mark a function declaration and definition as never throwing an exception.

(Also for C code).

This macro shall be used to mark C and C++ functions that never throw any exception. Note that also C functions may throw exceptions according to the compilers ABI and shall be marked with L4\_NOTHROW if they never do. In C++ this is equivalent to `throw()`.

```
int foo() L4_NOTHROW;
...
int foo() L4_NOTHROW
{
    ...
    return result;
}
```

Definition at line 196 of file [compiler.h](#).

### 9.35.2.3 #define L4\_EXPORT

Attribute to mark functions, variables, and data types as being exported from a library.

All data types, functions, and global variables that shall be exported from a library shall be marked with this attribute. The default may become to hide everything that is not marked as L4\_EXPORT from the users of a library and provide the possibility for aggressive optimization of all those internal functionality of a library.

Usage:

```
class L4_EXPORT My_class
{
    ...
};

int L4_EXPORT function(void);

int L4_EXPORT global_data; // global data is not recommended
```

Definition at line 224 of file [compiler.h](#).

### 9.35.2.4 #define L4\_HIDDEN

Attribute to mark functions, variables, and data types as being explicitly hidden from users of a library.

This attribute is intended for functions, data, and data types that shall never be visible outside of a library. In particular, for shared libraries this may result in much faster code within the library and short linking times.

```

class L4_HIDDEN My_class
{
    ...
};

int L4_HIDDEN function(void);

int L4_HIDDEN global_data; // global data is not recommended

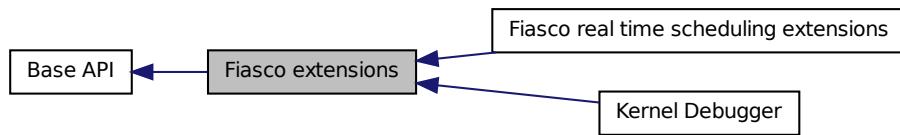
```

Definition at line 221 of file [compiler.h](#).

## 9.36 Fiasco extensions

Kernel debugger extensions of the Fiasco [L4](#) implementation.

Collaboration diagram for Fiasco extensions:



## Data Structures

- struct [l4\\_tracebuffer\\_status\\_t](#)  
*Trace buffer status.*
- struct [l4\\_tracebuffer\\_status\\_window\\_t](#)  
*Trace-buffer status window descriptor.*

## Modules

- [Fiasco real time scheduling extensions](#)  
*Real time scheduling extension for the Fiasco [L4](#) implementation.*
- [Kernel Debugger](#)  
*Kernel debugger related functionality.*

## Files

- file [segment.h](#)  
*l4f specific segment manipulation*

## Defines

- #define `LOG_EVENT_CONTEXT_SWITCH` 0  
*Event: context switch.*
- #define `LOG_EVENT_IPC_SHORTCUT` 1  
*Event: IPC shortcut.*
- #define `LOG_EVENT_IRQ_RAISED` 2  
*Event: IRQ occurred.*
- #define `LOG_EVENT_TIMER_IRQ` 3  
*Event: Timer IRQ occurred.*
- #define `LOG_EVENT_THREAD_EX_REGS` 4  
*Event: thread\_ex\_regs.*
- #define `LOG_EVENT_MAX_EVENTS` 16  
*Maximum number of events.*
- #define `LOG_EVENT_CONTEXT_SWITCH` 0  
*Event: context switch.*
- #define `LOG_EVENT_IPC_SHORTCUT` 1  
*Event: IPC shortcut.*
- #define `LOG_EVENT_IRQ_RAISED` 2  
*Event: IRQ occurred.*
- #define `LOG_EVENT_TIMER_IRQ` 3  
*Event: Timer IRQ occurred.*
- #define `LOG_EVENT_THREAD_EX_REGS` 4  
*Event: thread\_ex\_regs.*
- #define `LOG_EVENT_MAX_EVENTS` 16  
*Maximum number of events.*

## Enumerations

- enum  
*Log event types.*

## Functions

- **`l4_tracebuffer_status_t * fiasco_tbuf_get_status (void)`**  
*Return trace buffer status.*
- **`l4_addr_t fiasco_tbuf_get_status_phys (void)`**  
*Return the physical address of the trace buffer status struct.*
- **`l4_umword_t fiasco_tbuf_log (const char *text)`**  
*Create new trace buffer entry with describing <text>.*
- **`l4_umword_t fiasco_tbuf_log_3val (const char *text, unsigned v1, unsigned v2, unsigned v3)`**  
*Create new trace buffer entry with describing <text> and three additional values.*
- **`l4_umword_t fiasco_tbuf_log_binary (const unsigned char *data)`**  
*Create new trace buffer entry with binary data.*
- **`void fiasco_tbuf_clear (void)`**  
*Clear trace buffer.*
- **`void fiasco_tbuf_dump (void)`**  
*Dump trace buffer to kernel console.*
- **`void fiasco_profile_start (void) throw ()`**  
*Start profiling.*
- **`void fiasco_profile_stop_and_dump (void) throw ()`**  
*Stop profiling and dump result to console.*
- **`void fiasco_profile_stop (void) throw ()`**  
*Stop profiling.*
- **`void fiasco_watchdog_enable (void) throw ()`**  
*Enable Fiasco watchdog.*
- **`void fiasco_watchdog_disable (void) throw ()`**  
*Disable Fiasco watchdog.*
- **`void fiasco_watchdog_takeover (void) throw ()`**  
*Disable automatic resetting of watchdog.*
- **`void fiasco_watchdog_giveback (void) throw ()`**  
*Reenable automatic resetting of watchdog.*
- **`void fiasco_watchdog_touch (void) throw ()`**  
*Reset watchdog from user land.*
- **`long fiasco_ldt_set (l4_cap_idx_t task, void *ldt, unsigned int size, unsigned int entry_number_start, l4_utcb_t *utcb)`**  
*Set LDT segments descriptors.*

- long `fiasco_gdt_set` (`l4_cap_idx_t` thread, `void *desc`, `unsigned int size`, `unsigned int entry_number_start`, `l4_utcb_t *utcb`)  
*Set GDT segment descriptors.*
- unsigned `fiasco_gdt_get_entry_offset` (`l4_cap_idx_t` thread, `l4_utcb_t *utcb`)  
*Return the offset of the entry in the GDT.*

### 9.36.1 Detailed Description

Kernel debugger extensions of the Fiasco L4 implementation.

### 9.36.2 Function Documentation

#### 9.36.2.1 `l4_tracebuffer_status_t * fiasco_tbuf_get_status( void ) [inline]`

Return trace buffer status.

Return trace-buffer status.

Return tracebuffer status.

##### Returns

Pointer to trace buffer status struct.  
 Pointer to tracebuffer status struct.  
 Pointer to trace-buffer status struct.

Definition at line 171 of file `ktrace.h`.

#### 9.36.2.2 `l4_addr_t fiasco_tbuf_get_status_phys( void ) [inline]`

Return the physical address of the trace buffer status struct.

Return the physical address of the trace-buffer status struct.

Return the physical address of the tracebuffer status struct.

##### Returns

physical address of status struct.

Definition at line 178 of file `ktrace.h`.

References `enter_kdebug`.

#### 9.36.2.3 `l4_umword_t fiasco_tbuf_log( const char * text ) [inline]`

Create new trace buffer entry with describing <text>.

Create new trace-buffer entry with describing <text>.

Create new tracebuffer entry with describing <text>.

**Parameters**

*text* Logging text

**Returns**

Pointer to trace buffer entry

**Parameters**

*text* Logging text

**Returns**

Pointer to tracebuffer entry

**Parameters**

*text* Logging text

**Returns**

Pointer to trace-buffer entry

Definition at line 185 of file [ktrace.h](#).

### 9.36.2.4 l4\_umword\_t fiasco\_tbuf\_log\_3val ( const char \* *text*, unsigned *v1*, unsigned *v2*, unsigned *v3* ) [inline]

Create new trace buffer entry with describing <text> and three additional values.

Create new trace-buffer entry with describing <text> and three additional values.

Create new tracebuffer entry with describing <text> and three additional values.

**Parameters**

*text* Logging text

*v1* first value

*v2* second value

*v3* third value

**Returns**

Pointer to trace buffer entry

**Parameters**

*text* Logging text

*v1* first value

*v2* second value

*v3* third value

**Returns**

Pointer to tracebuffer entry

**Parameters**

*text* Logging text

*v1* first value

*v2* second value

*v3* third value

**Returns**

Pointer to trace-buffer entry

Definition at line 191 of file [ktrace.h](#).

**9.36.2.5 l4\_umword\_t fiasco\_tbuf\_log\_binary ( const unsigned char \* *data* ) [inline]**

Create new trace buffer entry with binary data.

Create new trace-buffer entry with binary data.

Create new tracebuffer entry with binary data.

**Parameters**

*data* binary data

**Returns**

Pointer to trace buffer entry

**Parameters**

*data* binary data

**Returns**

Pointer to tracebuffer entry

**Parameters**

*data* binary data

**Returns**

Pointer to trace-buffer entry

Definition at line 221 of file [ktrace.h](#).

**9.36.2.6 void fiasco\_tbuf\_clear ( void ) [inline]**

Clear trace buffer.

Clear trace-buffer.

Clear tracebuffer.

Definition at line 197 of file [ktrace.h](#).

**9.36.2.7 void fiasco\_tbuf\_dump( void ) [inline]**

Dump trace buffer to kernel console.

Dump trace-buffer to kernel console.

Dump tracebuffer to kernel console.

Definition at line 203 of file [ktrace.h](#).

**9.36.2.8 void fiasco\_watchdog\_takeover( void ) throw() [inline]**

Disable automatic resetting of watchdog.

User is responsible to call `fiasco_watchdog_touch` from time to time to ensure that the watchdog does not trigger.

Definition at line 407 of file [kdebug.h](#).

**9.36.2.9 void fiasco\_watchdog\_touch( void ) throw() [inline]**

Reset watchdog from user land.

This function **must** be called from time to time to prevent the watchdog from triggering if the watchdog is activated and if `fiasco_watchdog_takeover` was performed.

Definition at line 419 of file [kdebug.h](#).

**9.36.2.10 long fiasco\_ldt\_set( l4\_cap\_idx\_t task, void \* ldt, unsigned int size, unsigned int entry\_number\_start, l4\_utcb\_t \* utcb ) [inline]**

Set LDT segments descriptors.

**Parameters**

*task* Task to set the segment for.

*ldt* Pointer to LDT hardware descriptors.

*num\_desc* Number of descriptors.

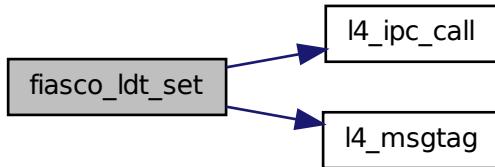
*entry\_number\_start* Entry number to start.

*utcb* UTCB of the caller.

Definition at line 94 of file [segment.h](#).

References `l4_ipc_call()`, `L4_IPC_NEVER`, `l4_mshtag()`, `L4_PROTO_TASK`, and `l4_msg_regs_t::mr`.

Here is the call graph for this function:



**9.36.2.11 long fiasco\_gdt\_set ( l4\_cap\_idx\_t *thread*, void \* *desc*, unsigned int *size*, unsigned int *entry\_number\_start*, l4\_utcb\_t \* *utcb* ) [inline]**

Set GDT segment descriptors.

Fiasco supports 3 consecutive entries, starting at the value returned by [fiasco\\_gdt\\_get\\_entry\\_offset\(\)](#)

#### Parameters

*thread* Thread to set the GDT entry for.

*desc* Pointer to GDT descriptors.

*size* Size of the descriptors in bytes (multiple of 8).

*entry\_number\_start* Entry number to start (valid values: 0-2).

*utcb* UTCB of the caller.

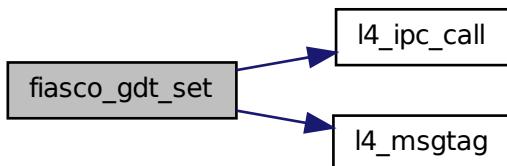
#### Returns

System call error

Definition at line 35 of file [segment.h](#).

References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_mshtag\(\)](#), [L4\\_PROTO\\_THREAD](#), and [l4\\_msg\\_regs\\_t::mr](#).

Here is the call graph for this function:



### 9.36.2.12 `unsigned fiasco_gdt_get_entry_offset ( l4_cap_idx_t thread, l4_utcb_t * utcb ) [inline]`

Return the offset of the entry in the GDT.

#### Parameters

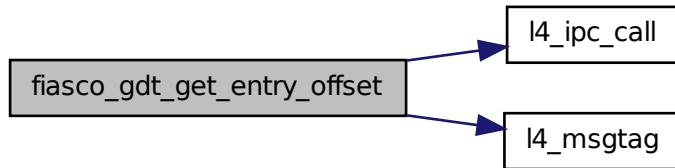
*thread* Thread to get info from.

*utcb* UTCB of the caller.

Definition at line 107 of file [segment.h](#).

References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_mshtag\(\)](#), [L4\\_PROTO\\_THREAD](#), and [l4\\_msg\\_regs\\_t::mr](#).

Here is the call graph for this function:



## 9.37 Fiasco real time scheduling extensions

Real time scheduling extension for the Fiasco [L4](#) implementation.

Collaboration diagram for Fiasco real time scheduling extensions:



## Functions

- int `l4_rt_add_time slice (l4_threadid_t dest, int prio, int time)`  
*Add a time slice for periodic execution.*
- int `l4_rt_change_time slice (l4_threadid_t dest, int id, int prio, int time)`

*Change a time slice for periodic execution.*

- int **l4\_rt\_begin\_strictly\_periodic** (l4\_threadid\_t dest, l4\_kernel\_clock\_t clock)  
*Start strictly periodic execution.*
- int **l4\_rt\_begin\_minimal\_periodic** (l4\_threadid\_t dest, l4\_kernel\_clock\_t clock)  
*Start periodic execution with minimal inter-release times.*
- int **l4\_rt\_end\_periodic** (l4\_threadid\_t dest)  
*Stop periodic execution.*
- int **l4\_rt\_remove** (l4\_threadid\_t dest)  
*Remove all reservation scheduling contexts*  
*This function removes all the scheduling contexts that were set up so far for the given thread.*
- void **l4\_rt\_set\_period** (l4\_threadid\_t dest, l4\_kernel\_clock\_t clock)  
*Set the length of the period*  
*This function sets the length of the period for periodic execution.*
- int **l4\_rt\_next\_reservation** (unsigned id, l4\_kernel\_clock\_t \*clock)  
*activate the next time slice (scheduling context)*
- int **l4\_rt\_next\_period** (void)  
*Wait for the next period, skipping all unused time slices.*
- l4\_threadid\_t **l4\_preemption\_id** (l4\_threadid\_t id)  
*Return the preemption id of a thread.*
- l4\_threadid\_t **l4\_next\_period\_id** (l4\_threadid\_t id)  
*Return thread-id that flags waiting for the next period.*
- int **l4\_rt\_generic** (l4\_threadid\_t dest, l4\_sched\_param\_t param, l4\_kernel\_clock\_t clock)  
*Generic real-time setup function.*

### 9.37.1 Detailed Description

Real time scheduling extension for the Fiasco L4 implementation.

### 9.37.2 Function Documentation

#### 9.37.2.1 int l4\_rt\_add\_time\_slice ( l4\_threadid\_t dest, int prio, int time ) [inline]

Add a time slice for periodic execution.

##### Parameters

- dest* thread to add the time slice to
- prio* priority of the time slice
- time* length of the time slice in microseconds

**Return values****0** OK**-1** Error, one of:

- dest does not exist
- insufficient MCP (old or new prio>MCP),
- dest running in periodic mode or transitioning to
- time quantum 0 or infinite

**9.37.2.2 int l4\_rt\_change\_time\_slice ( l4\_threadid\_t dest, int id, int prio, int time ) [inline]**

Change a time slice for periodic execution.

**Parameters***dest* thread whose timing parameters are to change*id* number of the time-slice to change (rt start at 1)*prio* new priority of the time slice*time* new length of the time slice in microseconds, 0: don't change.**Return values****0** OK**-1** Error, one of:

- dest does not exist
- insufficient MCP (old or new prio>MCP),
- time slice does not exist

This function modifies the priority and optionally the length of an existing time slice of a thread. When calling this function while the time slice is active, the effect may be delayed till the next period.

This function can be called as soon as the denoted time slice was added with [l4\\_rt\\_add\\_time\\_slice\(\)](#). Thus, the thread may have started periodic execution already, but it needs not.

**9.37.2.3 int l4\_rt\_begin\_strictly\_periodic ( l4\_threadid\_t dest, l4\_kernel\_clock\_t clock ) [inline]**

Start strictly periodic execution.

**Parameters***dest* thread that starts periodic execution*clock* absolute time to start.**Return values****0** OK**-1** Error, one of:

- dest does not exist
- insufficient MCP (old or new prio>MCP),

- dest running in periodic mode or transitioning to

Call this function to start the periodic execution after setting up the time slices using `l4_rt_add_time slice()` and `l4_rt_set_period()`.

By the time specified in clock thread dest must wait for the next period, e.g. by using `l4_rt_next_period()` or some other IPC with the L4\_RT\_NEXT\_PERIOD flag enabled. Otherwise the transition to periodic mode fails.

Definition at line 81 of file `rt_sched-impl.h`.

References `l4_rt_generic()`.

Here is the call graph for this function:



### 9.37.2.4 int l4\_rt\_begin\_minimal\_periodic ( l4\_threadid\_t dest, l4\_kernel\_clock\_t clock ) [inline]

Start periodic execution with minimal inter-release times.

#### Parameters

*dest* thread that starts periodic execution

*clock* absolute time to start.

#### Return values

**0** OK

**-1** Error, one of:

- dest does not exist
- insufficient MCP (old or new prio>MCP),
- dest running in periodic mode or transitioning to

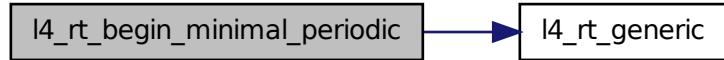
Call this function to start the periodic execution after setting up the time slices using `l4_rt_add_time slice()` and `l4_rt_set_period()`.

By the time specified in clock thread dest must wait for the next period, e.g. by using `l4_rt_next_period()` or some other IPC with the L4\_RT\_NEXT\_PERIOD flag enabled. Otherwise the transition to periodic mode fails.

Definition at line 91 of file `rt_sched-impl.h`.

References `l4_rt_generic()`.

Here is the call graph for this function:



### 9.37.2.5 int l4\_rt\_end\_periodic ( l4\_threadid\_t dest ) [inline]

Stop periodic execution.

#### Parameters

*dest* thread that stops periodic execution

#### Return values

**0** OK

**-1** Error, one of:

- dest does not exist
- insufficient MCP (old or new prio>MCP),
- dest not running in periodic mode and not transitioning to

This function aborts the periodic execution of thread dest. Thread dest returns to conventional scheduling then.

Definition at line 101 of file [rt\\_sched-impl.h](#).

References [l4\\_rt\\_generic\(\)](#).

Here is the call graph for this function:



### 9.37.2.6 int l4\_rt\_remove ( l4\_threadid\_t dest ) [inline]

Remove all reservation scheduling contexts

This function removes all the scheduling contexts that were set up so far for the given thread.

## Parameters

*dest* thread the scheduling contexts should be removed from

## Return values

*0* OK

*-1* Error, one of:

- dest does not exist
- insufficient MCP
- dest running in periodic mode or transitioning to

Definition at line 110 of file [rt\\_sched-impl.h](#).

References [l4\\_rt\\_generic\(\)](#).

Here is the call graph for this function:



### 9.37.2.7 void l4\_rt\_set\_period ( l4\_threadid\_t dest, l4\_kernel\_clock\_t clock ) [inline]

Set the length of the period

This function sets the length of the period for periodic execution.

## Parameters

*dest* destination thread

*clock* period length in microseconds. Will be rounded up by the kernel according to the timer granularity.

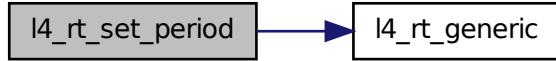
## Returns

This function always succeeds.

Definition at line 119 of file [rt\\_sched-impl.h](#).

References [l4\\_rt\\_generic\(\)](#).

Here is the call graph for this function:



### 9.37.2.8 int l4\_rt\_next\_reservation ( unsigned id, l4\_kernel\_clock\_t \* clock ) [inline]

activate the next time slice (scheduling context)

#### Parameters

*id* The ID of the time slice we think we are on (current time slice)

*clock* pointer to a l4\_kernel\_clock\_t variable

#### Return values

*0* OK, \*clock contains the remaining time of the time slice

*-1* Error, id did not match current time slice

Definition at line 129 of file [rt\\_sched-impl.h](#).

### 9.37.2.9 int l4\_rt\_next\_period ( void ) [inline]

Wait for the next period, skipping all unused time slices.

#### Return values

*0* OK

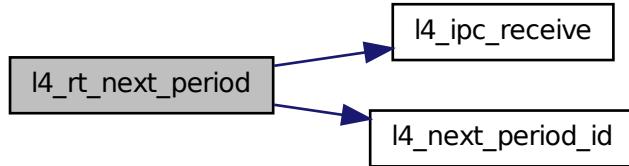
*!0* IPC Error.

< 0 receive and send timeout

Definition at line 155 of file [rt\\_sched-impl.h](#).

References [L4\\_IPC\\_BOTH\\_TIMEOUT\\_0](#), [l4\\_ipc\\_receive\(\)](#), and [l4\\_next\\_period\\_id\(\)](#).

Here is the call graph for this function:



### 9.37.2.10 `l4_threadid_t l4_preemption_id ( l4_threadid_t id ) [inline]`

Return the preemption id of a thread.

#### Parameters

*id* thread

#### Returns

thread-id of the (virtual) preemption IPC sender

Definition at line 303 of file [rt\\_sched-proto.h](#).

### 9.37.2.11 `l4_threadid_t l4_next_period_id ( l4_threadid_t id ) [inline]`

Return thread-id that flags waiting for the next period.

#### Parameters

*id* original thread-id

#### Returns

modified id, to be used in an IPC, waiting for the next period.

Definition at line 310 of file [rt\\_sched-proto.h](#).

Referenced by [l4\\_rt\\_next\\_period\(\)](#).

Here is the caller graph for this function:



### 9.37.2.12 int l4\_rt\_generic( l4\_threadid\_t dest, l4\_sched\_param\_t param, l4\_kernel\_clock\_t clock ) [inline]

Generic real-time setup function.

#### Parameters

*dest* destination thread

*param* scheduling parameter

*clock* clock parameter

#### Return values

0 OK

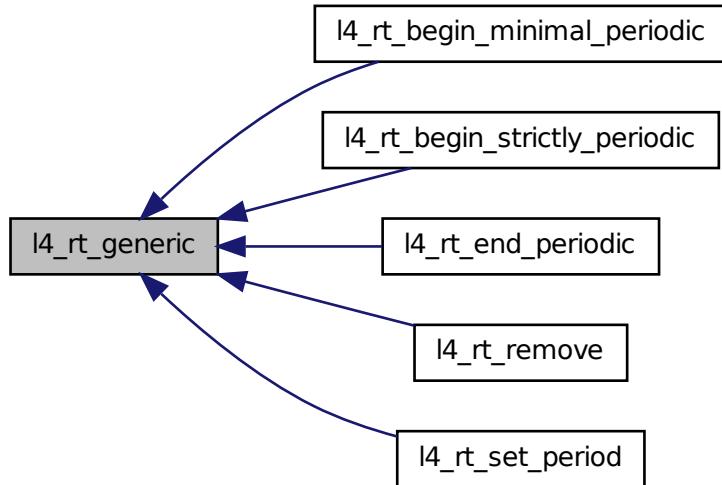
-1 Error.

This function is not meant to be used directly, it is merely used by others.

Definition at line 30 of file [rt\\_sched-impl.h](#).

Referenced by [l4\\_rt\\_begin\\_minimal\\_periodic\(\)](#), [l4\\_rt\\_begin\\_strictly\\_periodic\(\)](#), [l4\\_rt\\_end\\_periodic\(\)](#), [l4\\_rt\\_remove\(\)](#), and [l4\\_rt\\_set\\_period\(\)](#).

Here is the caller graph for this function:



## 9.38 Flex pages

Flex-page related API.

Collaboration diagram for Flex pages:



## Data Structures

- union `l4_fpage_t`  
`L4` flexpage type.
- struct `l4_snd_fpage_t`  
`Send-flex-page types.`

## Enumerations

- enum `l4_fpage_consts` {
   
`L4_FPAGE_RIGHTS_SHIFT` = 0, `L4_FPAGE_TYPE_SHIFT` = 4, `L4_FPAGE_SIZE_SHIFT` = 6,  
`L4_FPAGE_ADDR_SHIFT` = 12,  
`L4_FPAGE_RIGHTS_BITS` = 4, `L4_FPAGE_TYPE_BITS` = 2, `L4_FPAGE_SIZE_BITS` = 6, `L4_FPAGE_ADDR_BITS` = `L4_MWORD_BITS` - `L4_FPAGE_ADDR_SHIFT` }

*L4 flexpage structure.*

- enum { `L4_WHOLE_ADDRESS_SPACE` = 63 }

*Constants for flexpages.*

- enum `L4_fpage_rights` { `L4_FPAGE_RO` = 4, `L4_FPAGE_RW` = 6 }

*Flex-page rights.*

- enum `L4_cap_fpage_rights` { `L4_CAP_FPAGE_R` = 0x4, `L4_CAP_FPAGE_RO` = 0x4, `L4_CAP_FPAGE_RW` = 0x5 }

*Cap-flex-page rights.*

- enum `L4_fpage_type`

*Flex-page type.*

- enum `L4_fpage_control`

*Flex-page map control flags.*

- enum `L4_obj_fpage_ctl`

*Flex-page map control for capabilities (snd\_base).*

- enum `l4_fpage_cacheability_opt_t` { `L4_FPAGE_CACHE_OPT` = 0x1, `L4_FPAGE_CACHEABLE` = 0x3, `L4_FPAGE_BUFFERABLE` = 0x5, `L4_FPAGE_UNCACHEABLE` = 0x1 }

*Flex-page cacheability option.*

- enum { `L4_WHOLE_IOADDRESS_SPACE` = 16, `L4_IOPORT_MAX` = (`1L << L4_WHOLE_IOADDRESS_SPACE`) }

*Special constants for IO flex pages.*

## Functions

- `l4_fpage_t l4_fpage` (unsigned long address, unsigned int size, unsigned char rights) `L4_NOTHROW`  
*Create a memory flex page.*
- `l4_fpage_t l4_fpage_all` (void) `L4_NOTHROW`  
*Get a flex page, describing all address spaces at once.*
- `l4_fpage_t l4_fpage_invalid` (void) `L4_NOTHROW`  
*Get an invalid flex page.*
- `l4_fpage_t l4_iofpage` (unsigned long port, unsigned int size) `L4_NOTHROW`

*Create an IO-port flex page.*

- `l4_fpage_t l4_obj_fpage (l4_cap_idx_t obj, unsigned int order, unsigned char rights) L4_NOTHROW`

*Create a kernel-object flex page.*

- `int l4_is_fpage_writable (l4_fpage_t fp) L4_NOTHROW`

*Test if the flex page is writable.*

- `unsigned l4_fpage_rights (l4_fpage_t f) L4_NOTHROW`

*Return rights from a flex page.*

- `unsigned l4_fpage_type (l4_fpage_t f) L4_NOTHROW`

*Return type from a flex page.*

- `unsigned l4_fpage_size (l4_fpage_t f) L4_NOTHROW`

*Return size from a flex page.*

- `unsigned long l4_fpage_page (l4_fpage_t f) L4_NOTHROW`

*Return page from a flex page.*

- `l4_fpage_t l4_fpage_set_rights (l4_fpage_t src, unsigned char new_rights) L4_NOTHROW`

*Set new right in a flex page.*

- `int l4_fpage_contains (l4_fpage_t fpage, l4_addr_t addr, unsigned size) L4_NOTHROW`

*Test whether a given range is completely within an fpage.*

- `unsigned char l4_fpage_max_order (unsigned char order, l4_addr_t addr, l4_addr_t min_addr, l4_addr_t max_addr, l4_addr_t hotspot L4_DEFAULT_PARAM(0))`

*Determine maximum flex page size of a region.*

### 9.38.1 Detailed Description

Flex-page related API. A flex page is a page with a variable size, that can describe memory, IO-Ports (IA32 only), and sets of kernel objects.

A flex page describes an always size aligned region of an address space. The size is given in a log2 scale. This means the size in elements (bytes for memory, ports for IO-Ports, and capabilities for kernel objects) is always a power of two.

A flex page also carries type and access right information for the described region. The type information selects the address space in which the flex page is valid. Access rights have a meaning depending on the specific address space (type).

There exists a special type for defining *receive windows* or for the `l4_task_unmap()` method, that can be used to describe all address spaces (all types) with a single flex page.

### 9.38.2 Enumeration Type Documentation

#### 9.38.2.1 enum l4\_fpage\_consts

`L4` flexpage structure.

**Enumerator:**

- L4\_FPAGE\_RIGHTS\_SHIFT* Access permissions shift.
- L4\_FPAGE\_TYPE\_SHIFT* Flexpage type shift (memory, IO port, obj...).
- L4\_FPAGE\_SIZE\_SHIFT* Flexpage size shift (log2-based).
- L4\_FPAGE\_ADDR\_SHIFT* Page address shift.
- L4\_FPAGE\_RIGHTS\_BITS* Access permissions size.
- L4\_FPAGE\_TYPE\_BITS* Flexpage type size (memory, IO port, obj...).
- L4\_FPAGE\_SIZE\_BITS* Flexpage size size (log2-based).
- L4\_FPAGE\_ADDR\_BITS* Page address size.

Definition at line 55 of file [\\_\\_l4\\_fpage.h](#).

### 9.38.2.2 anonymous enum

Constants for flexpages.

**Enumerator:**

- L4\_WHOLE\_ADDRESS\_SPACE* Whole address space size.

Definition at line 86 of file [\\_\\_l4\\_fpage.h](#).

### 9.38.2.3 enum L4\_fpage\_rights

Flex-page rights.

**Enumerator:**

- L4\_FPAGE\_RO* Read-only flex page.
- L4\_FPAGE\_RW* Read-write flex page.

Definition at line 104 of file [\\_\\_l4\\_fpage.h](#).

### 9.38.2.4 enum L4\_cap\_fpage\_rights

Cap-flex-page rights.

**Enumerator:**

- L4\_CAP\_FPAGE\_R* Read-only cap.
- L4\_CAP\_FPAGE\_RO* Read-only cap.
- L4\_CAP\_FPAGE\_RW* Read-write cap.

Definition at line 117 of file [\\_\\_l4\\_fpage.h](#).

### 9.38.2.5 enum l4\_fpage\_cacheability\_opt\_t

Flex-page cacheability option.

**Enumerator:**

*L4\_FPAGE\_CACHE\_OPT* Enable the cacheability option in a send flex page.

*L4\_FPAGE\_CACHEABLE* Cacheability option to enable caches for the mapping.

*L4\_FPAGE\_BUFFERABLE* Cacheability option to enable buffered writes for the mapping.

*L4\_FPAGE\_UNCACHEABLE* Cacheability option to disable caching for the mapping.

Definition at line 164 of file [\\_l4\\_fpage.h](#).

### 9.38.2.6 anonymous enum

Special constants for IO flex pages.

**Enumerator:**

*L4\_WHOLE\_IOPADDRESS\_SPACE* Whole I/O address space size.

*L4\_IOPORT\_MAX* Maximum I/O port address.

Definition at line 183 of file [\\_l4\\_fpage.h](#).

## 9.38.3 Function Documentation

### 9.38.3.1 l4\_fpage\_t l4\_fpage ( *unsigned long address*, *unsigned int size*, *unsigned char rights* ) [inline]

Create a memory flex page.

**Parameters**

*address* Flex-page start address

*size* Flex-page size (log2), *L4\_WHOLE\_ADDRESS\_SPACE* to specify the whole address space (with *address* 0)

*rights* Access rights, see [l4\\_fpage\\_rights](#)

**Returns**

Memory flex page

Definition at line 453 of file [\\_l4\\_fpage.h](#).

### 9.38.3.2 l4\_fpage\_t l4\_fpage\_all ( *void* ) [inline]

Get a flex page, describing all address spaces at once.

**Returns**

Special *all-spaces* flex page.

Definition at line 471 of file [\\_l4\\_fpage.h](#).

References [L4\\_WHOLE\\_ADDRESS\\_SPACE](#).

**9.38.3.3 l4\_fpage\_t l4\_fpage\_invalid ( void ) [inline]**

Get an invalid flex page.

**Returns**

Special *invalid* flex page.

Definition at line 477 of file [\\_\\_l4\\_fpage.h](#).

**9.38.3.4 l4\_fpage\_t l4\_iofpage ( unsigned long port, unsigned int size ) [inline]**

Create an IO-port flex page.

**Parameters**

*port* I/O-flex-page port base

*size* I/O-flex-page size, [L4\\_WHOLE\\_IOADDRESS\\_SPACE](#) to specify the whole I/O address space (with *port* 0)

**Returns**

I/O flex page

Definition at line 459 of file [\\_\\_l4\\_fpage.h](#).

References [L4\\_FPAGE\\_ADDR\\_SHIFT](#), and [L4\\_FPAGE\\_RW](#).

**9.38.3.5 l4\_fpage\_t l4\_obj\_fpage ( l4\_cap\_idx\_t obj, unsigned int order, unsigned char rights ) [inline]**

Create a kernel-object flex page.

**Parameters**

*obj* Base capability selector.

*order* Log2 size (number of capabilities).

*rights* Access rights

**Returns**

Flex page for a set of kernel objects.

Definition at line 465 of file [\\_\\_l4\\_fpage.h](#).

Referenced by [L4::Cap\\_base::fpage\(\)](#), and [l4re\\_util\\_cap\\_release\(\)](#).

Here is the caller graph for this function:



### 9.38.3.6 int l4\_is\_fpage\_writable ( l4\_fpage\_t fp ) [inline]

Test if the flex page is writable.

#### Parameters

*fp* Flex page.

#### Returns

$\neq 0$  if flex page is writable, 0 if not

Definition at line 484 of file [\\_\\_l4\\_fpage.h](#).

References [l4\\_fpage\\_rights\(\)](#).

Here is the call graph for this function:



### 9.38.3.7 unsigned l4\_fpage\_rights ( l4\_fpage\_t f ) [inline]

Return rights from a flex page.

#### Parameters

*f* Flex page

#### Returns

Size part of the given flex page.

Definition at line 403 of file [\\_l4\\_fpage.h](#).

Referenced by [l4\\_is\\_fpage\\_writable\(\)](#).

Here is the caller graph for this function:



### 9.38.3.8 `unsigned l4_fpage_type ( l4_fpage_t f ) [inline]`

Return type from a flex page.

#### Parameters

*f* Flex page

#### Returns

Type part of the given flex page.

Definition at line 409 of file [\\_l4\\_fpage.h](#).

### 9.38.3.9 `unsigned l4_fpage_size ( l4_fpage_t f ) [inline]`

Return size from a flex page.

#### Parameters

*f* Flex page

#### Returns

Size part of the given flex page.

Definition at line 415 of file [\\_l4\\_fpage.h](#).

### 9.38.3.10 `unsigned long l4_fpage_page ( l4_fpage_t f ) [inline]`

Return page from a flex page.

#### Parameters

*f* Flex page

**Returns**

Page part of the given flex page.

Definition at line 421 of file [\\_\\_l4\\_fpage.h](#).

Referenced by [l4\\_fpage\\_contains\(\)](#).

Here is the caller graph for this function:



### **9.38.3.11 `l4_fpage_t l4_fpage_set_rights ( l4_fpage_t src, unsigned char new_rights ) [inline]`**

Set new right in a flex page.

**Parameters**

*src* Flex page

*new\_rights* New rights

**Returns**

Modified flex page with new rights.

Definition at line 444 of file [\\_\\_l4\\_fpage.h](#).

References [l4\\_fpage\\_t::raw](#).

### **9.38.3.12 `int l4_fpage_contains ( l4_fpage_t fpage, l4_addr_t addr, unsigned size ) [inline]`**

Test whether a given range is completely within an fpage.

**Parameters**

*fpage* Flex page

*addr* Address

*size* Size of range in log2.

Definition at line 503 of file [\\_\\_l4\\_fpage.h](#).

References [l4\\_fpage\\_page\(\)](#).

Here is the call graph for this function:



**9.38.3.13 `unsigned char l4_fpage_max_order ( unsigned char order, l4_addr_t addr, l4_addr_t min_addr, l4_addr_t max_addr, l4_addr_t hotspot L4_DEFAULT_PARAM0 ) [inline]`**

Determine maximum flex page size of a region.

#### Parameters

***order*** Order value to start with (e.g. for memory L4\_LOG2\_PAGESIZE would be used)

***addr*** Address to be covered by the flex page.

***min\_addr*** Start of region / minimal address (including).

***max\_addr*** End of region / maximal address (excluding).

***hotspot*** (Optional) hot spot.

#### Returns

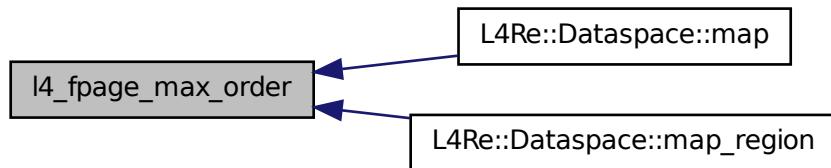
Maximum order (log2-size) possible.

#### Note

The start address of the flex-page can be determined with l4\_trunc\_size(addr, returnvalue)

Referenced by [L4Re::Dataspace::map\(\)](#), and [L4Re::Dataspace::map\\_region\(\)](#).

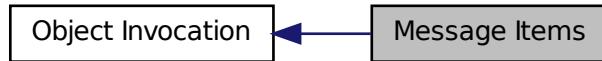
Here is the caller graph for this function:



## 9.39 Message Items

Message item related functions.

Collaboration diagram for Message Items:



### Enumerations

- enum `l4_msg_item_consts_t` {
   
`L4_ITEM_MAP` = 8, `L4_ITEM_CONT` = 1, `L4_MAP_ITEM_GRANT` = 2, `L4_MAP_ITEM_MAP` = 0,
   
`L4_RCV_ITEM_SINGLE_CAP` = `L4_ITEM_MAP` | 2, `L4_RCV_ITEM_LOCAL_ID` = 4 }
   
*Constants for message items.*

### Functions

- `l4_umword_t l4_map_control` (`l4_umword_t` spot, unsigned char cache, unsigned grant) L4\_NOTHROW
   
*Create the first word for a map item for the memory space.*
- `l4_umword_t l4_map_obj_control` (`l4_umword_t` spot, unsigned grant) L4\_NOTHROW
   
*Create the first word for a map item for the object space.*

#### 9.39.1 Detailed Description

Message item related functions. Message items are typed items that can be transferred via IPC operations. Message items are also used to specify receive windows for typed items to be received. Message items are placed in the message registers (MRs) of the UTCB of the sending thread. Receive items are placed in the buffer registers (BRs) of the UTCB of the receiving thread.

Message items are usually two-word data structures. The first word denotes the type of the message item (for example a memory flex-page, io flex-page or object flex-page) and the second word contains information depending on the type. There is actually one exception that is a small (one word) receive buffer item for a single capability.

### 9.39.2 Enumeration Type Documentation

#### 9.39.2.1 enum l4\_msg\_item\_consts\_t

Constants for message items.

**Enumerator:**

**L4\_ITEM\_MAP** Identify a message item as *map item*.

**L4\_ITEM\_CONT** Denote that the following item shall be put into the same receive item as this one.

**L4\_MAP\_ITEM\_GRANT** Flag as *grant* instead of *map* operation.

**L4\_MAP\_ITEM\_MAP** Flag as usual *map* operation.

**L4\_RCV\_ITEM\_SINGLE\_CAP** Mark the receive buffer to be a small receive item that describes a buffer for a single capability.

**L4\_RCV\_ITEM\_LOCAL\_ID** The receiver requests to receive a local ID instead of a mapping whenever possible.

Definition at line 193 of file [consts.h](#).

### 9.39.3 Function Documentation

#### 9.39.3.1 l4\_umword\_t l4\_map\_control( l4\_umword\_t spot, unsigned char cache, unsigned grant ) [inline]

Create the first word for a map item for the memory space.

**Parameters**

**spot** Hot spot address, used to determine what is actually mapped when send and receive flex page have differing sizes.

**cache** Cacheability hints for memory flex pages. See [Cacheability options](#)

**grant** Indicates if it is a map or a grant item.

**Returns**

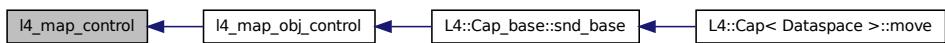
The value to be used as first word in a map item for memory.

Definition at line 490 of file [\\_\\_l4\\_fpage.h](#).

References [L4\\_ITEM\\_MAP](#).

Referenced by [l4\\_map\\_obj\\_control\(\)](#).

Here is the caller graph for this function:



### 9.39.3.2 `l4_umword_t l4_map_obj_control( l4_umword_t spot, unsigned grant ) [inline]`

Create the first word for a map item for the object space.

#### Parameters

*spot* Hot spot address, used to determine what is actually mapped when send and receive flex pages have different size.

*grant* Indicates if it is a map item or a grant item.

#### Returns

The value to be used as first word in a map item for kernel objects or IO-ports.

Definition at line 497 of file [\\_\\_l4\\_fpage.h](#).

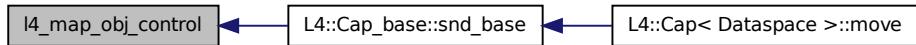
References [l4\\_map\\_control\(\)](#).

Referenced by [L4::Cap\\_base::snd\\_base\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



## 9.40 Timeouts

All kinds of timeouts and time related functions.

Collaboration diagram for Timeouts:



## Data Structures

- struct `l4_timeout_s`  
*Basic timeout specification.*
- union `l4_timeout_t`  
*Timeout pair.*

## Defines

- #define `L4_IPC_TIMEOUT_0` ((`l4_timeout_s`){0x0400})  
*Timeout constants.*
- #define `L4_IPC_TIMEOUT_NEVER` ((`l4_timeout_s`){0})  
*never timeout*
- #define `L4_IPC_NEVER_INITIALIZER` {0}  
*never timeout, init*
- #define `L4_IPC_NEVER` ((`l4_timeout_t`){0})  
*never timeout*
- #define `L4_IPC_RECV_TIMEOUT_0` ((`l4_timeout_t`){0x000000400})  
*0 receive timeout*
- #define `L4_IPC_SEND_TIMEOUT_0` ((`l4_timeout_t`){0x040000000})  
*0 send timeout*
- #define `L4_IPC_BOTH_TIMEOUT_0` ((`l4_timeout_t`){0x040000400})  
*0 receive and send timeout*

## Typedefs

- typedef struct `l4_timeout_s` `l4_timeout_s`  
*Basic timeout specification.*

- `typedef union l4_timeout_t l4_timeout_t`

*Timeout pair.*

## Enumerations

- `enum l4_timeout_abs_validity`

*Intervals of validity for absolute timeouts  
Times are actually  $2^x$  values (e.g.*

## Functions

- `l4_timeout_s l4_timeout_rel` (`unsigned man, unsigned exp`) L4\_NOTHROW  
*Get relative timeout consisting of mantissa and exponent.*
- `l4_timeout_t l4_ipc_timeout` (`unsigned snd_man, unsigned snd_exp, unsigned rcv_man, unsigned rcv_exp`) L4\_NOTHROW  
*Convert explicit timeout values to `l4_timeout_t` type.*
- `l4_timeout_t l4_timeout` (`l4_timeout_s snd, l4_timeout_s rcv`) L4\_NOTHROW  
*Combine send and receive timeout in a timeout.*
- `void l4_snd_timeout` (`l4_timeout_s snd, l4_timeout_t *to`) L4\_NOTHROW  
*Set send timeout in given to timeout.*
- `void l4_rcv_timeout` (`l4_timeout_s rcv, l4_timeout_t *to`) L4\_NOTHROW  
*Set receive timeout in given to timeout.*
- `l4_kernel_clock_t l4_timeout_rel_get` (`l4_timeout_s to`) L4\_NOTHROW  
*Get clock value of out timeout.*
- `unsigned l4_timeout_is_absolute` (`l4_timeout_s to`) L4\_NOTHROW  
*Return whether the given timeout is absolute or not.*
- `l4_kernel_clock_t l4_timeout_get` (`l4_kernel_clock_t cur, l4_timeout_s to`) L4\_NOTHROW  
*Get clock value for a clock + a timeout.*
- `l4_timeout_s l4_timeout_abs` (`l4_kernel_clock_t pint, int br`) throw ()  
*Set an absolute timeout.*

### 9.40.1 Detailed Description

All kinds of timeouts and time related functions.

## 9.40.2 Define Documentation

### 9.40.2.1 `#define L4_IPC_TIMEOUT_0 ((l4_timeout_s){0x0400})`

Timeout constants.

0 timeout

Definition at line [77](#) of file `__timeout.h`.

## 9.40.3 Typedef Documentation

### 9.40.3.1 `typedef struct l4_timeout_s l4_timeout_s`

Basic timeout specification.

Basically a floating point number with 10 bits mantissa and 5 bits exponent ( $t = m \cdot 2^e$ ).

The timeout can also specify an absolute point in time (bit 16 == 1).

### 9.40.3.2 `typedef union l4_timeout_t l4_timeout_t`

Timeout pair.

For IPC there are usually a send and a receive timeout. So this structure contains a pair of timeouts.

## 9.40.4 Enumeration Type Documentation

### 9.40.4.1 `enum l4_timeout_abs_validity`

Intervals of validity for absolute timeouts

Times are actually  $2^x$  values (e.g.

2ms -> 2048 $\mu$ s)

Definition at line [92](#) of file `__timeout.h`.

## 9.40.5 Function Documentation

### 9.40.5.1 `l4_timeout_s l4_timeout_rel( unsigned man, unsigned exp ) [inline]`

Get relative timeout consisting of mantissa and exponent.

#### Parameters

*man* Mantissa of timeout

*exp* Exponent of timeout

#### Returns

timeout value

Definition at line [245](#) of file `__timeout.h`.

#### 9.40.5.2 `l4_timeout_t l4_ipc_timeout ( unsigned snd_man, unsigned snd_exp, unsigned rcv_man, unsigned rcv_exp ) [inline]`

Convert explicit timeout values to `l4_timeout_t` type.

##### Parameters

- snd\_man* Mantissa of send timeout.
- snd\_exp* Exponent of send timeout.
- rcv\_man* Mantissa of receive timeout.
- rcv\_exp* Exponent of receive timeout.

Definition at line 210 of file `__timeout.h`.

References `l4_timeout_t::p`, `l4_timeout_t::recv`, `l4_timeout_t::snd`, and `l4_timeout_s::t`.

#### 9.40.5.3 `l4_timeout_t l4_timeout ( l4_timeout_s snd, l4_timeout_s recv ) [inline]`

Combine send and receive timeout in a timeout.

##### Parameters

- snd* Send timeout
- recv* Receive timeout

##### Returns

`L4` timeout

Definition at line 221 of file `__timeout.h`.

References `l4_timeout_t::p`, `l4_timeout_t::recv`, and `l4_timeout_t::snd`.

#### 9.40.5.4 `void l4_snd_timeout ( l4_timeout_s snd, l4_timeout_t * to ) [inline]`

Set send timeout in given to timeout.

##### Parameters

- snd* Send timeout

##### Return values

*to* `L4` timeout

Definition at line 231 of file `__timeout.h`.

#### 9.40.5.5 `void l4_rcv_timeout ( l4_timeout_s recv, l4_timeout_t * to ) [inline]`

Set receive timeout in given to timeout.

##### Parameters

- recv* Receive timeout

**Return values**

*to* L4 timeout

Definition at line 238 of file [\\_\\_timeout.h](#).

**9.40.5.6 l4\_kernel\_clock\_t l4\_timeout\_rel\_get( l4\_timeout\_s *to* ) [inline]**

Get clock value of out timeout.

**Parameters**

*to* L4 timeout

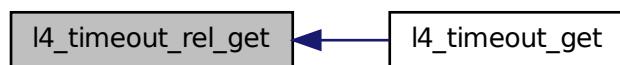
**Returns**

Clock value

Definition at line 252 of file [\\_\\_timeout.h](#).

Referenced by [l4\\_timeout\\_get\(\)](#).

Here is the caller graph for this function:

**9.40.5.7 unsigned l4\_timeout\_is\_absolute( l4\_timeout\_s *to* ) [inline]**

Return whether the given timeout is absolute or not.

**Parameters**

*to* L4 timeout

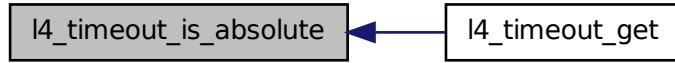
**Returns**

$\neq 0$  if absolute, 0 if relative

Definition at line 261 of file [\\_\\_timeout.h](#).

Referenced by [l4\\_timeout\\_get\(\)](#).

Here is the caller graph for this function:



#### 9.40.5.8 `l4_kernel_clock_t l4_timeout_get ( l4_kernel_clock_t cur, l4_timeout_s to ) [inline]`

Get clock value for a clock + a timeout.

##### Parameters

*cur* Clock value  
*to* L4 timeout

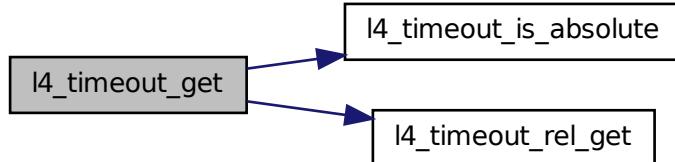
##### Returns

Clock sum

Definition at line 268 of file `__timeout.h`.

References `l4_timeout_is_absolute()`, and `l4_timeout_rel_get()`.

Here is the call graph for this function:



#### 9.40.5.9 `l4_timeout_s l4_timeout_abs ( l4_kernel_clock_t pint, int br ) throw () [inline]`

Set an absolute timeout.

##### Parameters

*pint* Point in time in clocks

*br* The buffer register the timeout shall be placed in. (

#### Note

On 32bit architectures the timeout needs two consecutive buffers.)

The absolute timeout value will be placed into the buffer register *br* of the current thread.

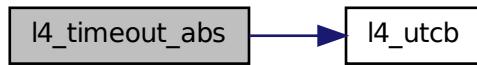
#### Returns

timeout value

Definition at line 357 of file [utcb.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



## 9.41 VM API for SVM

Virtual machine API for SVM.

Collaboration diagram for VM API for SVM:



## Data Structures

- struct [l4\\_vm\\_svm\\_vmcb\\_control\\_area](#)  
*VMCB structure for SVM VMs.*
- struct [l4\\_vm\\_svm\\_vmcb\\_state\\_save\\_area\\_seg](#)  
*State save area segment selector struct.*
- struct [l4\\_vm\\_svm\\_vmcb\\_state\\_save\\_area](#)

*State save area structure for SVM VMs.*

- struct [l4\\_vm\\_svm\\_vmcb\\_t](#)

*Control structure for SVM VMs.*

## Typedefs

- typedef struct [l4\\_vm\\_svm\\_vmcb\\_control\\_area](#) [l4\\_vm\\_svm\\_vmcb\\_control\\_area\\_t](#)

*VMCB structure for SVM VMs.*

- typedef struct [l4\\_vm\\_svm\\_vmcb\\_state\\_save\\_area\\_seg](#) [l4\\_vm\\_svm\\_vmcb\\_state\\_save\\_area\\_seg\\_t](#)

*State save area segment selector struct.*

- typedef struct [l4\\_vm\\_svm\\_vmcb\\_state\\_save\\_area](#) [l4\\_vm\\_svm\\_vmcb\\_state\\_save\\_area\\_t](#)

*State save area structure for SVM VMs.*

- typedef struct [l4\\_vm\\_svm\\_vmcb\\_t](#) [l4\\_vm\\_svm\\_vmcb\\_t](#)

*Control structure for SVM VMs.*

### 9.41.1 Detailed Description

Virtual machine API for SVM.

## 9.42 VM API for VMX

Virtual machine API for VMX.

Collaboration diagram for VM API for VMX:



## Enumerations

- enum

*Additional VMCS fields.*

## Functions

- `unsigned l4_vm_vmx_field_len (unsigned field)`  
*Return length in bytes of a VMCS field.*
- `void * l4_vm_vmx_field_ptr (void *vmcs, unsigned field)`  
*Get pointer into VMCS.*

### 9.42.1 Detailed Description

Virtual machine API for VMX.

### 9.42.2 Function Documentation

#### 9.42.2.1 `unsigned l4_vm_vmx_field_len ( unsigned field ) [inline]`

Return length in bytes of a VMCS field.

##### Parameters

*field* Field number.

##### Returns

Width of field in bytes.

Definition at line 71 of file `__vm-vmx.h`.

Referenced by `l4_vm_vmx_field_ptr()`.

Here is the caller graph for this function:



#### 9.42.2.2 `void * l4_vm_vmx_field_ptr ( void * vmcs, unsigned field ) [inline]`

Get pointer into VMCS.

##### Parameters

*vmcs* Pointer to VMCS buffer.

*field* Field number.

**Pointer** to field in the VMCS.

Definition at line 79 of file \_\_vm-vmx.h.

References [l4\\_vm\\_vmx\\_field\\_len\(\)](#).

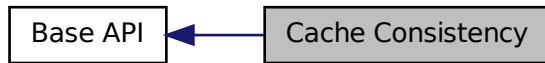
Here is the call graph for this function:



## 9.43 Cache Consistency

Various functions for cache consistency.

Collaboration diagram for Cache Consistency:



## Functions

- void [l4\\_cache\\_clean\\_data](#) (unsigned long start, unsigned long end) throw ()
   
*Cache clean a range in D-cache.*
- void [l4\\_cache\\_flush\\_data](#) (unsigned long start, unsigned long end) throw ()
   
*Cache flush a range.*
- void [l4\\_cache\\_inv\\_data](#) (unsigned long start, unsigned long end) throw ()
   
*Cache invalidate a range.*
- void [l4\\_cache\\_coherent](#) (unsigned long start, unsigned long end) throw ()
   
*Make memory coherent between I-cache and D-cache.*
- void [l4\\_cache\\_dma\\_coherent](#) (unsigned long start, unsigned long end) throw ()
   
*Make memory coherent for use with external memory.*

- void **l4\_cache\_dma\_coherent\_full** (void) throw()  
*Make memory coherent for use with external memory.*

### 9.43.1 Detailed Description

Various functions for cache consistency. #include <l4/sys/cache.h>

### 9.43.2 Function Documentation

#### 9.43.2.1 void **l4\_cache\_clean\_data** ( **unsigned long start, unsigned long end** ) throw() [**inline**]

Cache clean a range in D-cache.

##### Parameters

*start* Start of range (inclusive)  
*end* End of range (exclusive)

##### Examples:

[examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#).

Definition at line 81 of file [cache.h](#).

#### 9.43.2.2 void **l4\_cache\_flush\_data** ( **unsigned long start, unsigned long end** ) throw() [**inline**]

Cache flush a range.

##### Parameters

*start* Start of range (inclusive)  
*end* End of range (exclusive)

Definition at line 88 of file [cache.h](#).

#### 9.43.2.3 void **l4\_cache\_inv\_data** ( **unsigned long start, unsigned long end** ) throw() [**inline**]

Cache invalidate a range.

##### Parameters

*start* Start of range (inclusive)  
*end* End of range (exclusive)

Definition at line 95 of file [cache.h](#).

#### 9.43.2.4 void l4\_cache\_coherent ( `unsigned long start, unsigned long end` ) throw () [inline]

Make memory coherent between I-cache and D-cache.

##### Parameters

- start* Start of range (inclusive)
- end* End of range (exclusive)

Definition at line 102 of file [cache.h](#).

#### 9.43.2.5 void l4\_cache\_dma\_coherent ( `unsigned long start, unsigned long end` ) throw () [inline]

Make memory coherent for use with external memory.

##### Parameters

- start* Start of range (inclusive)
- end* End of range (exclusive)

Definition at line 109 of file [cache.h](#).

## 9.44 Memory related

Memory related constants, data types and functions.

Collaboration diagram for Memory related:



### Defines

- #define [L4\\_PAGESIZE](#)  
*Minimal page size (in bytes).*
- #define [L4\\_PAGEMASK](#)  
*Mask for the page number.*
- #define [L4\\_LOG2\\_PAGESIZE](#)  
*Number of bits used for page offset.*

- `#define L4_SUPERPAGESIZE`  
*Size of a large page.*
- `#define L4_SUPERPAGEMASK`  
*Mask for the number of a large page.*
- `#define L4_LOG2_SUPERPAGESIZE`  
*Number of bits used as offset for a large page.*
- `#define L4_INVALID_PTR ((void*)L4_INVALID_ADDR)`  
*Invalid address as pointer type.*
- `#define L4_PAGESHIFT 12`  
*Size of a page, log2-based.*
- `#define L4_SUPERPAGESHIFT 20`  
*Size of a large page, log2-based.*
- `#define L4_PAGESHIFT 12`  
*Size of a page, log2-based.*
- `#define L4_SUPERPAGESHIFT 21`  
*Size of a large page, log2-based.*
- `#define L4_PAGESHIFT 12`  
*Size of a page log2-based.*
- `#define L4_SUPERPAGESHIFT 22`  
*Size of a large page log2-based.*

## Enumerations

- `enum l4_addr_consts_t { L4_INVALID_ADDR = ~0UL }`  
*Address related constants.*

## Functions

- `l4_addr_t l4_trunc_page (l4_addr_t address) throw ()`  
*Round an address down to the next lower page boundary.*
- `l4_addr_t l4_trunc_size (l4_addr_t address, unsigned char bits) throw ()`  
*Round an address down to the next lower flex page with size bits.*
- `l4_addr_t l4_round_page (l4_addr_t address) throw ()`  
*Round address up to the next page.*
- `l4_addr_t l4_round_size (l4_addr_t address, unsigned char bits) throw ()`  
*Round address up to the next flex page with bits size.*

### 9.44.1 Detailed Description

Memory related constants, data types and functions.

### 9.44.2 Define Documentation

#### 9.44.2.1 #define L4\_PAGEMASK

Mask for the page number.

##### Note

The most significant bits are set.

Definition at line [285](#) of file `consts.h`.

#### 9.44.2.2 #define L4\_LOG2\_PAGESIZE

Number of bits used for page offset.

Size of page in log2.

Definition at line [294](#) of file `consts.h`.

Referenced by [L4Re::Dataspace::map\(\)](#).

#### 9.44.2.3 #define L4\_SUPERPAGESIZE

Size of a large page.

A large page is a *super page* on IA32 or a *section* on ARM.

Definition at line [303](#) of file `consts.h`.

#### 9.44.2.4 #define L4\_SUPERPAGEMASK

Mask for the number of a large page.

##### Note

The most significant bits are set.

Definition at line [312](#) of file `consts.h`.

#### 9.44.2.5 #define L4\_LOG2\_SUPERPAGESIZE

Number of bits used as offset for a large page.

Size of large page in log2

Definition at line [320](#) of file `consts.h`.

### 9.44.3 Enumeration Type Documentation

#### 9.44.3.1 enum l4\_addr\_consts\_t

Address related constants.

**Enumerator:**

*L4\_INVALID\_ADDR* Invalid address.

Definition at line 368 of file [consts.h](#).

### 9.44.4 Function Documentation

#### 9.44.4.1 l4\_addr\_t l4\_trunc\_page ( l4\_addr\_t *address* ) throw () [inline]

Round an address down to the next lower page boundary.

**Parameters**

*address* The address to round.

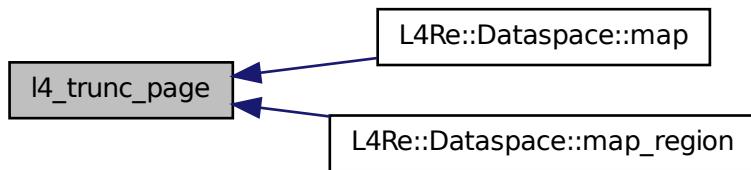
**Examples:**

[examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), and [examples/libs/l4re/c/ma+rm.c](#).

Definition at line 329 of file [consts.h](#).

Referenced by [L4Re::Dataspace::map\(\)](#), and [L4Re::Dataspace::map\\_region\(\)](#).

Here is the caller graph for this function:



#### 9.44.4.2 l4\_addr\_t l4\_trunc\_size ( l4\_addr\_t *address*, unsigned char *bits* ) throw () [inline]

Round an address down to the next lower flex page with size *bits*.

**Parameters**

*address* The address to round.

*bits* The size of the flex page ( $\log_2$ ).

Definition at line 340 of file [consts.h](#).

Referenced by [L4Re::Dataspace::map\\_region\(\)](#).

Here is the caller graph for this function:



#### 9.44.4.3 l4\_addr\_t l4\_round\_page ( l4\_addr\_t address ) throw () [inline]

Round address up to the next page.

##### Parameters

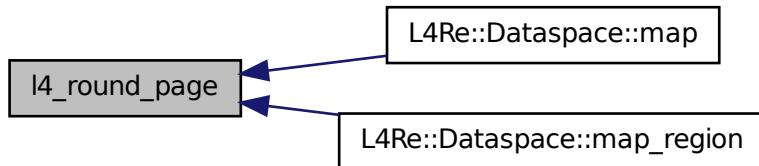
*address* The address to round up.

Definition at line 350 of file [consts.h](#).

References [L4\\_PAGESIZE](#).

Referenced by [L4Re::Dataspace::map\(\)](#), and [L4Re::Dataspace::map\\_region\(\)](#).

Here is the caller graph for this function:



#### 9.44.4.4 l4\_addr\_t l4\_round\_size ( l4\_addr\_t address, unsigned char bits ) throw () [inline]

Round address up to the next flex page with *bits* size.

### Parameters

*address* The address to round up to the next flex page.

*bits* The size of the flex page (log2).

Definition at line 361 of file `consts.h`.

Referenced by [L4Re::Dataspace::map\\_region\(\)](#).

Here is the caller graph for this function:



## 9.45 Kernel Debugger

Kernel debugger related functionality.

Collaboration diagram for Kernel Debugger:



## Data Structures

- class [L4::Debugger](#)

*Debugger interface.*

## Defines

- #define [enter\\_kdebug\(text\)](#)

*Enter L4 kernel debugger.*

- #define [asm\\_enter\\_kdebug\(text\)](#)

*Enter L4 kernel debugger (plain assembler version).*

- `#define kd_display(text)`  
*Show message with L4 kernel debugger, but do not enter debugger.*
- `#define ko(c)`  
*Output character with L4 kernel debugger.*
- `#define enter_kdebug(text)`  
*Enter L4 kernel debugger.*
- `#define asm_enter_kdebug(text)`  
*Enter L4 kernel debugger (plain assembler version).*
- `#define kd_display(text)`  
*Show message with L4 kernel debugger, but do not enter debugger.*
- `#define ko(c)`  
*Output character with L4 kernel debugger.*

## Functions

- `l4_msgtag_t l4_debugger_set_object_name (l4_cap_idx_t cap, const char *name) throw ()`  
*The string name of kernel object.*
- `unsigned long l4_debugger_global_id (l4_cap_idx_t cap) throw ()`  
*Get the globally unique ID of the object behind a capability.*
- `unsigned long l4_debugger_kobj_to_id (l4_cap_idx_t cap, l4_addr_t kobjp) throw ()`  
*Get the globally unique ID of the object behind the kobject pointer.*
- `void outchar (char c) throw ()`  
*Print character.*
- `void outstring (const char *text) throw ()`  
*Print character string.*
- `void outnstring (char const *text, unsigned len) throw ()`  
*Print character string.*
- `void outhex32 (int number) throw ()`  
*Print 32 bit number (hexadecimal).*
- `void outhex20 (int number) throw ()`  
*Print 20 bit number (hexadecimal).*
- `void outhex16 (int number) throw ()`  
*Print 16 bit number (hexadecimal).*
- `void outhex12 (int number) throw ()`

*Print 12 bit number (hexadecimal).*

- void [outhex8](#) (int number) throw ()

*Print 8 bit number (hexadecimal).*

- void [outdec](#) (int number) throw ()

*Print number (decimal).*

- char [l4kd\\_inchar](#) (void) throw ()

*Read character from console, non blocking.*

### 9.45.1 Detailed Description

Kernel debugger related functionality.

#### Attention

This API is subject to change!

This is a debugging facility, any call to any function might be invalid. Do not rely on it in any real code.

```
#include <L4/sys/debugger.h>
```

### 9.45.2 Define Documentation

#### 9.45.2.1 #define enter\_kdebug( *text* )

##### Value:

```
asm(\n    "int    $3    \n\t"\n    "jmp    1f    \n\t"\n    ".ascii  \"\"text  \"\"\n\t"\n    "1:        \n\t"\n)
```

Enter [L4](#) kernel debugger.

##### Parameters

*text* Text to be shown at kernel debugger prompt

##### Examples:

```
examples/sys/singlestep/main.c.
```

Definition at line [41](#) of file [kdebug.h](#).

Referenced by [fiasco\\_tbuf\\_get\\_status\\_phys\(\)](#).

### 9.45.2.2 #define asm\_enter\_kdebug( *text* )

#### Value:

```
"int $3 \n\t\
"jmp 1f \n\t\
".ascii \"text \"\n\t\
": \n\t"
```

Enter L4 kernel debugger (plain assembler version).

#### Parameters

*text* Text to be shown at kernel debugger prompt

Definition at line 63 of file [kdebug.h](#).

### 9.45.2.3 #define kd\_display( *text* )

#### Value:

```
asm(\n    "int $3 \n\t\
    "nop \n\t\
    "jmp 1f \n\t\
    ".ascii \"text \"\n\t\
    ": \n\t\
)
```

Show message with L4 kernel debugger, but do not enter debugger.

#### Parameters

*text* Text to be shown

Definition at line 76 of file [kdebug.h](#).

### 9.45.2.4 #define ko( *c* )

#### Value:

```
asm(\n    \
    "int $3 \n\t" \
    "cmpb %0,%al \n\t" \
    : /* No output */ \
    : "N" (c) \
)
```

Output character with L4 kernel debugger.

#### Parameters

*c* Character to be shown

Definition at line 92 of file [kdebug.h](#).

**9.45.2.5 #define enter\_kdebug( *text* )****Value:**

```
asm(\n    "int $3 \n\t"\n    "jmp 1f \n\t"\n    ".ascii \"\"text \"\"\n\t"\n    "1: \n\t"\n)
```

Enter L4 kernel debugger.

**Parameters***text* Text to be shown at kernel debugger promptDefinition at line 41 of file [kdebug.h](#).**9.45.2.6 #define asm\_enter\_kdebug( *text* )****Value:**

```
"int $3 \n\t"\n    "jmp 1f \n\t"\n    ".ascii \"\"text \"\"\n\t"\n    "1: \n\t"
```

Enter L4 kernel debugger (plain assembler version).

**Parameters***text* Text to be shown at kernel debugger promptDefinition at line 63 of file [kdebug.h](#).**9.45.2.7 #define kd\_display( *text* )****Value:**

```
asm(\n    "int $3 \n\t"\n    "nop \n\t"\n    "jmp 1f \n\t"\n    ".ascii \"\"text \"\"\n\t"\n    "1: \n\t"\n)
```

Show message with L4 kernel debugger, but do not enter debugger.

**Parameters***text* Text to be shownDefinition at line 76 of file [kdebug.h](#).

### 9.45.2.8 #define ko( c )

**Value:**

```
asm( "int $3\n\t\"cmpb %0,%al\n\t\"\\"
      : /* No output */
      : "N" (c) \\
      )
```

Output character with L4 kernel debugger.

**Parameters**

*c* Character to be shown

Definition at line 92 of file [kdebug.h](#).

## 9.45.3 Function Documentation

### 9.45.3.1 l4\_mshtag\_t l4\_debugger\_set\_object\_name ( l4\_cap\_idx\_t cap, const char \* name ) throw() [[inline](#)]

The string name of kernel object.

**Parameters**

*cap* Capability

*name* Name

This is a debugging facility, the call might be invalid.

**Examples:**

[examples/sys/aliens/main.c](#).

### 9.45.3.2 unsigned long l4\_debugger\_global\_id ( l4\_cap\_idx\_t cap ) throw() [[inline](#)]

Get the globally unique ID of the object behind a capability.

**Parameters**

*cap* Capability

**Returns**

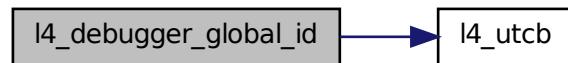
~0UL on non-valid capability, ID otherwise

This is a debugging facility, the call might be invalid.

Definition at line 297 of file [debugger.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.45.3.3 `unsigned long l4_debugger_kobj_to_id ( l4_cap_idx_t cap, l4_addr_t kobjp ) throw () [inline]`

Get the globally unique ID of the object behind the kobject pointer.

#### Parameters

*cap* Capability

*kobjp* Kobject pointer

#### Returns

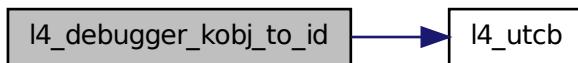
~0UL on non-valid capability or invalid kobject pointer, ID otherwise

This is a debugging facility, the call might be invalid.

Definition at line 303 of file [debugger.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.45.3.4 `void outchar ( char c ) throw () [inline]`

Print character.

#### Parameters

*c* Character

**9.45.3.5 void outstring( const char \* *text* ) throw() [inline]**

Print character string.

**Parameters**

*text* Character string

*text* String

**Examples:**

[examples/sys/aliens/main.c](#).

**9.45.3.6 void outnstring( char const \* *text*, unsigned *len* ) throw() [inline]**

Print character string.

**Parameters**

*text* Character string

*len* Number of characters

*text* String

*len* Number of characters

**Examples:**

[examples/sys/aliens/main.c](#).

**9.45.3.7 void outhex32( int *number* ) throw() [inline]**

Print 32 bit number (hexadecimal).

**Parameters**

*number* 32 bit number

**9.45.3.8 void outhex20( int *number* ) throw() [inline]**

Print 20 bit number (hexadecimal).

**Parameters**

*number* 20 bit number

**9.45.3.9 void outhex16( int *number* ) throw() [inline]**

Print 16 bit number (hexadecimal).

**Parameters**

*number* 16 bit number

**9.45.3.10 void outhex12 ( int *number* ) throw () [inline]**

Print 12 bit number (hexadecimal).

**Parameters**

*number* 12 bit number

**9.45.3.11 void outhex8 ( int *number* ) throw () [inline]**

Print 8 bit number (hexadecimal).

**Parameters**

*number* 8 bit number

**9.45.3.12 void outdec ( int *number* ) throw () [inline]**

Print number (decimal).

**Parameters**

*number* Number

**9.45.3.13 char l4kd\_inchar ( void ) throw () [inline]**

Read character from console, non blocking.

**Returns**

Input character, -1 if no character to read

## 9.46 Error codes

Common error codes.

Collaboration diagram for Error codes:



## Enumerations

- enum l4\_error\_code\_t {
   
*L4\_EOK* = 0, *L4\_EPERM* = 1, *L4\_ENOENT* = 2, *L4\_EIO* = 5,
   
*L4\_EAGAIN* = 11, *L4\_ENOMEM* = 12, *L4\_EACCESS* = 13, *L4\_EBUSY* = 16,
   
*L4\_EEXIST* = 17, *L4\_ENODEV* = 19, *L4\_EINVAL* = 22, *L4\_ERANGE* = 34,
   
*L4\_ENAMETOOLONG* = 36, *L4\_ENOSYS* = 38, *L4\_EBADPROTO* = 39, *L4\_EADDRNOTAVAIL* = 99,
   
*L4\_ERRNOMAX* = 100, *L4\_ENOREPLY* = 1000, *L4\_EIPC\_LO* = 2000, *L4\_EIPC\_HI* = 2000 + 0x1f
 }
- L4 error codes.*

### 9.46.1 Detailed Description

Common error codes. #include <L4/sys/err.h>

### 9.46.2 Enumeration Type Documentation

#### 9.46.2.1 enum l4\_error\_code\_t

*L4* error codes.

Those error codes are used by both the kernel and the user programs.

Enumerator:

- L4\_EOK* Ok.
- L4\_EPERM* No permission.
- L4\_ENOENT* No such entity.
- L4\_EIO* I/O error.
- L4\_EAGAIN* Try again.
- L4\_ENOMEM* No memory.
- L4\_EACCESS* Permission denied.
- L4\_EBUSY* Object currently busy, try later.
- L4\_EEXIST* Already exists.
- L4\_ENODEV* No such thing.
- L4\_EINVAL* Invalid argument.
- L4\_ERANGE* Range error.
- L4\_ENAMETOOLONG* Name too long.
- L4\_ENOSYS* No sys.
- L4\_EBADPROTO* Unsupported protocol.
- L4\_EADDRNOTAVAIL* Address not available.
- L4\_ERRNOMAX* Maximum error value.
- L4\_ENOREPLY* No reply.
- L4\_EIPC\_LO* Communication error-range low.
- L4\_EIPC\_HI* Communication error-range high.

Definition at line 41 of file [err.h](#).

## 9.47 Factory

A factory is used to create all kinds of kernel objects.

Collaboration diagram for Factory:



## Data Structures

- class [L4::Factory](#)  
*C++ L4 Factory, to create all kinds of kernel objects.*

## Functions

- [l4\\_mshtag\\_t l4\\_factory\\_create\\_task](#) (*l4\_cap\_idx\_t* factory, *l4\_cap\_idx\_t* target\_cap, *l4\_fpage\_t const utc\_b\_area*) throw ()  
*Create a new task.*
- [l4\\_mshtag\\_t l4\\_factory\\_create\\_thread](#) (*l4\_cap\_idx\_t* factory, *l4\_cap\_idx\_t* target\_cap) throw ()  
*Create a new thread.*
- [l4\\_mshtag\\_t l4\\_factory\\_create\\_factory](#) (*l4\_cap\_idx\_t* factory, *l4\_cap\_idx\_t* target\_cap, *unsigned long limit*) throw ()  
*Create a new factory.*
- [l4\\_mshtag\\_t l4\\_factory\\_create\\_gate](#) (*l4\_cap\_idx\_t* factory, *l4\_cap\_idx\_t* target\_cap, *l4\_cap\_idx\_t thread\_cap, l4\_umword\_t label*) throw ()  
*Create a new IPC gate.*
- [l4\\_mshtag\\_t l4\\_factory\\_create\\_semaphore](#) (*l4\_cap\_idx\_t* factory, *l4\_cap\_idx\_t* target\_cap) throw ()  
*Create a new semaphore.*
- [l4\\_mshtag\\_t l4\\_factory\\_create\\_irq](#) (*l4\_cap\_idx\_t* factory, *l4\_cap\_idx\_t* target\_cap) throw ()  
*Create a new IRQ.*
- [l4\\_mshtag\\_t l4\\_factory\\_create\\_vm](#) (*l4\_cap\_idx\_t* factory, *l4\_cap\_idx\_t* target\_cap) throw ()  
*Create a new virtual machine.*

### 9.47.1 Detailed Description

A factory is used to create all kinds of kernel objects. #include <14/sys/factory.h>

A factory provides the means to create all kinds of kernel objects. The factory is equipped with a limit that limits the amount of kernel memory available for that factory.

#### Note

The limit does not give any guarantee for the amount of available kernel memory.

### 9.47.2 Function Documentation

#### 9.47.2.1 l4\_msntag\_t l4\_factory\_create\_task ( l4\_cap\_idx\_t *factory*, l4\_cap\_idx\_t *target\_cap*, l4\_fpage\_t const *utcba\_area* ) throw () [inline]

Create a new task.

#### Parameters

*factory* Capability selector for factory to use for creation.

*target\_cap* Capability selector for the root capability of the new task.

*utcba\_area* Flexpage that describes the area for the UTCBs of the new task

#### Note

The size of the UTCB area specifies indirectly the maximum number of UTCBs available for this task and cannot be changed afterwards.

#### Returns

Syscall return tag

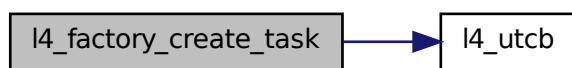
#### See also

[Task](#)

Definition at line 336 of file [factory.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.47.2.2 `l4_msntag_t l4_factory_create_thread( l4_cap_idx_t factory, l4_cap_idx_t target_cap ) throw() [inline]`

Create a new thread.

#### Parameters

*factory* Capability selector for factory to use for creation.

*target\_cap* Capability selector for the root capability of the new thread.

#### Returns

Syscall return tag

#### See also

[Thread](#)

#### Examples:

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 343 of file [factory.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.47.2.3 `l4_msntag_t l4_factory_create_factory( l4_cap_idx_t factory, l4_cap_idx_t target_cap, unsigned long limit ) throw() [inline]`

Create a new factory.

#### Parameters

*factory* Capability selector for factory to use for creation.

*target\_cap* Capability selector for the root capability of the new factory.

*limit* Limit for the new factory in bytes

#### Note

The limit of the new factory is subtracted from the available amount of the factory used for creation.

**Returns**

Syscall return tag

Definition at line 350 of file [factory.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



**9.47.2.4** `l4_msntag_t l4_factory_create_gate( l4_cap_idx_t factory, l4_cap_idx_t target_cap, l4_cap_idx_t thread_cap, l4_umword_t label ) throw() [inline]`

Create a new IPC gate.

**Parameters**

*factory* Capability selector for factory to use for creation.

*target\_cap* Capability selector for the root capability of the new IPC gate.

*thread\_cap* Thread to bind the gate to

*label* Label of the gate

**Returns**

Syscall return tag

**See also**

[IPC-Gate API](#)

Definition at line 358 of file [factory.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



**9.47.2.5 l4\_mshtag\_t l4\_factory\_create\_semaphore ( l4\_cap\_idx\_t *factory*, l4\_cap\_idx\_t *target\_cap* ) throw () [inline]**

Create a new semaphore.

**Parameters**

*factory* Capability selector for factory to use for creation.

*target\_cap* Capability selector for the root capability of the new semaphore.

**Returns**

Syscall return tag

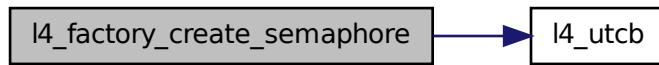
**See also**

[l4\\_sem\\_api](#)

Definition at line 366 of file [factory.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**9.47.2.6 l4\_mshtag\_t l4\_factory\_create\_irq ( l4\_cap\_idx\_t *factory*, l4\_cap\_idx\_t *target\_cap* ) throw () [inline]**

Create a new IRQ.

**Parameters**

*factory* Capability selector for factory to use for creation.

*target\_cap* Capability selector for the root capability of the new IRQ.

**Returns**

Syscall return tag

**See also**

[IRQs](#)

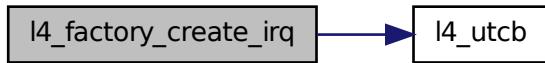
**Examples:**

[examples/sys/isr/main.c](#)

Definition at line 373 of file [factory.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 9.47.2.7 `l4_mshtag_t l4_factory_create_vm ( l4_cap_idx_t factory, l4_cap_idx_t target_cap ) throw () [inline]`

Create a new virtual machine.

##### Parameters

*factory* Capability selector for factory to use for creation.

*target\_cap* Capability selector for the root capability of the new VM.

##### Returns

Syscall return tag

##### See also

[Virtual Machines](#)

Definition at line 380 of file [factory.h](#).

References [l4\\_utcb\(\)](#).

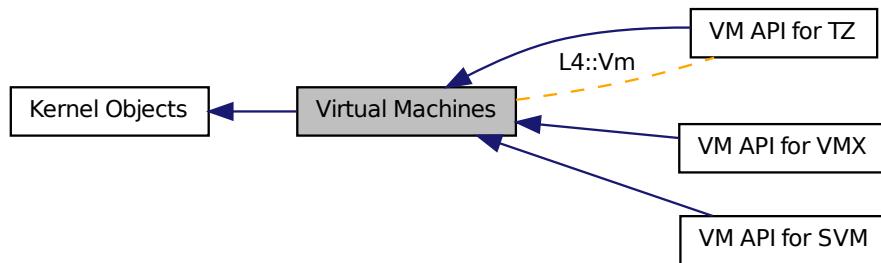
Here is the call graph for this function:



## 9.48 Virtual Machines

Virtual Machine API.

Collaboration diagram for Virtual Machines:



## Data Structures

- class [L4::Vm](#)  
*Virtual machine.*
- [VM API for SVM](#)  
*Virtual machine API for SVM.*
- [VM API for VMX](#)  
*Virtual machine API for VMX.*
- [VM API for TZ](#)  
*Virtual Machine API for ARM TrustZone.*

### 9.48.1 Detailed Description

Virtual Machine API.

## 9.49 Interrupt controller

The ICU class.

Collaboration diagram for Interrupt controller:



## Data Structures

- struct [l4\\_icu\\_info\\_t](#)  
*Info structure for an ICU.*
- class [L4::Icu](#)  
*C++ version of an interrupt controller.*
- class [L4::Icu::Info](#)  
*Info for an ICU.*

## Typedefs

- typedef struct [l4\\_icu\\_info\\_t](#) [l4\\_icu\\_info\\_t](#)  
*Info structure for an ICU.*

## Enumerations

- enum [L4\\_icu\\_flags](#) { [L4\\_ICU\\_FLAG\\_MSI](#) }  
*Flags for IRQ numbers used for the ICU.*

## Functions

- [l4\\_mshtag\\_t l4\\_icu\\_bind](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_cap\\_idx\\_t](#) irq) throw()  
*Bind an interrupt vector of an interrupt controller to an interrupt object.*
- [l4\\_mshtag\\_t l4\\_icu\\_unbind](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_cap\\_idx\\_t](#) irq) throw()  
*Remove binding of an interrupt vector from the interrupt controller object.*
- [l4\\_mshtag\\_t l4\\_icu\\_set\\_mode](#) ([l4\\_cap\\_idx\\_t](#) icu, unsigned irqnum, [l4\\_umword\\_t](#) mode) throw()  
*Set mode of interrupt.*
- [l4\\_mshtag\\_t l4\\_icu\\_info](#) ([l4\\_cap\\_idx\\_t](#) icu, [l4\\_icu\\_info\\_t](#) \*info) throw()

*Get info about capabilites of ICU.*

- `l4_msntag_t l4_icu_msi_info (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *msg) throw ()`  
*Get MSI info about IRQ.*
- `l4_msntag_t l4_icu_unmask (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label, l4_timeout_t to) throw ()`  
*Unmask an IRQ vector.*
- `l4_msntag_t l4_icu_mask (l4_cap_idx_t icu, unsigned irqnum, l4_umword_t *label, l4_timeout_t to) throw ()`  
*Mask an IRQ vector.*

### 9.49.1 Detailed Description

The ICU class. #include <l4/sys/icu.h>

### 9.49.2 Typedef Documentation

#### 9.49.2.1 `typedef struct l4_icu_info_t l4_icu_info_t`

Info structure for an ICU.

This structure contains information about the features of an ICU.

#### See also

[l4\\_icu\\_info\(\)](#).

### 9.49.3 Enumeration Type Documentation

#### 9.49.3.1 `enum L4_icu_flags`

Flags for IRQ numbers used for the ICU.

#### Enumerator:

**`L4_ICU_FLAG_MSI`** Flag to denote that the IRQ is actually an MSI. This flag may be used for `l4_icu_bind()` and `l4_icu_unbind()` functions to denote that the IRQ number is meant to be an MSI.

Definition at line 44 of file [icu.h](#).

### 9.49.4 Function Documentation

#### 9.49.4.1 `l4_msntag_t l4_icu_bind ( l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq ) throw () [inline]`

Bind an interrupt vector of an interrupt controller to an interrupt object.

**Parameters**

*icu* ICU to use.  
*irqnum* IRQ vector at the ICU.  
*irq* IRQ capability to bind the IRQ to.

**Returns**

Syscall return tag

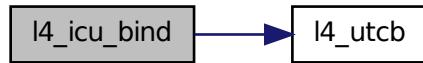
**Examples:**

[examples/sys/isr/main.c](#).

Definition at line 398 of file [icu.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 9.49.4.2 **`l4_mshtag_t l4_icu_unbind( l4_cap_idx_t icu, unsigned irqnum, l4_cap_idx_t irq ) throw() [inline]`**

Remove binding of an interrupt vector from the interrupt controller object.

**Parameters**

*icu* ICU to use.  
*irqnum* IRQ vector at the ICU.  
*irq* IRQ object to remove from the ICU.

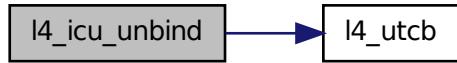
**Returns**

Syscall return tag

Definition at line 402 of file [icu.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 9.49.4.3 `l4_mshtag_t l4_icu_set_mode( l4_cap_idx_t icu, unsigned irqnum, l4_umword_t mode ) throw() [inline]`

Set mode of interrupt.

##### Parameters

*icu* ICU to use.

*irqnum* IRQ vector at the ICU.

*mode* Mode, see L4\_irq\_flow\_type.

##### Returns

Syscall return tag

Definition at line 424 of file [icu.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 9.49.4.4 `l4_mshtag_t l4_icu_info( l4_cap_idx_t icu, l4_icu_info_t * info ) throw() [inline]`

Get info about capabilites of ICU.

##### Parameters

*icu* ICU to use.

*info* Pointer to an info structure to be filled with information.

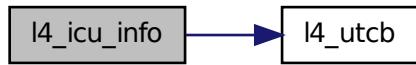
### Returns

Syscall return tag

Definition at line 406 of file [icu.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



**9.49.4.5 `l4_msntag_t l4_icu_msi_info( l4_cap_idx_t icu, unsigned irqnum, l4_umword_t * msg ) throw() [inline]`**

Get MSI info about IRQ.

### Parameters

*icu* ICU to use.

*irqnum* IRQ vector at the ICU.

*msg* Pointer to a word to receive the message that must be used for the PCI devices MSI message.

### Returns

Syscall return tag

Definition at line 410 of file [icu.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 9.49.4.6 `l4_mshtag_t l4_icu_unmask( l4_cap_idx_t icu, unsigned irqnum, l4_umword_t * label, l4_timeout_t to ) throw() [inline]`

Unmask an IRQ vector.

##### Parameters

- icu* ICU to use.
- irqnum* IRQ vector at the ICU.
- label* If non-NULL the function also waits for the next message.
- to* Timeout for message to ICU, if unsure use L4\_IPC\_NEVER.

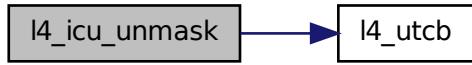
##### Returns

Syscall return tag

Definition at line 414 of file [icu.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 9.49.4.7 `l4_mshtag_t l4_icu_mask( l4_cap_idx_t icu, unsigned irqnum, l4_umword_t * label, l4_timeout_t to ) throw() [inline]`

Mask an IRQ vector.

##### Parameters

- icu* ICU to use.
- irqnum* IRQ vector at the ICU.
- label* If non-NULL the function also waits for the next message.
- to* Timeout for message to ICU, if unsure use L4\_IPC\_NEVER.

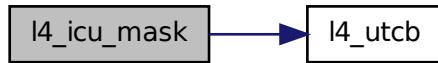
##### Returns

Syscall return tag

Definition at line 419 of file [icu.h](#).

References [l4\\_utcb\(\)](#).

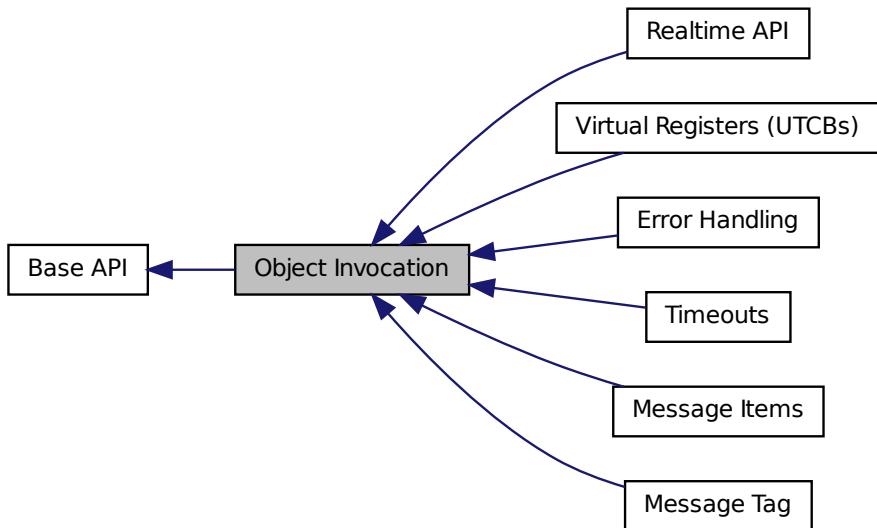
Here is the call graph for this function:



## 9.50 Object Invocation

API for L4 object invocation.

Collaboration diagram for Object Invocation:



## Modules

- [Message Items](#)

*Message item related functions.*

- [Timeouts](#)

*All kinds of timeouts and time related functions.*

- [Error Handling](#)

*Error handling for L4 object invocation.*

- [Realtime API](#)

- [Message Tag](#)

*API related to the message tag data type.*

- [Virtual Registers \(UTCBs\)](#)

*L4 Virtual Registers (UTCB).*

## Files

- file [utcb.h](#)

*UTCB definitions.*

## Enumerations

- enum [l4\\_syscall\\_flags\\_t](#) {

```
L4_SYSF_NONE, L4_SYSF_SEND, L4_SYSF_RECV, L4_SYSF_OPEN_WAIT,
L4_SYSF_REPLY, L4_SYSF_CALL, L4_SYSF_WAIT, L4_SYSF_SEND_AND_WAIT,
L4_SYSF_REPLY_AND_WAIT }
```

*Capability selector flags.*

## Functions

- [l4\\_msgtag\\_t l4\\_ipc\\_send \(l4\\_cap\\_idx\\_t dest, l4\\_utcb\\_t \\*utcb, l4\\_msgtag\\_t tag, l4\\_timeout\\_t timeout\) throw \(\)](#)

*Send a message to an object (do **not** wait for a reply).*

- [l4\\_msgtag\\_t l4\\_ipc\\_wait \(l4\\_utcb\\_t \\*utcb, l4\\_umword\\_t \\*label, l4\\_timeout\\_t timeout\) throw \(\)](#)

*Wait for an incoming message from any possible sender.*

- [l4\\_msgtag\\_t l4\\_ipc\\_receive \(l4\\_cap\\_idx\\_t object, l4\\_utcb\\_t \\*utcb, l4\\_timeout\\_t timeout\) throw \(\)](#)

*Wait for a message from a specific source.*

- [l4\\_msgtag\\_t l4\\_ipc\\_call \(l4\\_cap\\_idx\\_t object, l4\\_utcb\\_t \\*utcb, l4\\_msgtag\\_t tag, l4\\_timeout\\_t timeout\) throw \(\)](#)

*Object call (usual invocation).*

- [l4\\_msgtag\\_t l4\\_ipc\\_reply\\_and\\_wait \(l4\\_utcb\\_t \\*utcb, l4\\_msgtag\\_t tag, l4\\_umword\\_t \\*label, l4\\_timeout\\_t timeout\) throw \(\)](#)

*Reply and wait operation (uses the reply capability).*

- [l4\\_msgtag\\_t l4\\_ipc\\_send\\_and\\_wait \(l4\\_cap\\_idx\\_t dest, l4\\_utcb\\_t \\*utcb, l4\\_msgtag\\_t tag, l4\\_umword\\_t \\*label, l4\\_timeout\\_t timeout\) throw \(\)](#)

*Send a message and do an open wait.*

- `l4_mshtag_t l4_ipc (l4_cap_idx_t dest, l4_utcb_t *utcb, l4_umword_t flags, l4_umword_t slabel, l4_mshtag_t tag, l4_umword_t *rlabel, l4_timeout_t timeout) throw ()`

*Generic L4 object invocation.*

- `l4_mshtag_t l4_ipc_sleep (l4_timeout_t timeout) throw ()`

*Sleep for an amount of time.*

- `int l4_sndfpage_add (l4_fpage_t const snd_fpage, unsigned long snd_base, l4_mshtag_t *tag) throw ()`

*Add a flex-page to be sent to the UTCB.*

### 9.50.1 Detailed Description

API for L4 object invocation. #include <l4/sys/ipc.h>

General abstractions for L4 object invocation. The basic principle is that all objects are denoted by a capability that is accessed via a capability selector (see [Capabilities](#) ).

This set of functions is common to all kinds of objects provided by the L4 micro kernel. The concrete semantics of an invocation depends on the object that shall be invoked.

Objects may be invoked in various ways, the most common way is to use a *call* operation ([l4\\_ipc\\_call\(\)](#)). However, there are a lot more flavours available that have a semantics depending on the object.

#### See also

[IPC-Gate API](#)

### 9.50.2 Enumeration Type Documentation

#### 9.50.2.1 enum l4\_syscall\_flags\_t

Capability selector flags.

These flags determine the concrete operation when a kernel object is invoked.

##### Enumerator:

**L4\_SYSF\_NONE** Default flags (call to a kernel object). Using this value as flags in the capability selector for an invocation indicates a call (send and wait for a reply).

**L4\_SYSF\_SEND** Send-phase flag. Setting this flag in a capability selector induces a send phase, this means a message is send to the object denoted by the capability. For receive phase see [L4\\_SYSF\\_RECV](#).

**L4\_SYSF\_RECV** Receive-phase flag. Setting this flag in a capability selector induces a receive phase, this means the invoking thread waits for a message from the object denoted by the capability. For a send phase see [L4\\_SYSF\\_SEND](#).

**L4\_SYSF\_OPEN\_WAIT** Open-wait flag. This flag indicates that the receive operation (see [L4\\_SYSF\\_RECV](#)) shall be an *open wait*. *Open wait* means that the invoking thread shall wait for a message from any possible sender and *not* from the sender denoted by the capability.

**L4\_SYSF\_REPLY** Reply flag. This flag indicates that the send phase shall use the in-kernel reply capability instead of the capability denoted by the selector index.

**L4\_SYSF\_CALL** Call flags (combines send and receive). Combines L4\_SYSF\_SEND and L4\_SYSF\_RECV.

**L4\_SYSF\_WAIT** Wait flags (combines receive and open wait). Combines L4\_SYSF\_RECV and L4\_SYSF\_OPEN\_WAIT.

**L4\_SYSF\_SEND\_AND\_WAIT** Send-and-wait flags. Combines L4\_SYSF\_SEND and L4\_SYSF\_WAIT.

**L4\_SYSF\_REPLY\_AND\_WAIT** Reply-and-wait flags. Combines L4\_SYSF\_SEND, L4\_SYSF\_REPLY, and L4\_SYSF\_WAIT.

Definition at line 45 of file [consts.h](#).

### 9.50.3 Function Documentation

#### 9.50.3.1 l4\_mshtag\_t l4\_ipc\_send ( l4\_cap\_idx\_t dest, l4\_utcb\_t \* utcb, l4\_mshtag\_t tag, l4\_timeout\_t timeout ) throw () [inline]

Send a message to an object (do **not** wait for a reply).

##### Parameters

*dest* Capability selector for the destination object.

*utcb* UTCB of the caller.

*tag* Descriptor for the message to be sent.

*timeout* Timeout pair (see [l4\\_timeout\\_t](#)) only send part is relevant.

##### Returns

result tag

A message is sent to the destination object. There is no receive phase included. The invoker continues working after sending the message.

##### Attention

This is a special-purpose message transfer, objects usually support only invocation via [l4\\_ipc\\_call\(\)](#).

##### Examples:

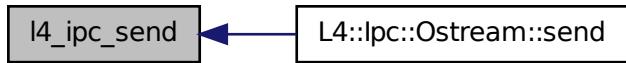
[examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 135 of file [ipc.h](#).

References [L4\\_SYSF\\_SEND](#), and [l4\\_mshtag\\_t::raw](#).

Referenced by [L4::Ipc::Ostream::send\(\)](#).

Here is the caller graph for this function:



### 9.50.3.2 `l4_mshtag_t l4_ipc_wait( l4_utcb_t * utcb, l4_umword_t * label, l4_timeout_t timeout ) throw() [inline]`

Wait for an incoming message from any possible sender.

#### Parameters

*utcb* UTCB of the caller.

#### Return values

*label* Label assigned to the source object (IPC gate or IRQ).

#### Parameters

*timeout* Timeout pair (see [l4\\_timeout\\_t](#), only the receive part is used).

#### Returns

return tag

This operation does an open wait, and therefore needs no capability to denote the possible source of a message. This means the calling thread waits for an incoming message from any possible source. There is no send phase included in this operation.

The usual usage of this function is to call that function when entering a server loop in a user-level server that implements user-level objects, see also [l4\\_ipc\\_reply\\_and\\_wait\(\)](#).

#### Examples:

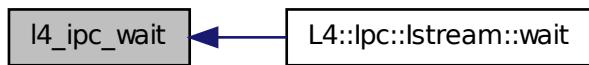
[examples/sys/ ipc/ ipc\\_example.c](#).

Definition at line [168](#) of file [ipc.h](#).

References [L4\\_INVALID\\_CAP](#), [L4\\_SYSF\\_WAIT](#), and [l4\\_mshtag\\_t::raw](#).

Referenced by [L4::Ipc::Istream::wait\(\)](#).

Here is the caller graph for this function:



### 9.50.3.3 `l4_mshtag_t l4_ipc_receive ( l4_cap_idx_t object, l4_utcb_t * utcb, l4_timeout_t timeout ) throw () [inline]`

Wait for a message from a specific source.

#### Parameters

*object* Object to receive a message from.

*timeout* Timeout pair (see [l4\\_timeout\\_t](#), only the receive part matters).

*utcb* UTCB of the caller.

#### Returns

result tag.

This operation waits for a message from the specified object. Messages from other sources are not accepted by this operation. The operation does not include a send phase, this means no message is sent to the object.

#### Note

This operation is usually used to receive messages from a specific IRQ or thread. However, it is not common to use this operation for normal applications.

#### Examples:

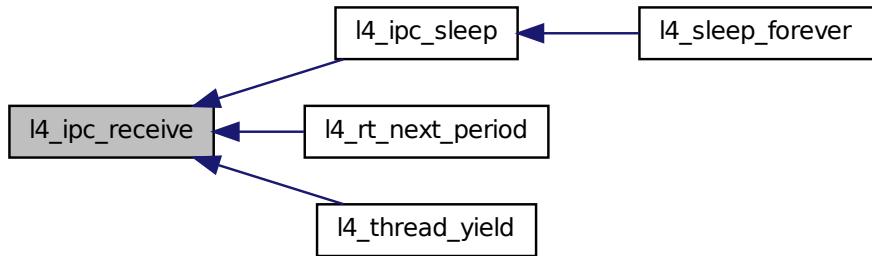
[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line [204](#) of file [ipc.h](#).

References [L4\\_SYSF\\_RECV](#), and [l4\\_mshtag\\_t::raw](#).

Referenced by [l4\\_ipc\\_sleep\(\)](#), [l4\\_rt\\_next\\_period\(\)](#), and [l4\\_thread\\_yield\(\)](#).

Here is the caller graph for this function:



#### 9.50.3.4 `l4_mshtag_t l4_ipc_call ( l4_cap_idx_t object, l4_utcb_t * utcb, l4_mshtag_t tag, l4_timeout_t timeout ) throw () [inline]`

Object call (usual invocation).

##### Parameters

- object* Capability selector for the object to call.
- utcb* UTCB of the caller.
- tag* Message tag to describe the message to be sent.
- timeout* Timeout pair for send an receive phase (see [l4\\_timeout\\_t](#)).

##### Returns

result tag

A message is sent to the object and the invoker waits for a reply from the object. Messages from other sources are not accepted.

##### Note

The send-to-receive transition needs no time, the object can reply with a send timeout of zero.

##### Examples:

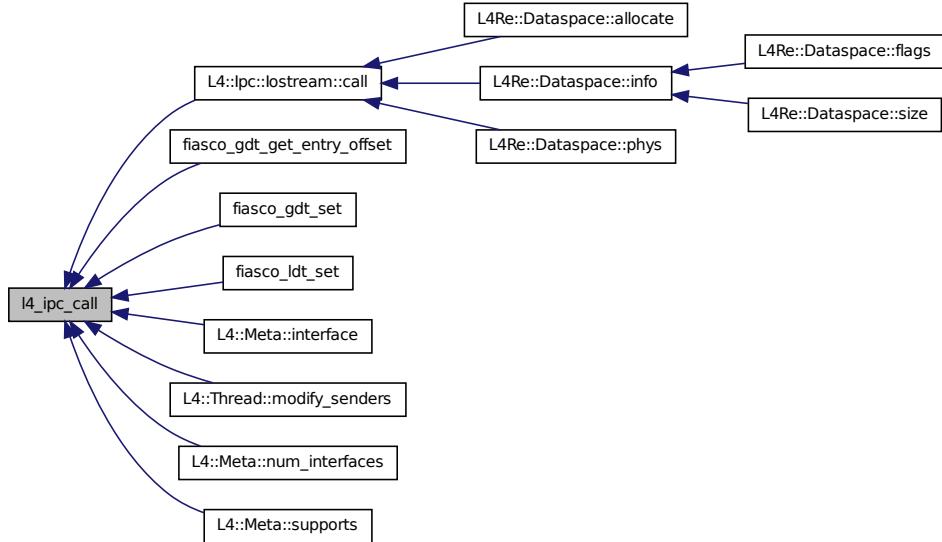
[examples/sys/aliens/main.c](#), [examples/sys/ipc/ipc\\_example.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line 34 of file [ipc.h](#).

References [L4\\_SYSF\\_CALL](#), and [l4\\_mshtag\\_t::raw](#).

Referenced by [L4::Ipc::Iostream::call\(\)](#), [fiasco\\_gdt\\_get\\_entry\\_offset\(\)](#), [fiasco\\_gdt\\_set\(\)](#), [fiasco\\_ldt\\_set\(\)](#), [L4::Meta::interface\(\)](#), [L4::Thread::modify\\_senders\(\)](#), [L4::Meta::num\\_interfaces\(\)](#), and [L4::Meta::supports\(\)](#).

Here is the caller graph for this function:



### 9.50.3.5 `l4_mshtag_t l4_ipc_reply_and_wait( l4_utcb_t * utcb, l4_mshtag_t tag, l4_umword_t * label, l4_timeout_t timeout ) throw() [inline]`

Reply and wait operation (uses the *reply* capability).

#### Parameters

*tag* Describes the message to be sent as reply.

*utcb* UTCB of the caller.

#### Return values

*label* Label assigned to the source object of the received message.

#### Parameters

*timeout* Timeout pair (see [l4\\_timeout\\_t](#)).

#### Returns

result tag

A message is sent to the previous caller using the implicit reply capability. Afterwards the invoking thread waits for a message from any source.

#### Note

This is the standard server operation: it sends a reply to the actual client and waits for the next incoming request, which may come from any other client.

**Examples:**

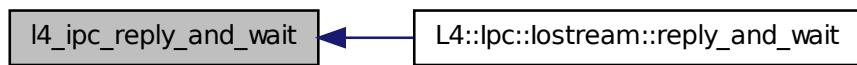
[examples/sys/ ipc/ ipc\\_example.c.](#)

Definition at line [65](#) of file [ipc.h](#).

References [L4\\_INVALID\\_CAP](#), [L4\\_SYSF\\_REPLY\\_AND\\_WAIT](#), and [l4\\_mshtag\\_t::raw](#).

Referenced by [L4::Ipc::Iostream::reply\\_and\\_wait\(\)](#).

Here is the caller graph for this function:



### 9.50.3.6 l4\_mshtag\_t l4\_ipc\_send\_and\_wait( l4\_cap\_idx\_t dest, l4\_utcb\_t \* utcb, l4\_mshtag\_t tag, l4\_umword\_t \* label, l4\_timeout\_t timeout ) throw() [inline]

Send a message and do an open wait.

**Parameters**

*dest* Object to send a message to.

*utcb* UTCB of the caller.

*tag* Describes the message that shall be sent.

**Return values**

*label* Label assigned to the source object of the receive phase.

**Parameters**

*timeout* Timeout pair (see [l4\\_timeout\\_t](#)).

**Returns**

result tag

A message is sent to the destination object and the invoking thread waits for a reply from any source.

**Note**

This is a special-purpose operation and shall not be used in general applications.

Definition at line [100](#) of file [ipc.h](#).

References [L4\\_SYSF\\_SEND\\_AND\\_WAIT](#), and [l4\\_mshtag\\_t::raw](#).

---

**9.50.3.7 `l4_mshtag_t l4_ipc ( l4_cap_idx_t dest, l4_utcb_t * utcb, l4_umword_t flags, l4_umword_t slabel, l4_mshtag_t tag, l4_umword_t * rlabel, l4_timeout_t timeout ) throw () [inline]`**

Generic L4 object invocation.

#### Parameters

*dest* Destination object.

*utcb* UTCB of the caller.

*flags* Invocation flags (see [l4\\_syscall\\_flags\\_t](#)).

*slabel* Send label if applicable (may be seen by the receiver).

*tag* Sending message tag.

#### Return values

*rlabel* Receiving label.

#### Parameters

*timeout* Timeout pair (see [l4\\_timeout\\_t](#)).

#### Returns

return tag

Definition at line 240 of file [ipc.h](#).

References [l4\\_mshtag\\_t::raw](#).

**9.50.3.8 `l4_mshtag_t l4_ipc_sleep ( l4_timeout_t timeout ) throw () [inline]`**

Sleep for an amount of time.

#### Parameters

*timeout* Timeout pair (see [l4\\_timeout\\_t](#), the receive part matters).

#### Returns

error code:

- [L4\\_IPC\\_RETIMEOUT](#): success
- [L4\\_IPC\\_RECANCELED](#) woken up by a different thread ([l4\\_thread\\_ex\\_regs\(\)](#)).

The invoking thread waits until the timeout is expired or the wait was aborted by another thread by [l4\\_thread\\_ex\\_regs\(\)](#).

Definition at line 28 of file [ipc-impl.h](#).

References [L4\\_INVALID\\_CAP](#), and [l4\\_ipc\\_receive\(\)](#).

Referenced by [l4\\_sleep\\_forever\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 9.50.3.9 int l4\_sndfpage\_add ( l4\_fpage\_t const *snd\_fpage*, unsigned long *snd\_base*, l4\_mshtag\_t \* *tag* ) throw () [inline]

Add a flex-page to be sent to the UTCB.

#### Parameters

*snd\_fpage* Flex-page.

*snd\_base* Send base.

*tag* Tag to be modified.

#### Return values

*tag* Modified tag, the number of items will be increased, all other values in the tag will be retained.

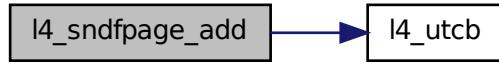
#### Returns

0 on success, negative error code otherwise

Definition at line 486 of file [ipc.h](#).

References [l4\\_utcb\(\)](#).

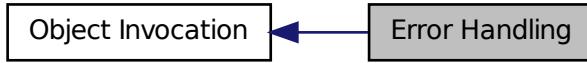
Here is the call graph for this function:



## 9.51 Error Handling

Error handling for L4 object invocation.

Collaboration diagram for Error Handling:



### Enumerations

- enum l4\_ipc\_tcr\_error\_t {
   
L4\_IPC\_ERROR\_MASK = 0x1F, L4\_IPC SND\_ERR\_MASK = 0x01, L4\_IPC\_ENOT\_EXISTENT = 0x04, L4\_IPC RETIMEOUT = 0x03,
   
L4\_IPC\_SETIMEOUT = 0x02, L4\_IPC\_RECANCELED = 0x07, L4\_IPC\_SECANCELED = 0x06,
 L4\_IPC\_REMAPFAILED = 0x11,
   
L4\_IPC\_SEMAPFAILED = 0x10, L4\_IPC RESNDPFTO = 0x0b, L4\_IPC SESNDPFTO = 0x0a,
 L4\_IPC\_RERCVPFTO = 0x0d,
   
L4\_IPC\_SERCVPFTO = 0x0c, L4\_IPC REABORTED = 0x0f, L4\_IPC\_SEABORTED = 0x0e,
 L4\_IPC\_REMSGCUT = 0x09,
   
L4\_IPC\_SEMSGCUT = 0x08 }

*Error codes in the error TCR.*

### Functions

- l4\_umword\_t l4\_ipc\_error (l4\_msntag\_t tag, l4\_utcb\_t \*utcb) throw ()
   
*Get the error code for an object invocation.*

- long `l4_error (l4_msntag_t tag) throw ()`  
*Return error code of a system call return message tag.*
- int `l4_ipc_is_snd_error (l4_utcb_t *utcb) throw ()`  
*Returns whether an error occurred in send phase of an invocation.*
- int `l4_ipc_is_rcv_error (l4_utcb_t *utcb) throw ()`  
*Returns whether an error occurred in receive phase of an invocation.*
- int `l4_ipc_error_code (l4_utcb_t *utcb) throw ()`  
*Get the error condition of the last invocation from the TCR.*

### 9.51.1 Detailed Description

Error handling for L4 object invocation. #include <l4/sys/ ipc.h>

### 9.51.2 Enumeration Type Documentation

#### 9.51.2.1 enum l4\_ipc\_tcr\_error\_t

Error codes in the *error* TCR.

The error codes are accessible via the *error* TCR, see `l4_thread_regs_t.error`.

**Enumerator:**

- `L4_IPC_ERROR_MASK`** Mask for error bits.
- `L4_IPC SND_ERR_MASK`** Send error mask.
- `L4_IPC ENOT_EXISTENT`** Non-existing destination or source.
- `L4_IPC RETIMEOUT`** Timeout during receive operation.
- `L4_IPC SETIMEOUT`** Timeout during send operation.
- `L4_IPC RECANCELED`** Receive operation canceled.
- `L4_IPC SECANCELED`** Send operation canceled.
- `L4_IPC REMAPFAILED`** Map flexpage failed in receive operation.
- `L4_IPC SEMAPFAILED`** Map flexpage failed in send operation.
- `L4_IPC RESNDPFTO`** Send-pagefault timeout in receive operation.
- `L4_IPC SESNDPFTO`** Send-pagefault timeout in send operation.
- `L4_IPC RERCVPFTO`** Receive-pagefault timeout in receive operation.
- `L4_IPC SERCVPFTO`** Receive-pagefault timeout in send operation.
- `L4_IPC REABORTED`** Receive operation aborted.
- `L4_IPC SEABORTED`** Send operation aborted.
- `L4_IPC REMSGCUT`** Cut receive message, due to message buffer is too small.
- `L4_IPC SEMSGCUT`** Cut send message, due to message buffer is too small,

Definition at line 75 of file `ipc.h`.

### 9.51.3 Function Documentation

#### 9.51.3.1 `l4_umword_t l4_ipc_error ( l4_mshtag_t tag, l4_utcb_t * utcb ) throw () [inline]`

Get the error code for an object invocation.

##### Parameters

*tag* Return value of the invocation.

*utcb* UTCB that was used for the invocation.

##### Returns

0 if no error condition is set, error code otherwise (see [l4\\_ipc\\_tcr\\_error\\_t](#)).

##### Examples:

[examples/clntsrv/client.cc](#), [examples/libs/l4re/streammap/client.cc](#), [examples/sys/ipc/ipc\\_example.c](#), [examples/sys/isr/main.c](#), and [examples/sys/start-with-exc/main.c](#).

Definition at line 430 of file [ipc.h](#).

References [l4\\_thread\\_regs\\_t::error](#), and [l4\\_mshtag\\_has\\_error\(\)](#).

Referenced by [L4Re::Util::Event\\_buffer\\_consumer\\_t< PAYLOAD >::process\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 9.51.3.2 `long l4_error ( l4_mshtag_t tag ) throw () [inline]`

Return error code of a system call return message tag.

## Parameters

*tag* System call return message type

## Returns

0 for no error, error number in case of error

## Examples:

[examples/sys/isr/main.c](#), and [examples/sys/migrate/thread\\_migrate.cc](#).

Definition at line 447 of file [ipc.h](#).

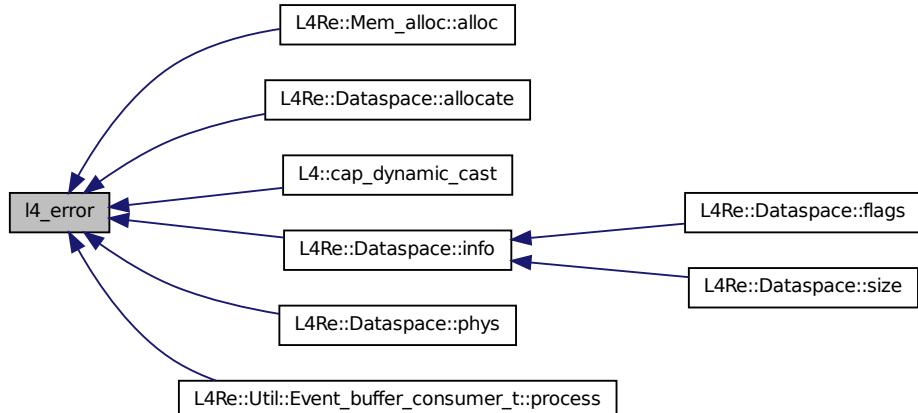
References [l4\\_utcb\(\)](#).

Referenced by [L4Re::Mem\\_alloc::alloc\(\)](#), [L4Re::Dataspace::allocate\(\)](#), [L4::cap\\_dynamic\\_cast\(\)](#), [L4Re::Dataspace::info\(\)](#), [L4Re::Dataspace::phys\(\)](#), and [L4Re::Util::Event\\_buffer\\_consumer\\_t< PAYLOAD >::process\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



**9.51.3.3 int l4\_ipc\_is\_snd\_error ( l4\_utcb\_t \* *utcb* ) throw () [inline]**

Returns whether an error occurred in send phase of an invocation.

**Precondition**

`l4_mshtag_has_error(tag) == true`

**Parameters**

*utcb* UTCB to check.

**Returns**

Boolean value.

Definition at line 453 of file [ipc.h](#).

References [l4\\_thread\\_regs\\_t::error](#).

**9.51.3.4 int l4\_ipc\_is\_rev\_error ( l4\_utcb\_t \* *utcb* ) throw () [inline]**

Returns whether an error occurred in receive phase of an invocation.

**Precondition**

`l4_mshtag_has_error(tag) == true`

**Parameters**

*utcb* UTCB to check.

**Returns**

Boolean value.

Definition at line 456 of file [ipc.h](#).

References [l4\\_thread\\_regs\\_t::error](#).

**9.51.3.5 int l4\_ipc\_error\_code ( l4\_utcb\_t \* *utcb* ) throw () [inline]**

Get the error condition of the last invocation from the TCR.

**Precondition**

`l4_mshtag_has_error(tag) == true`

**Parameters**

*utcb* UTCB to check.

**Returns**

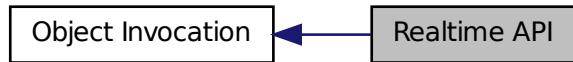
Error condition of type `l4_ipc_tcr_error_t`.

Definition at line 459 of file [ipc.h](#).

References [l4\\_thread\\_regs\\_t::error](#).

## 9.52 Realtime API

Collaboration diagram for Realtime API:



## 9.53 IRQs

The IRQ and IRQ class.

Collaboration diagram for IRQs:



## Data Structures

- class [L4::Irq](#)

*C++ version of an L4 IRQ.*

## Enumerations

- enum [L4\\_irq\\_flow\\_type](#) {
   
    [L4\\_IRQ\\_F\\_NONE](#) = 0, [L4\\_IRQ\\_F\\_LEVEL](#) = 0x2, [L4\\_IRQ\\_F\\_EDGE](#) = 0x0, [L4\\_IRQ\\_F\\_POS](#) = 0x0,
   
    [L4\\_IRQ\\_F\\_NEG](#) = 0x4, [L4\\_IRQ\\_F\\_LEVEL\\_HIGH](#) = 0x3, [L4\\_IRQ\\_F\\_LEVEL\\_LOW](#) = 0x7, [L4\\_IRQ\\_F\\_POS\\_EDGE](#) = 0x1,
   
    [L4\\_IRQ\\_F\\_NEG\\_EDGE](#) = 0x5, [L4\\_IRQ\\_F\\_MASK](#) = 0x7 }
   
*Interrupt flow types.*

## Functions

- `l4_msgtag_t l4_irq_attach (l4_cap_idx_t irq, l4_umword_t label, l4_cap_idx_t thread) throw ()`  
*Attach to an interrupt source.*
- `l4_msgtag_t l4_irq_chain (l4_cap_idx_t irq, l4_umword_t label, l4_cap_idx_t slave) throw ()`  
*Chain an IRQ to another master IRQ source.*
- `l4_msgtag_t l4_irq_detach (l4_cap_idx_t irq) throw ()`  
*Detach from an interrupt source.*
- `l4_msgtag_t l4_irq_trigger (l4_cap_idx_t irq) throw ()`  
*Trigger an IRQ.*
- `l4_msgtag_t l4_irq_receive (l4_cap_idx_t irq, l4_timeout_t to) throw ()`  
*Unmask and wait for specified IRQ.*
- `l4_msgtag_t l4_irq_wait (l4_cap_idx_t irq, l4_umword_t *label, l4_timeout_t to) throw ()`  
*Unmask IRQ and wait for any message.*
- `l4_msgtag_t l4_irq_unmask (l4_cap_idx_t irq) throw ()`  
*Unmask IRQ.*

### 9.53.1 Detailed Description

The IRQ and IRQ class. #include <l4/sys/irq.h>

The IRQ class provides access to abstract interrupts provided by the micro kernel. Interrupts may be hardware interrupts provided by the platform interrupt controller, virtual device interrupts provided by the micro kernel virtual devices (virtual serial or trace buffer), or IRQs (virtual interrupts that can be triggered by user programs).

IRQ objects can be created using a Factory, see [Factory \(l4\\_factory\\_create\\_irq\(\)\)](#).

### 9.53.2 Enumeration Type Documentation

#### 9.53.2.1 enum L4\_irq\_flow\_type

Interrupt flow types.

**Enumerator:**

- `L4_IRQ_F_NONE` None.
- `L4_IRQ_F_LEVEL` Level triggered.
- `L4_IRQ_F_EDGE` Edge triggered.
- `L4_IRQ_F_POS` Positive trigger.
- `L4_IRQ_F_NEG` Negative trigger.
- `L4_IRQ_F_LEVEL_HIGH` Level high trigger.
- `L4_IRQ_F_LEVEL_LOW` Level low trigger.

**L4\_IRQ\_F\_POS\_EDGE** Positive edge trigger.  
**L4\_IRQ\_F\_NEG\_EDGE** Negative edge trigger.  
**L4\_IRQ\_F\_MASK** Mask.

Definition at line 61 of file [icu.h](#).

### 9.53.3 Function Documentation

#### 9.53.3.1 `l4_mshtag_t l4_irq_attach ( l4_cap_idx_t irq, l4_umword_t label, l4_cap_idx_t thread ) throw () [inline]`

Attach to an interrupt source.

##### Parameters

*irq* IRQ to attach to.  
*label* Identifier of the IRQ.  
*flow\_type* Interrupt type, see L4\_irq\_flow\_type  
*thread* Thread to attach the interrupt to.

##### Returns

Syscall return tag

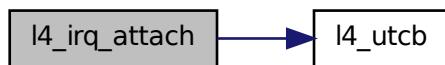
##### Examples:

[examples/sys/isr/main.c](#).

Definition at line 277 of file [irq.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 9.53.3.2 `l4_mshtag_t l4_irq_chain ( l4_cap_idx_t irq, l4_umword_t label, l4_cap_idx_t slave ) throw () [inline]`

Chain an IRQ to another master IRQ source.

The chaining feature of IRQ objects allows to deal with shared IRQs. For chaining IRQs there must be a master IRQ object, bound to the real IRQ source. Note, the master IRQ must not have a thread attached

to it. This function allows to add a limited number of slave IRQs to this master IRQ, with the semantics that each of the slave IRQs is triggered whenever the master IRQ is triggered. The master IRQ will be masked automatically when an IRQ is delivered and shall be unmasked when all attached slave IRQs are unmasked.

### Parameters

*irq* The master IRQ object.

*label* Identifier of the IRQ.

*flow\_type* Interrupt type, see L4\_irq\_flow\_type

*slave* The slave that shall be attached to the master.

### Returns

Syscall return tag

Definition at line 284 of file [irq.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.53.3.3 l4\_mshtag\_t l4\_irq\_detach ( l4\_cap\_idx\_t irq ) throw () [inline]

Detach from an interrupt source.

### Parameters

*irq* IRQ to detach from.

### Returns

Syscall return tag

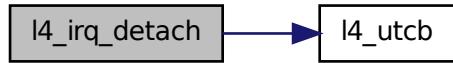
### Examples:

[examples/sys/isr/main.c](#).

Definition at line 291 of file [irq.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.53.3.4 l4\_mshtag\_t l4\_irq\_trigger( l4\_cap\_idx\_t irq ) throw() [inline]

Trigger an IRQ.

#### Parameters

*irq* IRQ to trigger.

#### Precondition

*irq* must be a reference to an IRQ.

#### Returns

Syscall return tag, note this is a send only operation, i.e. there is no return value except for failed sending.

Definition at line 297 of file [irq.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.53.3.5 l4\_mshtag\_t l4\_irq\_receive( l4\_cap\_idx\_t irq, l4\_timeout\_t to ) throw() [inline]

Unmask and wait for specified IRQ.

#### Parameters

*irq* IRQ to wait for.

*to* Timeout.

#### Returns

Syscall return tag

#### Examples:

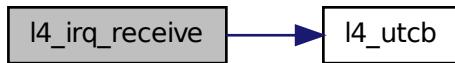
[examples/sys/isr/main.c](#).

Definition at line [303](#) of file [irq.h](#).

References [l4\\_utcb\(\)](#).

Referenced by [L4Re::Util::Event\\_buffer\\_consumer\\_t< PAYLOAD >::process\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 9.53.3.6 `l4_mshtag_t l4_irq_wait( l4_cap_idx_t irq, l4_umword_t * label, l4_timeout_t to ) throw() [inline]`

Unmask IRQ and wait for any message.

#### Parameters

*irq* IRQ to wait for.

*label* Receive label.

*to* Timeout.

#### Returns

Syscall return tag

Definition at line 309 of file [irq.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### **9.53.3.7 `l4_mshtag_t l4_irq_unmask( l4_cap_idx_t irq ) throw() [inline]`**

Unmask IRQ.

#### **Parameters**

*irq* IRQ to unmask.

#### **Returns**

Syscall return tag

#### **Note**

`l4_irq_wait` and `l4_irq_receive` are doing the unmask themselves.

Definition at line 316 of file [irq.h](#).

References [l4\\_utcb\(\)](#).

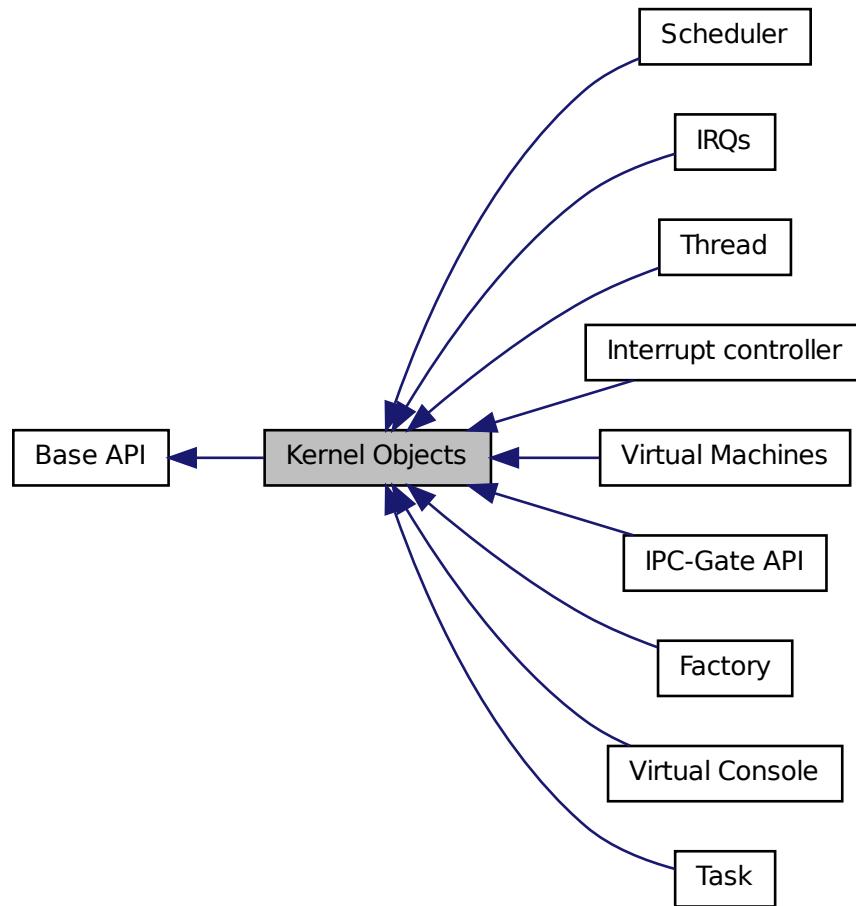
Here is the call graph for this function:



## **9.54 Kernel Objects**

API of kernel objects.

Collaboration diagram for Kernel Objects:



## Data Structures

- class [L4::Kobject](#)  
*Base class for all kinds of kernel objects, referred to by capabilities.*
- class [L4::Meta](#)  
*Meta interface that shall be implemented by each L4Re object and gives access to the dynamic type information for L4Re objects.*

## Modules

- [IPC-Gate API](#)

*Secure communication object.*

- [Factory](#)

*A factory is used to create all kinds of kernel objects.*

- [Virtual Machines](#)

*Virtual Machine API.*

- [Interrupt controller](#)

*The ICU class.*

- [IRQs](#)

*The IRQ and IRQ class.*

- [Scheduler](#)

*Scheduler object.*

- [Task](#)

*Class definition of the Task kernel object.*

- [Thread](#)

*Thread object.*

- [Virtual Console](#)

*Virtual console for simple character based input and output.*

## Defines

- `#define L4_DISABLE_COPY(_class)`

*Disable copy of a class.*

- `#define L4_KOBJECT_DISABLE_COPY(_class)`

*Disable copy and instantiation of a class.*

- `#define L4_KOBJECT(_class) L4_KOBJECT_DISABLE_COPY(_class)`

*Declare a kernel object class.*

### 9.54.1 Detailed Description

```
API of kernel objects. #include <14/sys/capability>
#include <14/sys/kernel_object.h>
```

### 9.54.2 Define Documentation

#### 9.54.2.1 `#define L4_DISABLE_COPY( _class )`

**Value:**

```
private:
    _class(_class const &);           \
    _class operator = (_class const &); \
```

Disable copy of a class.

#### Parameters

*\_class* is the name of the class that shall not have value copy semantics.

The typical use of this is:

```
class Non_value
{
    L4_DISABLE_COPY(Non_value)

    ...
}
```

Definition at line 397 of file [capability](#).

#### 9.54.2.2 #define L4\_KOBJECT\_DISABLE\_COPY( *\_class* )

##### Value:

```
protected:
    _class();
    L4_DISABLE_COPY(_class)
```

Disable copy and instantiation of a class.

#### Parameters

*\_class* is the name of the class to be not copyable and not instantiatable.

The typical use looks like:

```
class Type
{
    L4_KOBJECT_DISABLE_COPY(Type)
};
```

Definition at line 416 of file [capability](#).

#### 9.54.2.3 #define L4\_KOBJECT( *\_class* ) L4\_KOBJECT\_DISABLE\_COPY(*\_class*)

Declare a kernel object class.

#### Parameters

*\_class* is the class name.

The use of this macro disables copy and instantiation of the class as needed for kernel object classes derived from [L4::Kobject](#).

The typical use looks like:

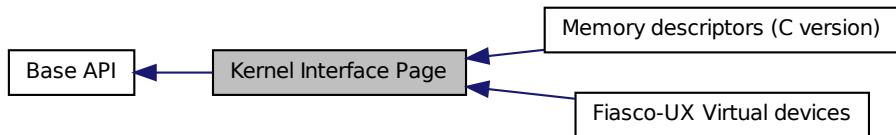
```
class Type : public L4::Kobject_t<Type, L4::Kobject>
{
    L4_KOBJECT(Type)
};
```

Definition at line 438 of file [capability](#).

## 9.55 Kernel Interface Page

Kernel Interface Page.

Collaboration diagram for Kernel Interface Page:



## Data Structures

- struct [l4\\_kernel\\_info\\_t](#)  
*L4 Kernel Interface Page.*
- class [L4::Kip::Mem\\_desc](#)  
*Memory descriptors stored in the kernel interface page.*

## Modules

- [Memory descriptors \(C version\)](#)  
*C Interface for KIP memory descriptors.*
- [Fiasco-UX Virtual devices](#)  
*Virtual hardware devices, provided by Fiasco-UX.*

## Defines

- #define [L4\\_KERNEL\\_INFO\\_MAGIC](#) (0x4BE6344CL)  
*Kernel Info Page identifier ("L4&lt;K").*

## Typedefs

- `typedef struct l4_kernel_info_t l4_kernel_info_t`  
`L4 Kernel Interface Page.`
- `typedef struct l4_kernel_info_t l4_kernel_info_t`  
`L4 Kernel Interface Page.`

## Functions

- `l4_umword_t l4_kip_version (l4_kernel_info_t *kip) throw ()`  
*Get the kernel version.*
- `const char * l4_kip_version_string (l4_kernel_info_t *kip) throw ()`  
*Get the kernel version string.*
- `int l4_kernel_info_version_offset (l4_kernel_info_t *kip) throw ()`  
*Return offset in bytes of version\_strings relative to the KIP base.*

### 9.55.1 Detailed Description

Kernel Interface Page. C interface for the Kernel Interface Page:

```
#include <l4/sys/kip.h>
```

C++ interface for the Kernel Interface Page:

```
#include <l4/sys/kip>
```

### 9.55.2 Function Documentation

#### 9.55.2.1 `l4_umword_t l4_kip_version ( l4_kernel_info_t * kip ) throw () [inline]`

Get the kernel version.

##### Parameters

`kip` Kernel Interface Page.

##### Returns

Kernel version string. 0 if KIP could not be mapped.

Definition at line 98 of file `kip.h`.

#### 9.55.2.2 `const char * l4_kip_version_string ( l4_kernel_info_t * kip ) throw () [inline]`

Get the kernel version string.

**Parameters**

*kip* Kernel Interface Page.

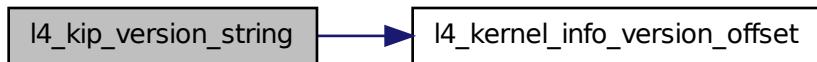
**Returns**

Kernel version string.

Definition at line 102 of file [kip.h](#).

References [l4\\_kernel\\_info\\_version\\_offset\(\)](#).

Here is the call graph for this function:



### 9.55.2.3 int l4\_kernel\_info\_version\_offset ( l4\_kernel\_info\_t \* kip ) throw () [inline]

Return offset in bytes of version\_strings relative to the KIP base.

**Parameters**

*kip* Pointer to the kernel into page (KIP).

**Returns**

offset of version\_strings relative to the KIP base address, in bytes.

Definition at line 106 of file [kip.h](#).

Referenced by [l4\\_kip\\_version\\_string\(\)](#).

Here is the caller graph for this function:



## 9.56 Memory descriptors (C version)

C Interface for KIP memory descriptors.

Collaboration diagram for Memory descriptors (C version):



### Data Structures

- struct `l4_kernel_info_mem_desc_t`  
*Memory descriptor data structure.*

### Typedefs

- typedef struct `l4_kernel_info_mem_desc_t` `l4_kernel_info_mem_desc_t`  
*Memory descriptor data structure.*

### Enumerations

- enum `l4_mem_type_t` {
   
`l4_mem_type_undefined` = 0x0, `l4_mem_type_conventional` = 0x1, `l4_mem_type_reserved` = 0x2,  
`l4_mem_type_dedicated` = 0x3,  
`l4_mem_type_shared` = 0x4, `l4_mem_type_bootloader` = 0xe, `l4_mem_type_archspecific` = 0xf }
   
*Type of a memory descriptor.*

### Functions

- `l4_kernel_info_mem_desc_t * l4_kernel_info_get_mem_descs (l4_kernel_info_t *kip) L4_NOTHROW`  
*Get pointer to memory descriptors from KIP.*
- `unsigned l4_kernel_info_get_num_mem_descs (l4_kernel_info_t *kip) L4_NOTHROW`  
*Get number of memory descriptors in KIP.*
- `void l4_kernel_info_set_mem_desc (l4_kernel_info_mem_desc_t *md, l4_addr_t start, l4_addr_t end, unsigned type, unsigned virt, unsigned sub_type) L4_NOTHROW`  
*Populate a memory descriptor.*

- `l4_umword_t l4_kernel_info_get_mem_desc_start (l4_kernel_info_mem_desc_t *md) L4_-NOTHROW`  
*Get start address of the region described by the memory descriptor.*
- `l4_umword_t l4_kernel_info_get_mem_desc_end (l4_kernel_info_mem_desc_t *md) L4_-NOTHROW`  
*Get end address of the region described by the memory descriptor.*
- `l4_umword_t l4_kernel_info_get_mem_desc_type (l4_kernel_info_mem_desc_t *md) L4_-NOTHROW`  
*Get type of the memory region.*
- `l4_umword_t l4_kernel_info_get_mem_desc_subtype (l4_kernel_info_mem_desc_t *md) L4_-NOTHROW`  
*Get sub-type of memory region.*
- `l4_umword_t l4_kernel_info_get_mem_desc_is_virtual (l4_kernel_info_mem_desc_t *md) L4_-NOTHROW`  
*Get virtual flag of the memory descriptor.*

### 9.56.1 Detailed Description

C Interface for KIP memory descriptors. #include <[l4/sys/memdesc.h](#)>

This module contains the C functions to access the memory descriptor in the kernel interface page (KIP).

### 9.56.2 Typedef Documentation

#### 9.56.2.1 `typedef struct l4_kernel_info_mem_desc_t l4_kernel_info_mem_desc_t`

Memory descriptor data structure.

##### Note

This data type is opaque, and must be accessed by the accessor functions defined in this module.

### 9.56.3 Enumeration Type Documentation

#### 9.56.3.1 `enum l4_mem_type_t`

Type of a memory descriptor.

##### Enumerator:

- `l4_mem_type_undefined` Undefined, unused descriptor.
- `l4_mem_type_conventional` Conventional memory.
- `l4_mem_type_reserved` Reserved memory for kernel etc.
- `l4_mem_type_dedicated` Dedicated memory (some device memory).

*l4\_mem\_type\_shared* Shared memory (not implemented).

*l4\_mem\_type\_bootloader* Memory owned by the boot loader.

*l4\_mem\_type\_archspecific* Architecture specific memory (e.g., ACPI memory).

Definition at line 44 of file [memdesc.h](#).

## 9.56.4 Function Documentation

### 9.56.4.1 `unsigned l4_kernel_info_get_num_mem_descs ( l4_kernel_info_t * kip ) [inline]`

Get number of memory descriptors in KIP.

#### Returns

Number of memory descriptors.

Definition at line 178 of file [memdesc.h](#).

### 9.56.4.2 `void l4_kernel_info_set_mem_desc ( l4_kernel_info_mem_desc_t * md, l4_addr_t start, l4_addr_t end, unsigned type, unsigned virt, unsigned sub_type ) [inline]`

Populate a memory descriptor.

#### Parameters

*md* Pointer to memory descriptor

*start* Start of region

*end* End of region

*type* Type of region

*virt* 1 if virtual region, 0 if physical region

*sub\_type* Sub type.

Definition at line 185 of file [memdesc.h](#).

### 9.56.4.3 `l4_umword_t l4_kernel_info_get_mem_desc_start ( l4_kernel_info_mem_desc_t * md ) [inline]`

Get start address of the region described by the memory descriptor.

#### Returns

Start address.

Definition at line 200 of file [memdesc.h](#).

**9.56.4.4 l4\_umword\_t l4\_kernel\_info\_get\_mem\_desc\_end ( l4\_kernel\_info\_mem\_desc\_t \* *md* ) [inline]**

Get end address of the region described by the memory descriptor.

**Returns**

End address.

Definition at line 207 of file [memdesc.h](#).

**9.56.4.5 l4\_umword\_t l4\_kernel\_info\_get\_mem\_desc\_type ( l4\_kernel\_info\_mem\_desc\_t \* *md* ) [inline]**

Get type of the memory region.

**Returns**

Type of the region (see [l4\\_mem\\_type\\_t](#)).

Definition at line 214 of file [memdesc.h](#).

**9.56.4.6 l4\_umword\_t l4\_kernel\_info\_get\_mem\_desc\_subtype ( l4\_kernel\_info\_mem\_desc\_t \* *md* ) [inline]**

Get sub-type of memory region.

**Returns**

Sub-type.

The sub type is defined for architecture specific memory descriptors (see [l4\\_mem\\_type\\_archspecific](#)) and has architecture specific meaning.

Definition at line 221 of file [memdesc.h](#).

**9.56.4.7 l4\_umword\_t l4\_kernel\_info\_get\_mem\_desc\_is\_virtual ( l4\_kernel\_info\_mem\_desc\_t \* *md* ) [inline]**

Get virtual flag of the memory descriptor.

**Returns**

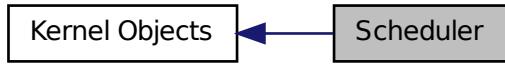
1 if region is virtual memory, 0 if region is physical memory

Definition at line 228 of file [memdesc.h](#).

## 9.57 Scheduler

Scheduler object.

Collaboration diagram for Scheduler:



## Data Structures

- struct `l4_sched_cpu_set_t`  
*CPU sets.*
- struct `l4_sched_param_t`  
*Scheduler parameter set.*

## Typedefs

- typedef struct `l4_sched_cpu_set_t` `l4_sched_cpu_set_t`  
*CPU sets.*
- typedef struct `l4_sched_param_t` `l4_sched_param_t`  
*Scheduler parameter set.*

## Enumerations

- enum `L4_scheduler_ops` { `L4_SCHEDULER_INFO_OP` = 0UL, `L4_SCHEDULER_RUN_THREAD_OP` = 1UL, `L4_SCHEDULER_IDLE_TIME_OP` = 2UL }  
*Operations on the Scheduler object.*

## Functions

- `l4_sched_cpu_set_t l4_sched_cpu_set (l4_umword_t offset, unsigned char granularity, l4_umword_t map=1) throw ()`  
*Get scheduler information.*
- `l4_mshtag_t l4_scheduler_info (l4_cap_idx_t scheduler, l4_umword_t *cpu_max, l4_sched_cpu_set_t *cpus) throw ()`  
*Construct scheduler parameter.*
- `l4_sched_param_t l4_sched_param (unsigned prio, l4_cpu_time_t quantum=0) throw ()`  
*Construct scheduler parameter.*

- `l4_msgtag_t l4_scheduler_run_thread (l4_cap_idx_t scheduler, l4_cap_idx_t thread, l4_sched_param_t const *sp) throw ()`  
*Run a thread on a Scheduler.*
- `l4_msgtag_t l4_scheduler_idle_time (l4_cap_idx_t scheduler, l4_sched_cpu_set_t const *cpus) throw ()`  
*Query idle time of a CPU, in  $\mu$ s.*
- `int l4_scheduler_is_online (l4_cap_idx_t scheduler, l4_umword_t cpu) throw ()`  
*Query if a CPU is online.*

### 9.57.1 Detailed Description

Scheduler object. #include <l4/sys/scheduler.h>

### 9.57.2 Enumeration Type Documentation

#### 9.57.2.1 enum L4\_scheduler\_ops

Operations on the Scheduler object.

**Enumerator:**

`L4_SCHEDULER_INFO_OP` Query infos about the scheduler.

`L4_SCHEDULER_RUN_THREAD_OP` Run a thread on this scheduler.

`L4_SCHEDULER_IDLE_TIME_OP` Query idle time for the scheduler.

Definition at line 185 of file `scheduler.h`.

### 9.57.3 Function Documentation

#### 9.57.3.1 l4\_sched\_cpu\_set\_t l4\_sched\_cpu\_set ( l4\_umword\_t offset, unsigned char granularity, l4\_umword\_t map = 1 ) throw () [inline]

##### Parameters

`offset` Offset.

`granularity` Granularity in log2 notation.

`map` Bitmap of CPUs, defaults to 1 in C++.

##### Returns

CPU set.

##### Examples:

[examples/sys/migrate/thread\\_migrate.cc](#).

Definition at line 195 of file [scheduler.h](#).

References [l4\\_sched\\_cpu\\_set\\_t::granularity](#), [l4\\_sched\\_cpu\\_set\\_t::map](#), and [l4\\_sched\\_cpu\\_set\\_t::offset](#).

Referenced by [l4\\_sched\\_param\(\)](#).

Here is the caller graph for this function:



### 9.57.3.2 [l4\\_mshtag\\_t l4\\_scheduler\\_info \( l4\\_cap\\_idx\\_t scheduler, l4\\_umword\\_t \\* cpu\\_max, l4\\_sched\\_cpu\\_set\\_t \\* cpus \) throw \(\) \[inline\]](#)

Get scheduler information.

#### Parameters

*scheduler* Scheduler object.

#### Return values

*cpu\_max* maximum number of CPUs ever available.

#### Parameters

*cpus* *cpus.offset* is first CPU of interest. *cpus.granularity* (see [l4\\_sched\\_cpu\\_set\\_t](#)).

#### Return values

*cpus* *cpus.map* Bitmap of online CPUs.

#### Returns

0 on success, <0 error code otherwise.

Definition at line 284 of file [scheduler.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



---

**9.57.3.3 `l4_mshtag_t l4_scheduler_run_thread ( l4_cap_idx_t scheduler, l4_cap_idx_t thread, l4_sched_param_t const * sp ) throw () [inline]`**

Run a thread on a Scheduler.

#### Parameters

***scheduler*** Scheduler object.

***thread*** Thread to run.

***sp*** Scheduling parameters.

#### Returns

0 on success, <0 error code otherwise.

Definition at line 291 of file [scheduler.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



**9.57.3.4 `l4_mshtag_t l4_scheduler_idle_time ( l4_cap_idx_t scheduler, l4_sched_cpu_set_t const * cpus ) throw () [inline]`**

Query idle time of a CPU, in Åts.

#### Parameters

***scheduler*** Scheduler object.

***cpus*** Set of CPUs to query.

The consumed time is returned as `l4_kernel_clock_t` at UTCB message register 0.

Definition at line 298 of file [scheduler.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



**9.57.3.5 int l4\_scheduler\_is\_online ( l4\_cap\_idx\_t scheduler, l4\_umword\_t cpu ) throw () [inline]**

Query if a CPU is online.

#### Parameters

*scheduler* Scheduler object.

*cpu* CPU number.

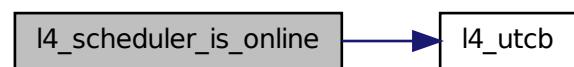
#### Returns

true if online, false if not (or any other query error).

Definition at line 304 of file [scheduler.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



## 9.58 Task

Class definition of the Task kernel object.

Collaboration diagram for Task:



## Data Structures

- class [L4::Task](#)

*An L4 Task.*

## Enumerations

- enum [l4\\_unmap\\_flags\\_t](#) { [L4\\_FP\\_ALL\\_SPACES](#), [L4\\_FP\\_DELETE\\_OBJ](#), [L4\\_FP\\_OTHER\\_SPACES](#) }

*Flags for the unmap operation.*

## Functions

- [l4\\_msgtag\\_t l4\\_task\\_map \(l4\\_cap\\_idx\\_t dst\\_task, l4\\_cap\\_idx\\_t src\\_task, l4\\_fpage\\_t const snd\\_fpage, l4\\_addr\\_t snd\\_base\) L4\\_NOTHROW](#)

*Map resources available in the source task to a destination task.*

- [l4\\_msgtag\\_t l4\\_task\\_unmap \(l4\\_cap\\_idx\\_t task, l4\\_fpage\\_t const fpage, l4\\_umword\\_t map\\_mask\) L4\\_NOTHROW](#)

*Revoke rights from the task.*

- [l4\\_msgtag\\_t l4\\_task\\_unmap\\_batch \(l4\\_cap\\_idx\\_t task, l4\\_fpage\\_t const \\*fpages, unsigned num\\_fpages, unsigned long map\\_mask\) L4\\_NOTHROW](#)

*Revoke rights from a task.*

- [l4\\_msgtag\\_t l4\\_task\\_cap\\_valid \(l4\\_cap\\_idx\\_t task, l4\\_cap\\_idx\\_t cap\) L4\\_NOTHROW](#)

*Test whether a capability selector points to a valid capability.*

- [l4\\_msgtag\\_t l4\\_task\\_cap\\_has\\_child \(l4\\_cap\\_idx\\_t task, l4\\_cap\\_idx\\_t cap\) L4\\_NOTHROW](#)

*Test whether a capability has child mappings (in another task).*

- [l4\\_msgtag\\_t l4\\_task\\_cap\\_equal \(l4\\_cap\\_idx\\_t task, l4\\_cap\\_idx\\_t cap\\_a, l4\\_cap\\_idx\\_t cap\\_b\) L4\\_NOTHROW](#)

*Test if two capabilities point to the same object.*

- `l4_mshtag_t l4_task_add_ku_mem (l4_cap_idx_t task, l4_fpage_t const ku_mem) L4_NOTHROW`

*Add kernel-user memory.*

### 9.58.1 Detailed Description

Class definition of the Task kernel object. #include <14/sys/task.h>

The `L4` task class represents a combination of the address spaces provided by the `L4` micro kernel. A task consists of at least a memory address space and an object address space. On IA32 there is also an IO-port address space.

A task object can be created using a Factory, see [Factory \(l4\\_factory\\_create\\_task\(\)\)](#).

### 9.58.2 Enumeration Type Documentation

#### 9.58.2.1 enum l4\_unmap\_flags\_t

Flags for the unmap operation.

##### See also

[L4::Task::unmap\(\)](#) and [l4\\_task\\_unmap\(\)](#)

##### Enumerator:

`L4_FP_ALL_SPACES` Flag to tell the unmap operation to unmap all child mappings including the mapping in the invoked task.

##### See also

[L4::Task::unmap\(\)](#) [l4\\_task\\_unmap\(\)](#)

`L4_FP_DELETE_OBJ` Flag that indicates that the unmap operation on a capability shall try to delete the corresponding objects immediately.

##### See also

[L4::Task::unmap\(\)](#) [l4\\_task\\_unmap\(\)](#)

`L4_FP_OTHER_SPACES` Counterpart to `L4_FP_ALL_SPACES`, unmap only child mappings.

##### See also

[L4::Task::unmap\(\)](#) [l4\\_task\\_unmap\(\)](#)

Definition at line 163 of file `consts.h`.

### 9.58.3 Function Documentation

#### 9.58.3.1 l4\_mshtag\_t l4\_task\_map ( l4\_cap\_idx\_t dst\_task, l4\_cap\_idx\_t src\_task, l4\_fpage\_t const snd\_fpage, l4\_addr\_t snd\_base ) [inline]

Map resources available in the source task to a destination task.

##### Parameters

`dst_task` Capability selector of destination task

*src\_task* Capability selector of source task

*snd\_fpage* Send flexpage that describes an area in the address space or object space of the source task

*snd\_base* Send base that describes an offset in the receive window of the destination task.

### Returns

Syscall return tag

This method allows for asynchronous rights delegation from one task to another. It can be used to share memory as well as to delegate access to objects.

Referenced by [L4::Cap< Dataspace >::move\(\)](#).

Here is the caller graph for this function:



### 9.58.3.2 l4\_mshtag\_t l4\_task\_unmap ( l4\_cap\_idx\_t task, l4\_fpage\_t const fpage, l4\_umword\_t map\_mask ) [inline]

Revoke rights from the task.

### Parameters

*task* Capability selector of destination task

*fpage* Flexpage that describes an area in the address space or object space of the destination task

*map\_mask* Unmap mask, see [l4\\_unmap\\_flags\\_t](#)

### Returns

Syscall return tag

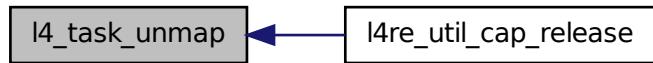
This method allows to revoke rights from the destination task and from all the tasks that got the rights delegated from that task (i.e., this operation does a recursive rights revocation).

### Note

Calling this function on the object space can cause a root capability of an object to be destructed, which destroys the object itself.

Referenced by [l4re\\_util\\_cap\\_release\(\)](#).

Here is the caller graph for this function:



### 9.58.3.3 `l4_mshtag_t l4_task_unmap_batch( l4_cap_idx_t task, l4_fpage_t const * fpages, unsigned num_fpages, unsigned long map_mask ) [inline]`

Revoke rights from a task.

#### Parameters

`task` Capability selector of destination task

`fpages` An array of flexpages that describes an area in the address space or object space of the destination task each

`num_fpages` The size of the fpages array in elements (number of fpages sent).

`map_mask` Unmap mask, see [l4\\_unmap\\_flags\\_t](#)

#### Returns

Syscall return tag

This method allows to revoke rights from the destination task and from all the tasks that got the rights delegated from that task (i.e., this operation does a recursive rights revocation).

#### Precondition

The caller needs to take care that `num_fpages` is not bigger than `L4_UTCB_GENERIC_DATA_SIZE - 2`.

#### Note

Calling this function on the object space can cause a root capability of an object to be destructed, which destroys the object itself.

Definition at line [326](#) of file [task.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.58.3.4 `l4_mshtag_t l4_task_cap_valid( l4_cap_idx_t task, l4_cap_idx_t cap ) [inline]`

Test whether a capability selector points to a valid capability.

#### Parameters

- task* Capability selector of the destination task to do the lookup in
- cap* Capability selector to look up in the destination task

#### Returns

label contains 1 if valid, 0 if invalid

Definition at line 334 of file [task.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.58.3.5 `l4_mshtag_t l4_task_cap_has_child( l4_cap_idx_t task, l4_cap_idx_t cap ) [inline]`

Test whether a capability has child mappings (in another task).

#### Parameters

- task* Capability selector of the destination task to do the lookup in
- cap* Capability selector to look up in the destination task

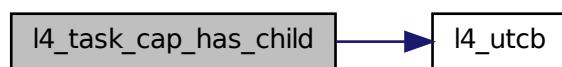
### Returns

label contains 1 if it has at least one child, 0 if not or invalid

Definition at line 340 of file [task.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.58.3.6 `l4_mshtag_t l4_task_cap_equal ( l4_cap_idx_t task, l4_cap_idx_t cap_a, l4_cap_idx_t cap_b ) [inline]`

Test if two capabilities point to the same object.

### Parameters

*task* Capability selector of the destination task to do the lookup in

*cap\_a* Capability selector to compare

*cap\_b* Capability selector to compare

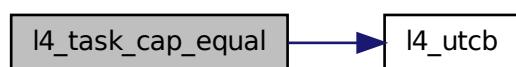
### Returns

label contains 1 if equal, 0 if not equal

Definition at line 346 of file [task.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.58.3.7 `l4_mshtag_t l4_task_add_ku_mem( l4_cap_idx_t task, l4_fpage_t const ku_mem ) [inline]`

Add kernel-user memory.

#### Parameters

*task* Capability selector of the task to add the memory to

*ku\_mem* Flexpage describing the virtual area the memory goes to.

#### Returns

Syscall return tag

Definition at line 353 of file [task.h](#).

References [l4\\_utcb\(\)](#).

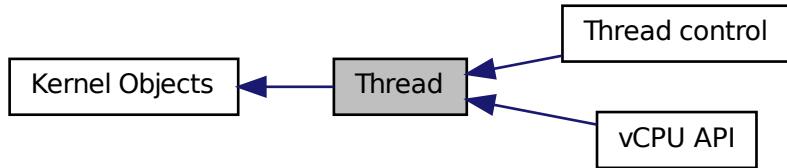
Here is the call graph for this function:



## 9.59 Thread

Thread object.

Collaboration diagram for Thread:



## Data Structures

- class [L4::Thread](#)

*L4 kernel thread.*

## Modules

- Thread control

*API for Thread Control method.*

- vCPU API

*vCPU API*

## Enumerations

- enum L4\_thread\_ops {

*L4\_THREAD\_CONTROL\_OP = 0UL, L4\_THREAD\_EX\_REGS\_OP = 1UL, L4\_THREAD\_SWITCH\_OP = 2UL, L4\_THREAD\_STATS\_OP = 3UL,  
 L4\_THREAD\_VCPU\_RESUME\_OP = 4UL, L4\_THREAD\_REGISTER\_DELETE\_IRQ = 5UL,  
 L4\_THREAD MODIFY\_SENDER = 6UL, L4\_THREAD\_VCPU\_CONTROL = 7UL,  
 L4\_THREAD\_GDT\_X86\_OP = 0x10UL, L4\_THREAD\_OPCODE\_MASK = 0xffff }*

*Operations on thread objects.*

- enum L4\_thread\_control\_flags {

*L4\_THREAD\_CONTROL\_SET\_PAGER = 0x0010000, L4\_THREAD\_CONTROL\_BIND\_TASK = 0x0200000, L4\_THREAD\_CONTROL\_ALIEN = 0x0400000, L4\_THREAD\_CONTROL\_UX\_NATIVE = 0x0800000,*

*L4\_THREAD\_CONTROL\_SET\_EXC\_HANDLER = 0x1000000 }*

*Flags for the thread control operation.*

- enum L4\_thread\_control\_mr\_indices {

*L4\_THREAD\_CONTROL\_MR\_IDX\_FLAGS = 0, L4\_THREAD\_CONTROL\_MR\_IDX\_PAGER = 1, L4\_THREAD\_CONTROL\_MR\_IDX\_EXC\_HANDLER = 2, L4\_THREAD\_CONTROL\_MR\_IDX\_FLAG\_VALS = 4,*

*L4\_THREAD\_CONTROL\_MR\_IDX\_BIND\_UTCB = 5, L4\_THREAD\_CONTROL\_MR\_IDX\_BIND\_TASK = 6 }*

*Indices for the values in the message register for thread control.*

- enum L4\_thread\_ex\_regs\_flags { L4\_THREAD\_EX\_REGS\_CANCEL = 0x10000UL, L4\_THREAD\_EX\_REGS\_TRIGGER\_EXCEPTION = 0x20000UL }

*Flags for the thread ex-reg operation.*

## Functions

- `l4_msgtag_t l4_thread_ex_regs (l4_cap_idx_t thread, l4_addr_t ip, l4_addr_t sp, l4_umword_t flags)`  
`L4_NOTHROW`

*Exchange basic thread registers.*

- `l4_mshtag_t l4_thread_ex_regs_ret (l4_cap_idx_t thread, l4_addr_t *ip, l4_addr_t *sp, l4_umword_t *flags) L4_NOTHROW`  
*Exchange basic thread registers and return previous values.*
- `l4_mshtag_t l4_thread_yield (void) L4_NOTHROW`  
*Yield current time slice.*
- `l4_mshtag_t l4_thread_switch (l4_cap_idx_t to_thread) L4_NOTHROW`  
*Switch to another thread (and donate the remaining time slice).*
- `l4_mshtag_t l4_thread_stats_time (l4_cap_idx_t thread) L4_NOTHROW`  
*Get consumed timed of thread in Âµs.*
- `l4_mshtag_t l4_thread_vcpu_resume_start (void) L4_NOTHROW`  
*vCPU return from event handler.*
- `l4_mshtag_t l4_thread_vcpu_resume_commit (l4_cap_idx_t thread, l4_mshtag_t tag) L4_NOTHROW`  
*Commit vCPU resume.*
- `l4_mshtag_t l4_thread_vcpu_control (l4_cap_idx_t thread, l4_addr_t vcpu_state) L4_NOTHROW`  
*Enable or disable the vCPU feature for the thread.*
- `l4_mshtag_t l4_thread_vcpu_control_ext (l4_cap_idx_t thread, l4_addr_t ext_vcpu_state) L4_NOTHROW`  
*Enable or disable the extended vCPU feature for the thread.*
- `l4_mshtag_t l4_thread_register_del_irq (l4_cap_idx_t thread, l4_cap_idx_t irq) L4_NOTHROW`  
*Register an IRQ that will trigger upon deletion events.*
- `l4_mshtag_t l4_thread_modify_sender_start (void) L4_NOTHROW`  
*Start a thread sender modification sequence.*
- `int l4_thread_modify_sender_add (l4_umword_t match_mask, l4_umword_t match, l4_umword_t del_bits, l4_umword_t add_bits, l4_mshtag_t *tag) L4_NOTHROW`  
*Add a modification pattern to a sender modification sequence.*
- `l4_mshtag_t l4_thread_modify_sender_commit (l4_cap_idx_t thread, l4_mshtag_t tag) L4_NOTHROW`  
*Apply (commit) a sender modification sequence.*

### 9.59.1 Detailed Description

Thread object. #include <l4/sys/thread.h>

The thread class defines a thread of execution in the L4 context. Usually user-level and kernel threads are mapped 1:1 to each other. Thread kernel object are created using a Factory, see [Factory \(l4\\_factory\\_create\\_thread\(\)\)](#).

An L4 thread encapsulates:

- CPU state
  - General-purpose registers
  - Program counter
  - Stack pointer
- FPU state
- Scheduling parameters
  - CPU-set
  - Priority (0-255)
  - Time slice length
- Execution state
  - Blocked, Runnable, Running

Thread objects provide an API for

- Thread configuration and manipulation
- Thread switching.

The thread control functions are used to control various aspects of a thread. See [l4\\_thread\\_control\\_start\(\)](#) for more information.

## 9.59.2 Enumeration Type Documentation

### 9.59.2.1 enum L4\_thread\_ops

Operations on thread objects.

**Enumerator:**

- L4\_THREAD\_CONTROL\_OP** Control operation.
- L4\_THREAD\_EX\_REGS\_OP** Exchange registers operation.
- L4\_THREAD\_SWITCH\_OP** Do a thread switch.
- L4\_THREAD\_STATS\_OP** Thread statistics.
- L4\_THREAD\_VCPU\_RESUME\_OP** VCPU resume.
- L4\_THREAD\_REGISTER\_DELETE\_IRQ** Register an IPC-gate deletion IRQ.
- L4\_THREAD MODIFY\_SENDER** Modify all senders IDs that match the given pattern.
- L4\_THREAD\_VCPU\_CONTROL** Enable / disable VCPU feature.
- L4\_THREAD\_GDT\_X86\_OP** Gdt.
- L4\_THREAD\_OPCODE\_MASK** Mask for opcodes.

Definition at line [585](#) of file [thread.h](#).

### 9.59.2.2 enum L4\_thread\_control\_flags

Flags for the thread control operation.

**Enumerator:**

- L4\_THREAD\_CONTROL\_SET\_PAGER* The pager will be given.
- L4\_THREAD\_CONTROL\_BIND\_TASK* The task to bind the thread to will be given.
- L4\_THREAD\_CONTROL\_ALIEN* Alien state of the thread is set.
- L4\_THREAD\_CONTROL\_UX\_NATIVE* Fiasco-UX only: pass-through of host system calls is set.
- L4\_THREAD\_CONTROL\_SET\_EXC\_HANDLER* The exception handler of the thread will be given.

Definition at line 610 of file [thread.h](#).

### 9.59.2.3 enum L4\_thread\_control\_mr\_indices

Indices for the values in the message register for thread control.

**Enumerator:**

**See also**

- L4\_THREAD\_CONTROL\_MR\_IDX\_FLAGS* [L4\\_thread\\_control\\_flags](#).
- L4\_THREAD\_CONTROL\_MR\_IDX\_PAGER* Index for pager cap.
- L4\_THREAD\_CONTROL\_MR\_IDX\_EXC\_HANDLER* Index for exception handler.
- L4\_THREAD\_CONTROL\_MR\_IDX\_FLAG\_VALS* Index for feature values.
- L4\_THREAD\_CONTROL\_MR\_IDX\_BIND\_UTCB* Index for UTCB address for bind.
- L4\_THREAD\_CONTROL\_MR\_IDX\_BIND\_TASK* Index for task flex-page for bind.

Definition at line 633 of file [thread.h](#).

### 9.59.2.4 enum L4\_thread\_ex\_regs\_flags

Flags for the thread ex-reg operation.

**Enumerator:**

- L4\_THREAD\_EX\_REGS\_CANCEL* Cancel ongoing IPC in the thread.
- L4\_THREAD\_EX\_REGS\_TRIGGER\_EXCEPTION* Trigger artificial exception in thread.

Definition at line 648 of file [thread.h](#).

## 9.59.3 Function Documentation

### 9.59.3.1 l4\_msgtag\_t l4\_thread\_ex\_regs ( l4\_cap\_idx\_t *thread*, l4\_addr\_t *ip*, l4\_addr\_t *sp*, l4\_umword\_t *flags* ) [inline]

Exchange basic thread registers.

## Parameters

*thread* Thread to manipulate

*ip* New instruction pointer, use ~0UL to leave the instruction pointer unchanged

*sp* New stack pointer, use ~0UL to leave the stack pointer unchanged

*flags* Ex-reg flags, see [L4\\_thread\\_ex\\_regs\\_flags](#)

## Returns

System call return tag

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see *flags*).

## Examples:

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 788 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.59.3.2 l4\_mshtag\_t l4\_thread\_ex\_regs\_ret ( l4\_cap\_idx\_t thread, l4\_addr\_t \* ip, l4\_addr\_t \* sp, l4\_umword\_t \* flags ) [inline]

Exchange basic thread registers and return previous values.

## Parameters

*thread* Thread to manipulate

## Return values

*ip* New instruction pointer, use ~0UL to leave the instruction pointer unchanged, return previous instruction pointer

*sp* New stack pointer, use ~0UL to leave the stack pointer unchanged, returns previous stack pointer

*flags* Ex-reg flags, see [L4\\_thread\\_ex\\_regs\\_flags](#), return previous CPU flags of the thread.

## Returns

System call return tag

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see *flags*).

Returned values are valid only if function returns successfully.

Definition at line 795 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.59.3.3 l4\_mshtag\_t l4\_thread\_yield ( void ) [inline]

Yield current time slice.

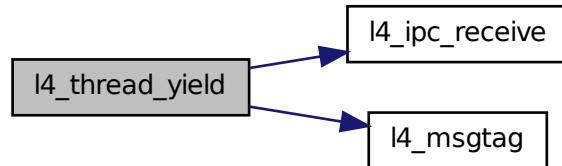
#### Returns

system call return tag

Definition at line 748 of file [thread.h](#).

References [L4\\_INVALID\\_CAP](#), [L4\\_IPC\\_BOTH\\_TIMEOUT\\_0](#), [l4\\_ipc\\_receive\(\)](#), and [l4\\_mshtag\(\)](#).

Here is the call graph for this function:



### 9.59.3.4 l4\_mshtag\_t l4\_thread\_switch ( l4\_cap\_idx\_t to\_thread ) [inline]

Switch to another thread (and donate the remaining time slice).

**Parameters**

*to\_thread* The thread to switch to.

**Returns**

system call return tag

Definition at line 848 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**9.59.3.5 l4\_mshtag\_t l4\_thread\_stats\_time ( l4\_cap\_idx\_t *thread* ) [inline]**

Get consumed timed of thread in Ås.

**Parameters**

*thread* Thread to get the consumed time from.

The consumed time is returned as l4\_kernel\_clock\_t at UTCB message register 0.

Definition at line 857 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**9.59.3.6 l4\_mshtag\_t l4\_thread\_vcpu\_resume\_start ( void ) [inline]**

vCPU return from event handler.

## Returns

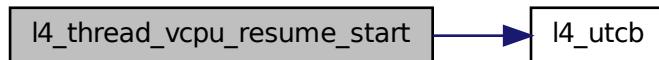
Message tag to be used for [l4\\_sndfpage\\_add\(\)](#) and [l4\\_thread\\_vcpu\\_commit\(\)](#)

The vCPU resume functionality is split in multiple functions to allow the specification of additional send-flex-pages using [l4\\_sndfpage\\_add\(\)](#).

Definition at line 863 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### **9.59.3.7 [l4\\_mshtag\\_t l4\\_thread\\_vcpu\\_resume\\_commit\( l4\\_cap\\_idx\\_t thread, l4\\_mshtag\\_t tag \) \[inline\]](#)**

Commit vCPU resume.

## Parameters

*thread* Thread to be resumed, using the invalid cap can be used for the current thread.

*tag* Tag to use, returned by [l4\\_thread\\_vcpu\\_resume\\_start\(\)](#)

## Returns

System call result message tag. In extended vCPU mode and when the virtual interrupts are cleared, the return code 1 flags an incoming IPC message, whereas 0 indicates a VM exit.

To resume into another address space the capability to the target task must be set in the vCPU-state (

## See also

[l4\\_vcpu\\_state\\_t](#)). The task needs to be set only once, consecutive resumes to the same address space should use an invalid capability. The kernel resets the field to [L4\\_INVALID\\_CAP](#). To release a task use a different task or use an invalid capability with the [L4\\_SYSF\\_REPLY](#) flag set.

Definition at line 869 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.59.3.8 `l4_mshtag_t l4_thread_vcpu_control( l4_cap_idx_t thread, l4_addr_t vcpu_state ) [inline]`

Enable or disable the vCPU feature for the thread.

#### Parameters

*thread* The thread for which the vCPU feature shall be enabled or disabled.

*vcpu\_state* The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address.

#### Returns

Systemcall result message tag.

This function enables the vCPU feature of the *thread* if *vcpu\_state* is set to a valid kernel-user-memory address, or disables the vCPU feature if *vcpu\_state* is 0.

Definition at line 906 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.59.3.9 `l4_mshtag_t l4_thread_vcpu_control_ext( l4_cap_idx_t thread, l4_addr_t ext_vcpu_state ) [inline]`

Enable or disable the extended vCPU feature for the thread.

## Parameters

***thread*** The thread for which the extended vCPU feature shall be enabled or disabled.

***vcpu\_state*** The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address.

## Returns

Systemcall result message tag.

The extended vCPU feature allows the use of hardware-virtualization features such as Intel's VT or AMD's SVM.

This function enables the extended vCPU feature of the *thread* if *vcpu\_state* is set to a valid kernel-user-memory address, or disables the vCPU feature if *vcpu\_state* is 0.

Definition at line 921 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.59.3.10 **`l4_msgtag_t l4_thread_register_del_irq( l4_cap_idx_t thread, l4_cap_idx_t irq ) [inline]`**

Register an IRQ that will trigger upon deletion events.

## Parameters

***thread*** Thread to register IRQ for.

***irq*** Irq to register.

## Returns

System call result message tag.

Definition at line 889 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.59.3.11 `l4_mshtag_t l4_thread_modify_sender_start( void ) [inline]`

Start a thread sender modification sequence.

Add modification rules with `l4_thread_modify_sender_add()` and commit with `l4_thread_modify_sender_commit()`. Do not touch the UTCB between `l4_thread_modify_sender_start()` and `l4_thread_modify_sender_commit()`.

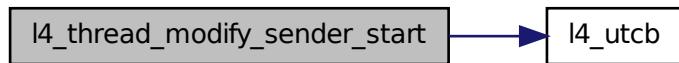
#### See also

`l4_thread_modify_sender_add`  
`l4_thread_modify_sender_commit`

Definition at line 962 of file `thread.h`.

References `l4_utcb()`.

Here is the call graph for this function:



### 9.59.3.12 `int l4_thread_modify_sender_add( l4_umword_t match_mask, l4_umword_t match, l4_umword_t del_bits, l4_umword_t add_bits, l4_mshtag_t * tag ) [inline]`

Add a modification pattern to a sender modification sequence.

#### Parameters

`tag` Tag received from `l4_thread_modify_sender_start()` or previous `l4_thread_modify_sender_add()` calls from the same sequence.

`match_mask` Bitmask of bits to match the label.

***match*** Bitmask that must be equal to the label after applying match\_mask.

***del\_bits*** Bits to be deleted from the label.

***add\_bits*** Bits to be added to the label.

### Returns

0 on sucess, <0 on error

In pseudo code: if ((sender\_label & match\_mask) == match) { label = (label & ~del\_bits) | add\_bits; }

Only the first match is applied.

### See also

[l4\\_thread\\_modify\\_sender\\_start](#)  
[l4\\_thread\\_modify\\_sender\\_commit](#)

Definition at line 968 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.59.3.13 ***l4\_mshtag\_t l4\_thread\_modify\_sender\_commit ( l4\_cap\_idx\_t thread, l4\_mshtag\_t tag ) [inline]***

Apply (commit) a sender modification sequence.

### See also

[l4\\_thread\\_modify\\_sender\\_start](#)  
[l4\\_thread\\_modify\\_sender\\_add](#)

Definition at line 979 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

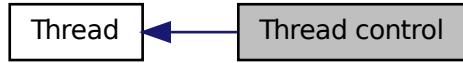
Here is the call graph for this function:



## 9.60 Thread control

API for Thread Control method.

Collaboration diagram for Thread control:



### Functions

- void [l4\\_thread\\_control\\_start](#) (void) L4\_NOTHROW  
*Start a thread control API sequence.*
- void [l4\\_thread\\_control\\_pager](#) ([l4\\_cap\\_idx\\_t](#) pager) L4\_NOTHROW  
*Set the pager.*
- void [l4\\_thread\\_control\\_exc\\_handler](#) ([l4\\_cap\\_idx\\_t](#) exc\_handler) L4\_NOTHROW  
*Set the exception handler.*
- void [l4\\_thread\\_control\\_bind](#) ([l4\\_utcb\\_t](#) \*thread\_utcb, [l4\\_cap\\_idx\\_t](#) task) L4\_NOTHROW  
*Bind the thread to a task.*
- void [l4\\_thread\\_control\\_alien](#) (int on) L4\_NOTHROW  
*Enable alien mode.*
- void [l4\\_thread\\_control\\_ux\\_host\\_syscall](#) (int on) L4\_NOTHROW  
*Enable pass through of native host (Linux) system calls.*
- [l4\\_msgtag\\_t](#) [l4\\_thread\\_control\\_commit](#) ([l4\\_cap\\_idx\\_t](#) thread) L4\_NOTHROW

*Commit the thread control parameters.*

### 9.60.1 Detailed Description

API for Thread Control method. The thread control API provides access to almost any parameter of a thread object. The API is based on a single invocation of the thread object. However, because of the huge amount of parameters, the API provides a set of functions to set specific parameters of a thread and a commit function to commit the thread control call (see [l4\\_thread\\_control\\_commit\(\)](#)).

A thread control operation must always start with [l4\\_thread\\_control\\_start\(\)](#) and be committed with [l4\\_thread\\_control\\_commit\(\)](#). All other thread control parameter setter functions must be called between these two functions.

An example for a sequence of thread control API calls can be found below.

```
l4_utcb_t *u = l4_utcb();
l4_thread_control_start(u);
l4_thread_control_pager(u, pager_cap);
l4_thread_control_bind (u, thread_utcb, task);
l4_thread_control_commit(u, thread_cap);
```

### 9.60.2 Function Documentation

#### 9.60.2.1 void l4\_thread\_control\_start ( void ) [inline]

Start a thread control API sequence.

This function starts a sequence of thread control API functions. After this functions any of following functions may be called in any order.

- [l4\\_thread\\_control\\_pager\(\)](#)
- [l4\\_thread\\_control\\_exc\\_handler\(\)](#)
- [l4\\_thread\\_control\\_bind\(\)](#)
- [l4\\_thread\\_control\\_alien\(\)](#)
- [l4\\_thread\\_control\\_ux\\_host\\_syscall\(\)](#) (Fiasco-UX only)

To commit the changes to the thread [l4\\_thread\\_control\\_commit\(\)](#) must be called in the end.

#### Note

The thread control API calls store the parameters for the thread in the UTCB of the caller, this means between [l4\\_thread\\_control\\_start\(\)](#) and [l4\\_thread\\_control\\_commit\(\)](#) no functions that modify the UTCB contents must be called.

#### Examples:

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 802 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.60.2.2 void l4\_thread\_control\_pager ( l4\_cap\_idx\_t *pager* ) [inline]

Set the pager.

#### Parameters

*pager* Capability selector invoked to send a page-fault IPC.

#### Note

The pager capability selector is interpreted in the task the thread is bound to (executes in).

#### Examples:

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 808 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.60.2.3 void l4\_thread\_control\_exc\_handler ( l4\_cap\_idx\_t *exc\_handler* ) [inline]

Set the exception handler.

**Parameters**

*exc\_handler* Capability selector invoked to send an exception IPC.

**Note**

The exception-handler capability selector is interpreted in the task the thread is bound to (executes in).

**Examples:**

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 814 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.60.2.4 void l4\_thread\_control\_bind ( l4\_utcb\_t \* *thread\_utcb*, l4\_cap\_idx\_t *task* ) [inline]

Bind the thread to a task.

**Parameters**

*thread\_utcb* The address of the UTCB in the target task.

*task* The target task of the thread.

Binding a thread to a task has the effect that the thread afterwards executes code within that task and has access to the resources visible within that task.

**Note**

There should not be more than one thread use a UTCB to prevent data corruption.

**Examples:**

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 821 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 9.60.2.5 void l4\_thread\_control\_alien( int on ) [inline]

Enable alien mode.

##### Parameters

*on* Boolean value defining the state of the feature.

Alien mode means the thread is not allowed to invoke L4 kernel objects directly and it is also not allowed to allocate FPU state. All those operations result in an exception IPC that gets sent through the pager capability. The responsible pager can then selectively allow an object invocation or allocate FPU state for the thread.

This feature can be used to attach a debugger to a thread and trace all object invocations.

##### Examples:

[examples/sys/aliens/main.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line 827 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 9.60.2.6 void l4\_thread\_control\_ux\_host\_syscall( int on ) [inline]

Enable pass through of native host (Linux) system calls.

**Parameters**

*on* Boolean value defining the state of the feature.

**Precondition**

Running on Fiasco-UX

This enables the thread to do host system calls. This feature is only available in Fiasco-UX and ignored in other environments.

Definition at line 833 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.60.2.7 l4\_mshtag\_t l4\_thread\_control\_commit( l4\_cap\_idx\_t *thread* ) [inline]

Commit the thread control parameters.

**Parameters**

*thread* Capability selector of target thread to commit to.

**Returns**

system call return tag

**Examples:**

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 839 of file [thread.h](#).

References [l4\\_utcb\(\)](#).

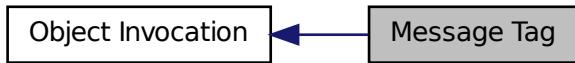
Here is the call graph for this function:



## 9.61 Message Tag

API related to the message tag data type.

Collaboration diagram for Message Tag:



## Data Structures

- struct [l4\\_mshtag\\_t](#)  
*Message tag data structure.*

## Typedefs

- typedef struct [l4\\_mshtag\\_t](#) [l4\\_mshtag\\_t](#)  
*Message tag data structure.*

## Enumerations

- enum [l4\\_mshtag\\_protocol](#) {
   
    [L4\\_PROTO\\_NONE](#) = 0, [L4\\_PROTO\\_ALLOW\\_SYSCALL](#) = 1, [L4\\_PROTO\\_PF\\_EXCEPTION](#) = 1, [L4\\_PROTO\\_IRQ](#) = -1L,
   
    [L4\\_PROTO\\_PAGE\\_FAULT](#) = -2L, [L4\\_PROTO\\_PREEMPTION](#) = -3L, [L4\\_PROTO\\_SYS\\_EXCEPTION](#) = -4L, [L4\\_PROTO\\_EXCEPTION](#) = -5L,
 }

```
L4_PROTO_SIGMA0 = -6L, L4_PROTO_IO_PAGE_FAULT = -8L, L4_PROTO_KOBJECT = -10L, L4_PROTO_TASK = -11L,
L4_PROTO_THREAD = -12L, L4_PROTO_LOG = -13L, L4_PROTO_SCHEDULER = -14L, L4_PROTO_FACTORY = -15L }
```

*Message tag for IPC operations.*

- enum l4\_mshtag\_flags {
 L4\_MSGTAG\_ERROR, L4\_MSGTAG\_XCPU, L4\_MSGTAG\_TRANSFER\_FPU, L4\_MSGTAG\_SCHEDULE,
 L4\_MSGTAG\_PROPAGATE, L4\_MSGTAG\_FLAGS }

*Flags for message tags.*

## Functions

- l4\_mshtag\_t l4\_mshtag (long label, unsigned words, unsigned items, unsigned flags) throw ()
 *Create a message tag from the specified values.*
- long l4\_mshtag\_label (l4\_mshtag\_t t) throw ()
 *Get the protocol of tag.*
- unsigned l4\_mshtag\_words (l4\_mshtag\_t t) throw ()
 *Get the number of untyped words.*
- unsigned l4\_mshtag\_items (l4\_mshtag\_t t) throw ()
 *Get the number of typed items.*
- unsigned l4\_mshtag\_flags (l4\_mshtag\_t t) throw ()
 *Get the flags.*
- unsigned l4\_mshtag\_has\_error (l4\_mshtag\_t t) throw ()
 *Test for error indicator flag.*
- unsigned l4\_mshtag\_is\_page\_fault (l4\_mshtag\_t t) throw ()
 *Test for page-fault protocol.*
- unsigned l4\_mshtag\_is\_preemption (l4\_mshtag\_t t) throw ()
 *Test for preemption protocol.*
- unsigned l4\_mshtag\_is\_sys\_exception (l4\_mshtag\_t t) throw ()
 *Test for system-exception protocol.*
- unsigned l4\_mshtag\_is\_exception (l4\_mshtag\_t t) throw ()
 *Test for exception protocol.*
- unsigned l4\_mshtag\_is\_sigma0 (l4\_mshtag\_t t) throw ()
 *Test for sigma0 protocol.*
- unsigned l4\_mshtag\_is\_io\_page\_fault (l4\_mshtag\_t t) throw ()
 *Test for IO-page-fault protocol.*

### 9.61.1 Detailed Description

API related to the message tag data type. #include <l4/sys/types.h>

### 9.61.2 Typedef Documentation

#### 9.61.2.1 `typedef struct l4_mshtag_t l4_mshtag_t`

Message tag data structure.

```
#include <l4/sys/types.h>
```

Describes the details of an IPC operation, in particular which parts of the UTCB have to be transmitted, and also flags to enable real-time and FPU extensions.

The message tag also contains a user-defined label that could be used to specify a protocol ID. Some negative values are reserved for kernel protocols such as page faults and exceptions.

The type must be treated completely opaque.

### 9.61.3 Enumeration Type Documentation

#### 9.61.3.1 `enum l4_mshtag_protocol`

Message tag for IPC operations.

All predefined protocols used by the kernel.

**Enumerator:**

`L4_PROTO_NONE` Default protocol tag to reply to kernel.

`L4_PROTO_ALLOW_SYSCALL` Allow an alien the system call.

`L4_PROTO_PF_EXCEPTION` Make an exception out of a page fault.

`L4_PROTO_IRQ` IRQ message.

`L4_PROTO_PAGE_FAULT` Page fault message.

`L4_PROTO_PREEMPTION` Preemption message.

`L4_PROTO_SYS_EXCEPTION` System exception.

`L4_PROTO_EXCEPTION` Exception.

`L4_PROTO_SIGMA0` Sigma0 protocol.

`L4_PROTO_IO_PAGEFAULT` I/O page fault message.

`L4_PROTO_KOBJECT` Protocol for messages to a generic kobject.

`L4_PROTO_TASK` Protocol for messages to a task object.

`L4_PROTO_THREAD` Protocol for messages to a thread object.

`L4_PROTO_LOG` Protocol for messages to a log object.

`L4_PROTO_SCHEDULER` Protocol for messages to a scheduler object.

`L4_PROTO_FACTORY` Protocol for messages to a factory object.

Definition at line 49 of file [types.h](#).

### 9.61.3.2 enum l4\_mshtag\_flags

Flags for message tags.

**Enumerator:**

***L4\_MSGTAG\_ERROR*** Error indicator flag.

***L4\_MSGTAG\_XCPU*** Cross-CPU invocation indicator flag.

***L4\_MSGTAG\_TRANSFER\_FPU*** Enable FPU transfer flag for IPC. By enabling this flag when sending IPC, the sender indicates that the contents of the FPU shall be transferred to the receiving thread. However, the receiver has to indicate its willingness to receive FPU context in its buffer descriptor register (BDR).

***L4\_MSGTAG\_SCHEDULE*** Enable schedule in IPC flag. Usually IPC operations donate the remaining time slice of a thread to the called thread. Enabling this flag when sending IPC does a real scheduling decision. However, this flag decreases IPC performance.

***L4\_MSGTAG\_PROPAGATE*** Enable IPC propagation. This flag enables IPC propagation, which means an IPC reply-connection from the current caller will be propagated to the new IPC receiver. This makes it possible to propagate an IPC call to a third thread, which may then directly answer to the caller.

***L4\_MSGTAG\_FLAGS*** Mask for all flags.

Definition at line 89 of file [types.h](#).

### 9.61.4 Function Documentation

#### 9.61.4.1 l4\_mshtag\_t l4\_mshtag ( long *label*, unsigned *words*, unsigned *items*, unsigned *flags* ) throw () [inline]

Create a message tag from the specified values.

Message tag functions.

##### Parameters

*label* the user-defined label

*words* the number of untyped words within the UTCB

*items* the number of typed items (e.g., flex pages) within the UTCB

*flags* the IPC flags for realtime and FPU extensions

##### Returns

Message tag

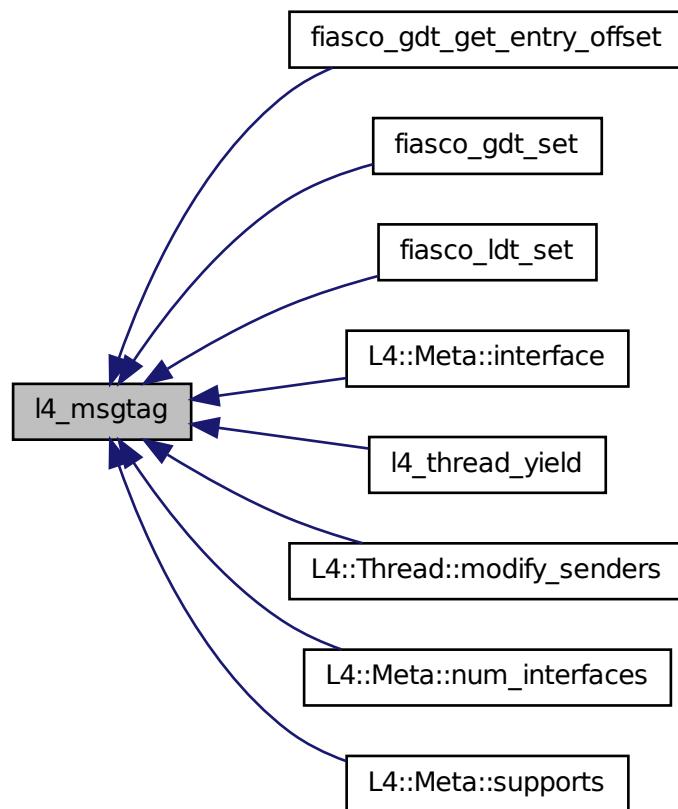
##### Examples:

[examples/sys/aliens/main.c](#), [examples/sys/ipc/ipc\\_example.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 366 of file [types.h](#).

Referenced by [fiasco\\_gdt\\_get\\_entry\\_offset\(\)](#), [fiasco\\_gdt\\_set\(\)](#), [fiasco\\_ldt\\_set\(\)](#), [L4::Meta::interface\(\)](#), [l4\\_thread\\_yield\(\)](#), [L4::Thread::modify\\_senders\(\)](#), [L4::Meta::num\\_interfaces\(\)](#), and [L4::Meta::supports\(\)](#).

Here is the caller graph for this function:



#### 9.61.4.2 long l4\_mshtag\_label( l4\_mshtag\_t t ) throw() [inline]

Get the protocol of tag.

##### Parameters

`t` The tag

##### Returns

Label

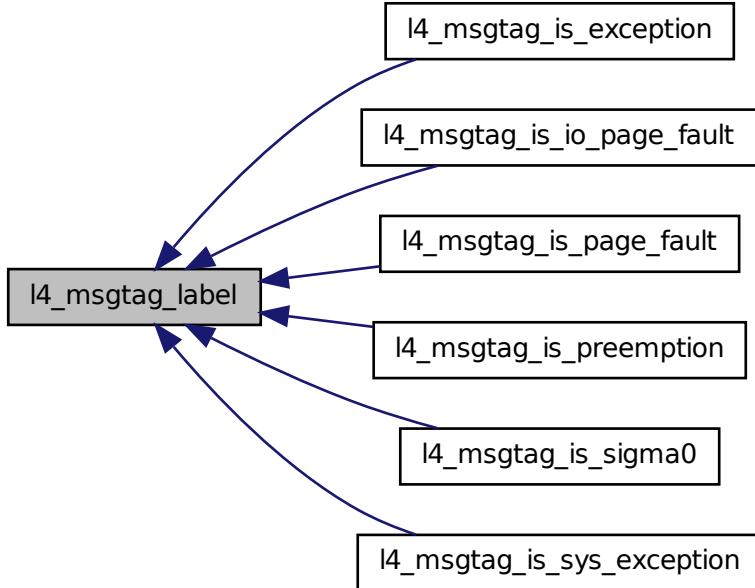
##### Examples:

[examples/sys/singlestep/main.c](#), and [examples/sys/start-with-exc/main.c](#).

Definition at line 376 of file [types.h](#).

Referenced by [l4\\_mshtag\\_is\\_exception\(\)](#), [l4\\_mshtag\\_is\\_io\\_page\\_fault\(\)](#), [l4\\_mshtag\\_is\\_page\\_fault\(\)](#), [l4\\_mshtag\\_is\\_preemption\(\)](#), [l4\\_mshtag\\_is\\_sigma0\(\)](#), and [l4\\_mshtag\\_is\\_sys\\_exception\(\)](#).

Here is the caller graph for this function:



#### 9.61.4.3 `unsigned l4_mshtag_words ( l4_mshtag_t t ) throw () [inline]`

Get the number of untyped words.

##### Parameters

*t* The tag

##### Returns

Number of words

##### Examples:

[examples/sys/utcb-ipc/main.c](#).

Definition at line [380](#) of file [types.h](#).

#### 9.61.4.4 `unsigned l4_mshtag_items ( l4_mshtag_t t ) throw () [inline]`

Get the number of typed items.

**Parameters**

*t* The tag

**Returns**

Number of items.

Definition at line 384 of file [types.h](#).

**9.61.4.5 unsigned l4\_mshtag\_flags ( l4\_mshtag\_t t ) throw () [inline]**

Get the flags.

The flag are defined by [l4\\_mshtag\\_flags](#).

**Parameters**

*t* the tag

**Returns**

Flags

Definition at line 388 of file [types.h](#).

**9.61.4.6 unsigned l4\_mshtag\_has\_error ( l4\_mshtag\_t t ) throw () [inline]**

Test for error indicator flag.

**Parameters**

*t* the tag

**Returns**

>0 for yes, 0 for no

Return whether the kernel operation caused a communication error, e.g. with IPC. if true: utcb->error is valid, otherwise utcb->error is not valid

**Examples:**

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 393 of file [types.h](#).

Referenced by [l4\\_ipc\\_error\(\)](#).

Here is the caller graph for this function:



#### 9.61.4.7 `unsigned l4_mshtag_is_page_fault( l4_mshtag_t t ) throw() [inline]`

Test for page-fault protocol.

##### Parameters

*t* the tag

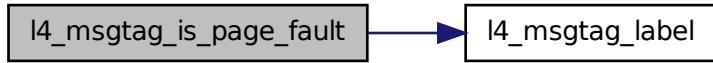
##### Returns

Boolean value

Definition at line 398 of file [types.h](#).

References [l4\\_mshtag\\_label\(\)](#).

Here is the call graph for this function:



#### 9.61.4.8 `unsigned l4_mshtag_is_preemption( l4_mshtag_t t ) throw() [inline]`

Test for preemption protocol.

##### Parameters

*t* the tag

##### Returns

Boolean value

Definition at line 401 of file [types.h](#).

References [l4\\_mshtag\\_label\(\)](#).

Here is the call graph for this function:



**9.61.4.9 unsigned l4\_mshtag\_is\_sys\_exception ( l4\_mshtag\_t t ) throw () [inline]**

Test for system-exception protocol.

**Parameters**

*t* the tag

**Returns**

Boolean value

Definition at line 404 of file [types.h](#).

References [l4\\_mshtag\\_label\(\)](#).

Here is the call graph for this function:

**9.61.4.10 unsigned l4\_mshtag\_is\_exception ( l4\_mshtag\_t t ) throw () [inline]**

Test for exception protocol.

**Parameters**

*t* the tag

**Returns**

Boolean value

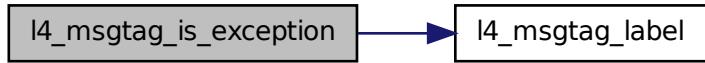
**Examples:**

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/start-with-exc/main.c](#).

Definition at line 407 of file [types.h](#).

References [l4\\_mshtag\\_label\(\)](#).

Here is the call graph for this function:



#### 9.61.4.11 `unsigned l4_msntag_is_sigma0 ( l4_msntag_t t ) throw () [inline]`

Test for sigma0 protocol.

##### Parameters

*t* the tag

##### Returns

Boolean value

Definition at line 410 of file [types.h](#).

References [l4\\_msntag\\_label\(\)](#).

Here is the call graph for this function:



#### 9.61.4.12 `unsigned l4_msntag_is_io_page_fault ( l4_msntag_t t ) throw () [inline]`

Test for IO-page-fault protocol.

##### Parameters

*t* the tag

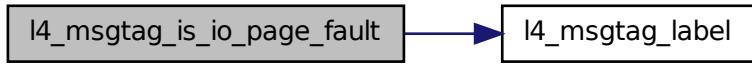
##### Returns

Boolean value

Definition at line 413 of file [types.h](#).

References [l4\\_msgtag\\_label\(\)](#).

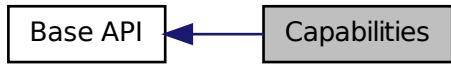
Here is the call graph for this function:



## 9.62 Capabilities

Functions and definitions related to capabilities.

Collaboration diagram for Capabilities:



## Data Structures

- class [L4::Cap\\_base](#)  
*Base class for all kinds of capabilities.*
- class [L4::Cap< T >](#)  
*Capability Selector a la C++.*

## TypeDefs

- typedef unsigned long [l4\\_cap\\_idx\\_t](#)  
*L4 Capability selector Type.*

## Enumerations

- enum [l4\\_cap\\_consts\\_t](#) { [L4\\_CAP\\_SHIFT](#), [L4\\_CAP\\_SIZE](#) , [L4\\_CAP\\_MASK](#), [L4\\_INVALID\\_CAP](#) }

*Constants related to capability selectors.*

- enum `l4_default_caps_t` {
   
`L4_BASE_TASK_CAP, L4_BASE_FACTORY_CAP, L4_BASE_THREAD_CAP, L4_BASE_-`
  
`PAGER_CAP,`
  
`L4_BASE_LOG_CAP, L4_BASE_ICU_CAP, L4_BASE_SCHEDULER_CAP }`

*Default capabilities setup for the initial tasks.*

## Functions

- template<typename T, typename F>
   
`Cap< T > L4::cap_cast (Cap< F > const &c) throw ()`
  
*static\_cast for capabilities.*
- template<typename T, typename F>
   
`Cap< T > L4::cap_reinterpret_cast (Cap< F > const &c) throw ()`
  
*reinterpret\_cast for capabilities.*
- template<typename T, typename F>
   
`Cap< T > L4::cap_dynamic_cast (Cap< F > const &c) throw ()`
  
*dynamic\_cast for capabilities.*
- unsigned `l4_is_invalid_cap (l4_cap_idx_t c)` throw ()
   
*Test if a capability selector is the invalid capability.*
- unsigned `l4_is_valid_cap (l4_cap_idx_t c)` throw ()
   
*Test if a capability selector is a valid selector.*
- unsigned `l4_capability_equal (l4_cap_idx_t c1, l4_cap_idx_t c2)` throw ()
   
*Test if two capability selectors are equal.*

### 9.62.1 Detailed Description

Functions and definitions related to capabilities. #include <l4/sys/consts.h>

C++ interface for capabilities:

```
#include <l4/sys/capability>
```

C interface for capabilities:

```
#include <l4/sys/types.h>
```

### 9.62.2 Typedef Documentation

#### 9.62.2.1 `typedef unsigned long l4_cap_idx_t`

`L4` Capability selector Type.

```
#include <l4/sys/types.h>
```

Definition at line 319 of file [types.h](#).

### 9.62.3 Enumeration Type Documentation

#### 9.62.3.1 enum l4\_cap\_consts\_t

Constants related to capability selectors.

##### Enumerator:

- L4\_CAP\_SHIFT* Capability index shift.
- L4\_CAP\_SIZE* Offset of two consecutive capability selectors.
- L4\_CAP\_MASK* Mask to get only the relevant bits of an l4\_cap\_idx\_t.
- L4\_INVALID\_CAP* Invalid capability selector.

Definition at line 134 of file [consts.h](#).

#### 9.62.3.2 enum l4\_default\_caps\_t

Default capabilities setup for the initial tasks.

```
#include <l4/sys/consts.h>
```

These capability selectors are setup per default by the micro kernel for the two initial tasks, the Root-Pager (Sigma0) and the Root-Task (Moe).

##### Attention

This constants do not have any particular meaning for applications started by Moe, see [Initial Environment](#) for this kind of information.

##### See also

[Initial Environment](#) for information useful for normal user applications.

##### Enumerator:

- L4\_BASE\_TASK\_CAP* Capability selector for the current task.
- L4\_BASE\_FACTORY\_CAP* Capability selector for the factory.
- L4\_BASE\_THREAD\_CAP* Capability selector for the first thread.
- L4\_BASE\_PAGER\_CAP* Capability selector for the pager gate.
- L4\_BASE\_LOG\_CAP* Capability selector for the log object.
- L4\_BASE\_ICU\_CAP* Capability selector for the base icu object.
- L4\_BASE\_SCHEDULER\_CAP* Capability selector for the scheduler cap.

Definition at line 248 of file [consts.h](#).

### 9.62.4 Function Documentation

**9.62.4.1 template<typename T , typename F > Cap<T> L4::cap\_cast ( Cap<F> const & c ) throw () [inline]**

static\_cast for capabilities.

#### Parameters

*T* is the target type of the capability

*F* is the source type (and is usually implicitly set)

*c* is the source capability that shall be casted

#### Returns

A capability typed to the interface *T*.

The use of this cast operator is similar to the static\_cast<>() for C++ pointers. It does the same type checking and adjustment like C++ does on pointers.

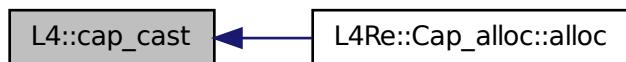
Example code:

```
L4::Cap<L4::Kobject> obj = ... ;
L4::Cap<L4::Icu> icu = L4::cap_cast<L4::Icu>(obj) ;
```

Definition at line 344 of file [capability](#).

Referenced by [L4Re::Cap\\_alloc::alloc\(\)](#).

Here is the caller graph for this function:



**9.62.4.2 template<typename T , typename F > Cap<T> L4::cap\_reinterpret\_cast ( Cap<F> const & c ) throw () [inline]**

reinterpret\_cast for capabilities.

#### Parameters

*T* is the target type of the capability

*F* is the source type (and is usually implicitly set)

*c* is the source capability that shall be casted

## Returns

A capability typed to the interface  $T$ .

The use of this cast operator is similar to the `reinterpret_cast<>()` for C++ pointers. It does not do any type checking or type adjustment.

Example code:

```
L4::Cap<L4::Kobject> obj = ... ;
L4::Cap<L4::Icu> icu = L4::cap_reinterpret_cast<L4::Icu>(obj);
```

Definition at line 368 of file [capability](#).

Referenced by [L4::cap\\_dynamic\\_cast\(\)](#).

Here is the caller graph for this function:



### 9.62.4.3 template<typename T , typename F > Cap<T> L4::cap\_dynamic\_cast ( Cap< F > const & c ) throw () [inline]

`dynamic_cast` for capabilities.

## Parameters

$T$  is the target type of the capability

$F$  is the source type (and is usually implicitly set)

$c$  is the source capability that shall be casted

## Returns

A capability typed to the interface  $T$ . If the object does not support the target interface  $T$  or does not support the [L4::Meta](#) interface the result is the invalid capability selector.

The use of this cast operator is similar to the `dynamic_cast<>()` for C++ pointers. It also induces overhead, because it uses the meta interface ([L4::Meta](#)) to do runtime type checking.

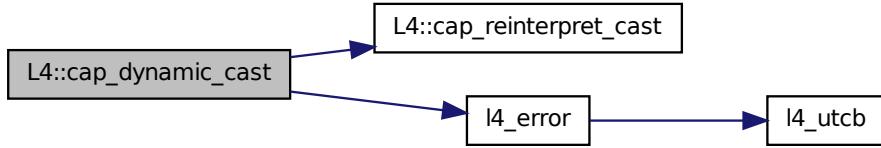
Example code:

```
L4::Cap<L4::Kobject> obj = ... ;
L4::Cap<L4::Icu> icu = L4::cap_dynamic_cast<L4::Icu>(obj);
```

Definition at line 548 of file [capability](#).

References [L4::cap\\_reinterpret\\_cast\(\)](#), and [l4\\_error\(\)](#).

Here is the call graph for this function:



#### 9.62.4.4 `unsigned l4_is_invalid_cap( l4_cap_idx_t c ) throw() [inline]`

Test if a capability selector is the invalid capability.

##### Parameters

*c* Capability selector

##### Returns

Boolean value

##### Examples:

[examples/libs/l4re/c/ma+rm.c](#), [examples/sys/aliens/main.c](#), [examples/sys/isr/main.c](#), [examples/sys/singlestep/main.c](#), [examples/sys/start-with-exc/main.c](#), and [examples/sys/utcb-ipc/main.c](#).

Definition at line 350 of file [types.h](#).

#### 9.62.4.5 `unsigned l4_is_valid_cap( l4_cap_idx_t c ) throw() [inline]`

Test if a capability selector is a valid selector.

##### Parameters

*c* Capability selector

##### Returns

Boolean value

Definition at line 354 of file [types.h](#).

#### 9.62.4.6 `unsigned l4_capability_equal( l4_cap_idx_t c1, l4_cap_idx_t c2 ) throw() [inline]`

Test if two capability selectors are equal.

**Parameters***c1* Capability*c2* Capability**Returns**

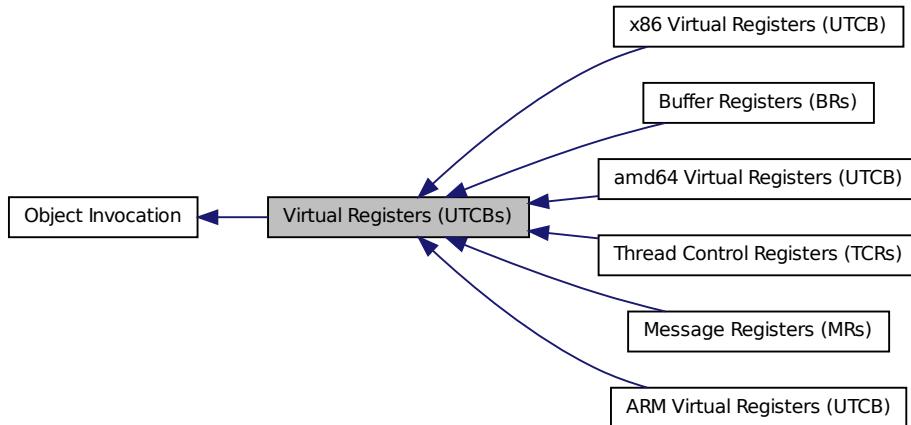
1 if equal, 0 if not equal

Definition at line 358 of file [types.h](#).References [L4\\_CAP\\_SHIFT](#).

## 9.63 Virtual Registers (UTCBs)

[L4](#) Virtual Registers (UTCB).

Collaboration diagram for Virtual Registers (UTCBs):



## Modules

- [Message Registers \(MRs\)](#)
- [Buffer Registers \(BRs\)](#)
- [Thread Control Registers \(TCRs\)](#)
- [ARM Virtual Registers \(UTCB\)](#)
- [amd64 Virtual Registers \(UTCB\)](#)
- [x86 Virtual Registers \(UTCB\)](#)

## Files

- file [utcbl.h](#)

*UTCB definitions for ARM.*

- file `utcb.h`

*UTCB definitions for amd64.*

- file `utcb.h`

*UTCB definitions for X86.*

## Typedefs

- typedef struct `l4_utcb_t` `l4_utcb_t`

*Opaque type for the UTCB.*

## Functions

- `l4_utcb_t * l4_utcb()` (void) throw ()

*Get the UTCB address.*

- `l4_msg_regs_t * l4_utcb_mr()` (void) throw ()

*Get the message-register block of a UTCB.*

- `l4_buf_regs_t * l4_utcb_br()` (void) throw ()

*Get the buffer-register block of a UTCB.*

- `l4_thread_regs_t * l4_utcb_tcr()` (void) throw ()

*Get the thread-control-register block of a UTCB.*

### 9.63.1 Detailed Description

[L4](#) Virtual Registers (UTCB). Includes:

```
#include <14/sys/utcb.h>
```

The virtual registers are part of the micro-kernel API and are located in the user-level thread control block (UTCB). The UTCB is a data structure defined by the micro kernel and located on kernel-provided memory. Each [L4](#) thread gets a unique UTCB assigned when it is bound to a task (see [Thread Control](#), `l4_thread_control_bind()` for more information).

The UTCB is arranged in three blocks of virtual registers.

- [Thread Control Registers \(TCRs\)](#)
- [Message Registers \(MRs\)](#)
- [Buffer Registers \(BRs\)](#)

To access the contents of the virtual registers the `l4_utcb_mr()`, `l4_utcb_tcr()`, and `l4_utcb_br()` functions must be used.

### 9.63.2 Typedef Documentation

#### 9.63.2.1 `typedef struct l4_utcb_t l4_utcb_t`

Opaque type for the UTCB.

To access the contents of the virtual registers the `l4_utcb_mr()`, `l4_utcb_tcr()`, and `l4_utcb_br()` functions must be used.

Definition at line 68 of file `utcb.h`.

### 9.63.3 Function Documentation

#### 9.63.3.1 `l4_msg_regs_t * l4_utcb_mr( void ) throw() [inline]`

Get the message-register block of a UTCB.

##### Returns

A pointer to the message-register block of u.

##### Examples:

`examples/sys/aliens/main.c`, `examples/sys/ipc/ipc_example.c`, `examples/sys/singlestep/main.c`, and `examples/sys/utcb-ipc/main.c`.

Definition at line 331 of file `utcb.h`.

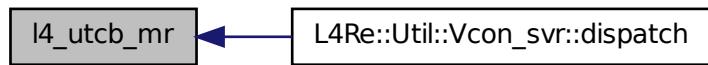
References `l4_utcb()`.

Referenced by `L4Re::Util::Vcon_svr< SVR >::dispatch()`.

Here is the call graph for this function:



Here is the caller graph for this function:



### 9.63.3.2 `l4_buf_regs_t * l4_utcb_br( void ) throw() [inline]`

Get the buffer-register block of a UTCB.

#### Returns

A pointer to the buffer-register block of `u`.

Definition at line 334 of file [utcb.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.63.3.3 `l4_thread_regs_t * l4_utcb_tcr( void ) throw() [inline]`

Get the thread-control-register block of a UTCB.

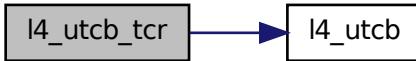
#### Returns

A pointer to the thread-control-register block of `u`.

Definition at line 337 of file [utcb.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



## 9.64 Message Registers (MRs)

Collaboration diagram for Message Registers (MRs):



### Data Structures

- struct [l4\\_msg\\_regs\\_t](#)  
*Encapsulation of the message-register block in the UTCB.*

### Modules

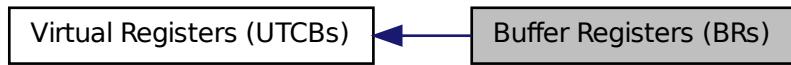
- [Exception registers](#)  
*Overly definition of the MRs for exception messages.*

### Typedefs

- typedef struct [l4\\_msg\\_regs\\_t](#) [l4\\_msg\\_regs\\_t](#)  
*Encapsulation of the message-register block in the UTCB.*

## 9.65 Buffer Registers (BRs)

Collaboration diagram for Buffer Registers (BRs):



### Data Structures

- struct [l4\\_buf\\_regs\\_t](#)

*Encapsulation of the buffer-registers block in the UTCB.*

## Typedefs

- `typedef struct l4_buf_regs_t l4_buf_regs_t`

*Encapsulation of the buffer-registers block in the UTCB.*

## Enumerations

- `enum l4_buffer_desc_consts_t { L4_BDR_MEM_SHIFT = 0, L4_BDR_IO_SHIFT = 5, L4_BDR_OBJ_SHIFT = 10 }`

*Constants for buffer descriptors.*

## Functions

- `void l4_utcb_inherit_fpu (int switch_on) throw ()`

*Enable or disable inheritance of FPU state to receiver.*

### 9.65.1 Enumeration Type Documentation

#### 9.65.1.1 enum l4\_buffer\_desc\_consts\_t

Constants for buffer descriptors.

##### Enumerator:

`L4_BDR_MEM_SHIFT` Bit offset for the memory-buffer index.

`L4_BDR_IO_SHIFT` Bit offset for the IO-buffer index.

`L4_BDR_OBJ_SHIFT` Bit offset for the capability-buffer index.

Definition at line 225 of file `consts.h`.

## 9.66 Thread Control Registers (TCRs)

Collaboration diagram for Thread Control Registers (TCRs):



## Data Structures

- struct `l4_thread_regs_t`

*Encapsulation of the thread-control-register block of the UTCB.*

## Typedefs

- typedef struct `l4_thread_regs_t` `l4_thread_regs_t`

*Encapsulation of the thread-control-register block of the UTCB.*

## 9.67 Exception registers

Overly definition of the MRs for exception messages.

Collaboration diagram for Exception registers:



## Functions

- `l4_exc_regs_t * l4_utcb_exc (void) throw ()`  
*Get the message-register block of a UTCB (for an exception IPC).*
- `l4_umword_t l4_utcb_exc_pc (l4_exc_regs_t *u) throw ()`  
*Access function to get the program counter of the exception state.*
- `void l4_utcb_exc_pc_set (l4_exc_regs_t *u, l4_addr_t pc) throw ()`  
*Set the program counter register in the exception state.*
- `unsigned long l4_utcb_exc_typeval (l4_exc_regs_t *u) throw ()`  
*Get the value out of an exception UTCB that describes the type of exception.*
- `int l4_utcb_exc_is_pf (l4_exc_regs_t *u) throw ()`  
*Check whether an exception IPC is a page fault.*
- `l4_addr_t l4_utcb_exc_pfa (l4_exc_regs_t *u) throw ()`  
*Function to get the L4 style page fault address out of an exception.*

### 9.67.1 Detailed Description

Overly definition of the MRs for exception messages.

### 9.67.2 Function Documentation

#### 9.67.2.1 `l4_exc_regs_t * l4_utcb_exc( void ) throw() [inline]`

Get the message-register block of a UTCB (for an exception IPC).

##### Returns

A pointer to the exception message in `u`.

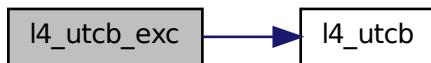
##### Examples:

[examples/sys/aliens/main.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line [340](#) of file [utcb.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



#### 9.67.2.2 `l4_umword_t l4_utcb_exc_pc( l4_exc_regs_t * u ) throw() [inline]`

Access function to get the program counter of the exception state.

##### Parameters

`u` UTCB

##### Returns

The program counter register out of the exception state.

##### Examples:

[examples/sys/aliens/main.c](#), and [examples/sys/singlestep/main.c](#).

Definition at line [89](#) of file [utcb.h](#).

**9.67.2.3 void l4\_utcb\_exc\_pc\_set( l4\_exc\_regs\_t \* u, l4\_addr\_t pc ) throw() [inline]**

Set the program counter register in the exception state.

**Parameters**

*u* UTCB

*pc* The program counter to set.

Definition at line 94 of file [utcb.h](#).

**9.67.2.4 int l4\_utcb\_exc\_is\_pf( l4\_exc\_regs\_t \* u ) throw() [inline]**

Check whether an exception IPC is a page fault.

**Returns**

0 if not, != 0 if yes

Function to check whether an exception IPC is a page fault, also applies to I/O pagefaults.

Definition at line 104 of file [utcb.h](#).

## 9.68 Virtual Console

Virtual console for simple character based input and output.

Collaboration diagram for Virtual Console:



## Data Structures

- class [L4::Vcon](#)  
*C++ L4 Vcon.*
- struct [l4\\_vcon\\_attr\\_t](#)  
*Vcon attribute structure.*

## Typedefs

- `typedef struct l4_vcon_attr_t l4_vcon_attr_t`  
*Vcon attribute structure.*

## Enumerations

- `enum L4_vcon_write_consts { L4_VCON_WRITE_SIZE = (L4_UTCB_GENERIC_DATA_SIZE - 2) * sizeof(l4_umword_t) }`  
*Constants for l4\_vcon\_write.*
- `enum L4_vcon_i_flags { L4_VCON_INLCR = 000100, L4_VCON_IGNCR = 000200, L4_VCON_ICRNL = 000400 }`  
*Input flags.*
- `enum L4_vcon_o_flags { L4_VCON_ONLCR = 000004, L4_VCON_OCRNL = 000010, L4_VCON_ONLRET = 000040 }`  
*Output flags.*
- `enum L4_vcon_l_flags { L4_VCON_ECHO = 000010 }`  
*Local flags.*
- `enum L4_vcon_ops { L4_VCON_WRITE_OP = 0UL, L4_VCON_SET_ATTR_OP = 2UL, L4_VCON_GET_ATTR_OP = 3UL }`  
*Operations on the vcon objects.*

## Functions

- `l4_mshtag_t l4_vcon_send (l4_cap_idx_t vcon, char const *buf, int size) L4_NOTHROW`  
*Send data to virtual console.*
- `long l4_vcon_write (l4_cap_idx_t vcon, char const *buf, int size) L4_NOTHROW`  
*Write data to virtual console.*
- `int l4_vcon_read (l4_cap_idx_t vcon, char *buf, int size) L4_NOTHROW`  
*Read data from virtual console.*
- `l4_mshtag_t l4_vcon_set_attr (l4_cap_idx_t vcon, l4_vcon_attr_t const *attr) L4_NOTHROW`  
*Set attributes of a Vcon.*
- `l4_mshtag_t l4_vcon_get_attr (l4_cap_idx_t vcon, l4_vcon_attr_t *attr) L4_NOTHROW`  
*Get attributes of a Vcon.*

### 9.68.1 Detailed Description

Virtual console for simple character based input and output. #include <14/sys/vcon.h>  
 Interrupt for read events are provided by the virtual key interrupt.

## 9.68.2 Enumeration Type Documentation

### 9.68.2.1 enum L4\_vcon\_write\_consts

Constants for l4\_vcon\_write.

**Enumerator:**

*L4\_VCON\_WRITE\_SIZE* Maximum size that can be written with one l4\_vcon\_write call.

Definition at line 83 of file [vcon.h](#).

### 9.68.2.2 enum L4\_vcon\_i\_flags

Input flags.

**Enumerator:**

*L4\_VCON\_INLCR* Translate NL to CR.

*L4\_VCON\_IGNCR* Ignore CR.

*L4\_VCON\_ICRNL* Translate CR to NL if L4\_VCON\_IGNCR is not set.

Definition at line 126 of file [vcon.h](#).

### 9.68.2.3 enum L4\_vcon\_o\_flags

Output flags.

**Enumerator:**

*L4\_VCON\_ONLCR* Translate NL to CR-NL.

*L4\_VCON\_OCRNL* Translate CR to NL.

*L4\_VCON\_ONLRET* Do not output CR.

Definition at line 137 of file [vcon.h](#).

### 9.68.2.4 enum L4\_vcon\_l\_flags

Local flags.

**Enumerator:**

*L4\_VCON\_ECHO* Echo input.

Definition at line 148 of file [vcon.h](#).

### 9.68.2.5 enum L4\_vcon\_ops

Operations on the vcon objects.

**Enumerator:**

- L4\_VCON\_WRITE\_OP* Write.
- L4\_VCON\_SET\_ATTR\_OP* Get console attributes.
- L4\_VCON\_GET\_ATTR\_OP* Set console attributes.

Definition at line 197 of file [vcon.h](#).

## 9.68.3 Function Documentation

### 9.68.3.1 l4\_mshtag\_t l4\_vcon\_send ( l4\_cap\_idx\_t vcon, char const \* buf, int size ) [inline]

Send data to virtual console.

**Parameters**

- vcon* Vcon object.
- buf* Pointer to data buffer.
- size* Size of buffer in bytes.

**Returns**

Syscall return tag

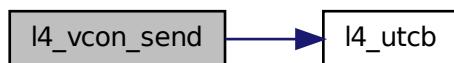
**Note**

Size must not exceed L4\_VCON\_WRITE\_SIZE.

Definition at line 221 of file [vcon.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.68.3.2 long l4\_vcon\_write ( l4\_cap\_idx\_t vcon, char const \* buf, int size ) [inline]

Write data to virtual console.

### Parameters

*vcon* Vcon object.  
*buf* Pointer to data buffer.  
*size* Size of buffer in bytes.

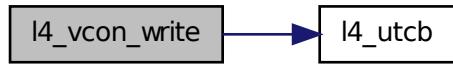
### Returns

Number of bytes written to the virtual console.

Definition at line 242 of file [vcon.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.68.3.3 int l4\_vcon\_read ( l4\_cap\_idx\_t vcon, char \* buf, int size ) [inline]

Read data from virtual console.

### Parameters

*vcon* Vcon object.  
*buf* Pointer to data buffer.  
*size* Size of buffer in bytes.

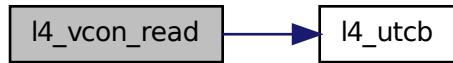
### Returns

Negative error code on error, > size if more to read, size bytes are in the buffer, <= size bytes read

Definition at line 279 of file [vcon.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.68.3.4 `l4_mshtag_t l4_vcon_set_attr ( l4_cap_idx_t vcon, l4_vcon_attr_t const * attr ) [inline]`

Set attributes of a Vcon.

#### Parameters

*vcon* Vcon object.

*attr* Attribute structure.

#### Returns

Syscall return tag

Definition at line 299 of file [vcon.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 9.68.3.5 `l4_mshtag_t l4_vcon_get_attr ( l4_cap_idx_t vcon, l4_vcon_attr_t * attr ) [inline]`

Get attributes of a Vcon.

#### Parameters

*vcon* Vcon object.

#### Return values

*attr* Attribute structure.

#### Returns

Syscall return tag

Definition at line 323 of file [vcon.h](#).

References [l4\\_utcb\(\)](#).

Here is the call graph for this function:



## 9.69 vCPU API

vCPU API

Collaboration diagram for vCPU API:



### Data Structures

- struct [l4\\_vcpu\\_state\\_t](#)  
*State of a vCPU.*
- struct [l4\\_vcpu\\_regs\\_t](#)  
*vCPU registers.*
- struct [l4\\_vcpu\\_ipc\\_regs\\_t](#)  
*vCPU message registers.*

### Typedefs

- typedef struct [l4\\_vcpu\\_state\\_t](#) [l4\\_vcpu\\_state\\_t](#)  
*State of a vCPU.*
- typedef struct [l4\\_vcpu\\_regs\\_t](#) [l4\\_vcpu\\_regs\\_t](#)  
*vCPU registers.*
- typedef struct [l4\\_vcpu\\_ipc\\_regs\\_t](#) [l4\\_vcpu\\_ipc\\_regs\\_t](#)

*vCPU message registers.*

- `typedef struct l4_vcpu_regs_t l4_vcpu_regs_t`  
*vCPU registers.*
- `typedef struct l4_vcpu_ipc_regs_t l4_vcpu_ipc_regs_t`  
*vCPU message registers.*
- `typedef struct l4_vcpu_regs_t l4_vcpu_regs_t`  
*vCPU registers.*
- `typedef struct l4_vcpu_ipc_regs_t l4_vcpu_ipc_regs_t`  
*vCPU message registers.*

## Enumerations

- `enum L4_vcpu_state_flags {  
 L4_VCPU_F_IRQ = 0x01, L4_VCPU_F_PAGE_FAULTS = 0x02, L4_VCPU_F_EXCEPTIONS =  
 0x04, L4_VCPU_F_DEBUG_EXC = 0x08,  
 L4_VCPU_F_USER_MODE = 0x20, L4_VCPU_F_FPU_ENABLED = 0x80 }  
State flags of a vCPU.`
- `enum L4_vcpu_sticky_flags { L4_VCPU_SF_IRQ_PENDING = 0x01 }  
Sticky flags of a vCPU.`
- `enum L4_vcpu_state_offset { L4_VCPU_OFFSET_EXT_STATE = 0x400 }  
Offsets for vCPU state layouts.`

### 9.69.1 Detailed Description

vCPU API

### 9.69.2 Enumeration Type Documentation

#### 9.69.2.1 enum L4\_vcpu\_state\_flags

State flags of a vCPU.

**Enumerator:**

`L4_VCPU_F_IRQ` IRQs (events) enabled.  
`L4_VCPU_F_PAGE_FAULTS` Page faults enabled.  
`L4_VCPU_F_EXCEPTIONS` Exception enabled.  
`L4_VCPU_F_DEBUG_EXC` Debug exception enabled.  
`L4_VCPU_F_USER_MODE` User task will be used.  
`L4_VCPU_F_FPU_ENABLED` FPU enabled.

Definition at line 55 of file `vcpu.h`.

### 9.69.2.2 enum L4\_vcpu\_sticky\_flags

Sticky flags of a vCPU.

**Enumerator:**

**L4\_VCPU\_SF\_IRQ\_PENDING** An event (e.g. IRQ) is pending.

Definition at line 69 of file [vcpu.h](#).

### 9.69.2.3 enum L4\_vcpu\_state\_offset

Offsets for vCPU state layouts.

**Enumerator:**

**L4\_VCPU\_OFFSET\_EXT\_STATE** Offset where extended state begins.

Definition at line 78 of file [vcpu.h](#).

## 9.70 Fiasco-UX Virtual devices

Virtual hardware devices, provided by Fiasco-UX.

Collaboration diagram for Fiasco-UX Virtual devices:



## Data Structures

- struct [l4\\_vhw\\_entry](#)  
*Description of a device.*
- struct [l4\\_vhw\\_descriptor](#)  
*Virtual hardware devices description.*

## Enumerations

- enum [l4\\_vhw\\_entry\\_type](#) { [L4\\_TYPE\\_VHW\\_NONE](#), [L4\\_TYPE\\_VHW\\_FRAMEBUFFER](#), [L4\\_TYPE\\_VHW\\_INPUT](#), [L4\\_TYPE\\_VHW\\_NET](#) }  
*Type of device.*

### 9.70.1 Detailed Description

Virtual hardware devices, provided by Fiasco-UX. #include <14/sys/vhw.h>

### 9.70.2 Enumeration Type Documentation

#### 9.70.2.1 enum l4\_vhw\_entry\_type

Type of device.

**Enumerator:**

**L4\_TYPE\_VHW\_NONE** None entry.

**L4\_TYPE\_VHW\_FRAMEBUFFER** Framebuffer device.

**L4\_TYPE\_VHW\_INPUT** Input device.

**L4\_TYPE\_VHW\_NET** Network device.

Definition at line 44 of file vhw.h.

## 9.71 Memory operations.

Operations for memory access.

Collaboration diagram for Memory operations.:  


```

graph LR
    MO[Memory operations.] --> BA[Base API]
  
```

### Enumerations

- enum **L4\_mem\_op\_widths** { **L4\_MEM\_WIDTH\_1BYTE** = 0, **L4\_MEM\_WIDTH\_2BYTE** = 1, **L4\_MEM\_WIDTH\_4BYTE** = 2 }

*Memory access width definitions.*

### Functions

- unsigned long **l4\_mem\_read** (unsigned long virtaddress, unsigned width)  
*Read memory from kernel privilege level.*
- void **l4\_mem\_write** (unsigned long virtaddress, unsigned width, unsigned long value)  
*Write memory from kernel privilege level.*

### 9.71.1 Detailed Description

Operations for memory access. This module provides functionality to access user task memory from the kernel. This is needed for some devices that are only accessible from privileged processor mode. Only use this when absolutely required. This functionality is only available on the ARM architecture.

```
#include <14/sys/mem_op.h>
```

### 9.71.2 Enumeration Type Documentation

#### 9.71.2.1 enum L4\_mem\_op\_widths

Memory access width definitions.

**Enumerator:**

*L4\_MEM\_WIDTH\_1BYTE* Access one byte (8-bit width).

*L4\_MEM\_WIDTH\_2BYTE* Access two bytes (16-bit width).

*L4\_MEM\_WIDTH\_4BYTE* Access four bytes (32-bit width).

Definition at line 51 of file [mem\\_op.h](#).

### 9.71.3 Function Documentation

#### 9.71.3.1 unsigned long l4\_mem\_read ( unsigned long *virtaddress*, unsigned *width* ) [inline]

Read memory from kernel privilege level.

##### Parameters

*virtaddress* Virtual address in the calling task.

*width* Width of access in bytes in log2,

##### See also

[L4\\_mem\\_op\\_widths](#)

##### Returns

Read value.

Upon a given invalid address or invalid width value the function does nothing.

Definition at line 141 of file [mem\\_op.h](#).

#### 9.71.3.2 void l4\_mem\_write ( unsigned long *virtaddress*, unsigned *width*, unsigned long *value* ) [inline]

Write memory from kernel privilege level.

##### Parameters

*virtaddress* Virtual address in the calling task.

**width** Width of access in bytes in log2 (i.e. allowed values: 0, 1, 2)

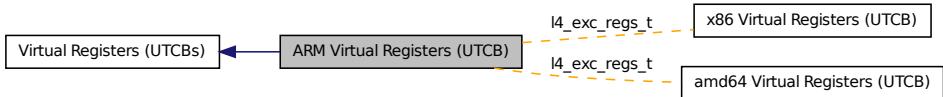
**value** Value to write.

Upon an given invalid address or invalid width value the function does nothing.

Definition at line 147 of file [mem\\_op.h](#).

## 9.72 ARM Virtual Registers (UTCB)

Collaboration diagram for ARM Virtual Registers (UTCB):



## Data Structures

- struct [l4\\_exc\\_regs\\_t](#)

*UTCB structure for exceptions.*

## Typedefs

- typedef struct [l4\\_exc\\_regs\\_t](#) [l4\\_exc\\_regs\\_t](#)

*UTCB structure for exceptions.*

## Enumerations

- enum [L4\\_utcb\\_consts\\_arm](#)

*UTCB constants for ARM.*

## 9.73 VM API for TZ

Virtual Machine API for ARM TrustZone.

Collaboration diagram for VM API for TZ:



## Data Structures

- class [L4::Vm](#)

*Virtual machine.*

- struct [l4\\_vm\\_state](#)

*state structure for TrustZone VMs*

## Functions

- [l4\\_msgtag\\_t l4\\_vm\\_run \( l4\\_cap\\_idx\\_t vm \) L4\\_NO\\_THROW](#)

*Run a VM.*

### 9.73.1 Detailed Description

Virtual Machine API for ARM TrustZone.

### 9.73.2 Function Documentation

#### 9.73.2.1 [l4\\_msgtag\\_t l4\\_vm\\_run \( l4\\_cap\\_idx\\_t vm \) \[inline\]](#)

Run a VM.

##### Parameters

*vm* Capability selector for VM

## 9.74 amd64 Virtual Registers (UTCB)

Collaboration diagram for amd64 Virtual Registers (UTCB):



## Data Structures

- struct [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

## Typedefs

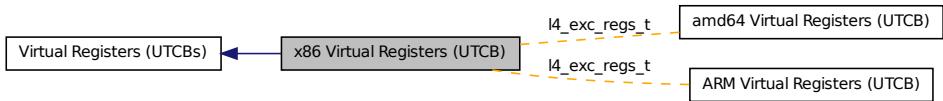
- typedef struct [l4\\_exc\\_regs\\_t](#) [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

## Enumerations

- enum [L4\\_utcb\\_consts\\_amd64](#)  
*UTCB constants for AMD64.*

## 9.75 x86 Virtual Registers (UTCB)

Collaboration diagram for x86 Virtual Registers (UTCB):



## Data Structures

- struct [l4\\_exc\\_regs\\_t](#)  
*UTCB structure for exceptions.*

## Typedefs

- `typedef struct l4_exc_regs_t l4_exc_regs_t`  
*UTCB structure for exceptions.*

## Enumerations

- `enum L4_utcb_consts_x86 {`  
`L4_UTCB_EXCEPTION_REGS_SIZE = 16, L4_UTCB_GENERIC_DATA_SIZE = 63, L4_UTCB_GENERIC_BUFFERS_SIZE = 58, L4_UTCB_MSG_REGS_OFFSET = 0,`  
`L4_UTCB_BUF_REGS_OFFSET = 64 * sizeof(l4_umword_t), L4_UTCB_THREAD_REGS_OFFSET = 123 * sizeof(l4_umword_t), L4_UTCB_INHERIT_FPU = 1UL << 24, L4_UTCB_OFFSET = 512 }`  
*UTCB constants for x86.*

### 9.75.1 Enumeration Type Documentation

#### 9.75.1.1 enum L4\_utcb\_consts\_x86

UTCB constants for x86.

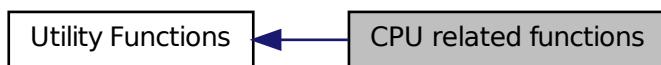
Enumerator:

- `L4_UTCB_EXCEPTION_REGS_SIZE` Number of message registers used for exception IPC.
- `L4_UTCB_GENERIC_DATA_SIZE` Total number of message register (MRs) available.
- `L4_UTCB_GENERIC_BUFFERS_SIZE` Total number of buffer registers (BRs) available.
- `L4_UTCB_MSG_REGS_OFFSET` Offset of MR[0] relative to the UTCB pointer.
- `L4_UTCB_BUF_REGS_OFFSET` Offset of BR[0] relative to the UTCB pointer.
- `L4_UTCB_THREAD_REGS_OFFSET` Offset of TCR[0] relative to the UTCB pointer.
- `L4_UTCB_INHERIT_FPU` BDR flag to accept reception of FPU state.
- `L4_UTCB_OFFSET` Offset of two consecutive UTCBs.

Definition at line 41 of file `utcb.h`.

## 9.76 CPU related functions

Collaboration diagram for CPU related functions:



## Functions

- int [l4util\\_cpu\\_has\\_cpuid](#) (void)  
*Check whether the CPU supports the "cpuid" instruction.*
- unsigned int [l4util\\_cpu\\_capabilities](#) (void)  
*Returns the CPU capabilities if the "cpuid" instruction is available.*
- unsigned int [l4util\\_cpu\\_capabilities\\_nocheck](#) (void)  
*Returns the CPU capabilities.*
- void [l4util\\_cpu\\_cpuid](#) (unsigned long mode, unsigned long \*eax, unsigned long \*ebx, unsigned long \*ecx, unsigned long \*edx)  
*Generic CPUID access function.*

### 9.76.1 Function Documentation

#### 9.76.1.1 int l4util\_cpu\_has\_cpuid ( void ) [inline]

Check whether the CPU supports the "cpuid" instruction.

##### Returns

1 if it has, 0 if it has not

Definition at line 66 of file [cpu.h](#).

Referenced by [l4util\\_cpu\\_capabilities\(\)](#).

Here is the caller graph for this function:



#### 9.76.1.2 unsigned int l4util\_cpu\_capabilities ( void ) [inline]

Returns the CPU capabilities if the "cpuid" instruction is available.

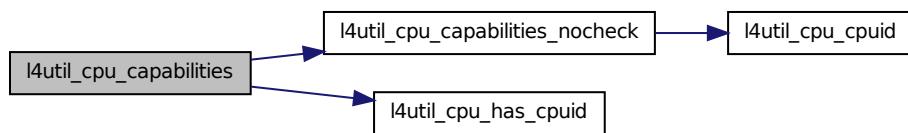
##### Returns

CPU capabilities if the "cpuid" instruction is available, 0 if the "cpuid" instruction is not supported.

Definition at line 114 of file [cpu.h](#).

References [l4util\\_cpu\\_capabilities\\_nocheck\(\)](#), and [l4util\\_cpu\\_has\\_cpuid\(\)](#).

Here is the call graph for this function:



### 9.76.1.3 unsigned int l4util\_cpu\_capabilities\_nocheck ( void ) [inline]

Returns the CPU capabilities.

#### Returns

CPU capabilities.

Definition at line 103 of file [cpu.h](#).

References [l4util\\_cpu\\_cpuid\(\)](#).

Referenced by [l4util\\_cpu\\_capabilities\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



## 9.77 Functions to manipulate the local IDT

Collaboration diagram for Functions to manipulate the local IDT:



### Data Structures

- struct [l4util\\_idt\\_desc\\_t](#)  
*IDT entry.*
- struct [l4util\\_idt\\_header\\_t](#)  
*Header of an IDT table.*

## 9.78 Timestamp Counter

Collaboration diagram for Timestamp Counter:



### Files

- file [rdtsc.h](#)  
*time stamp counter related functions*
- file [rdtsc.h](#)  
*time stamp counter related functions*

## Defines

- `#define L4_TSC_INIT_AUTO 0`  
*Automatic init.*
- `#define L4_TSC_INIT_KERNEL 1`  
*Initialized by kernel.*
- `#define L4_TSC_INIT_CALIBRATE 2`  
*Initialized by user-level.*
- `#define L4_TSC_INIT_AUTO 0`  
*Automatic init.*
- `#define L4_TSC_INIT_KERNEL 1`  
*Initialized by kernel.*
- `#define L4_TSC_INIT_CALIBRATE 2`  
*Initialized by user-level.*

## Functions

- `l4_cpu_time_t l4_rdtsc (void)`  
*Read current value of CPU-internal time stamp counter.*
- `l4_uint32_t l4_rdtsc_32 (void)`  
*Read the least significant 32 bit of the TSC.*
- `l4_cpu_time_t l4_rdpmc (int nr)`  
*Return current value of CPU-internal performance measurement counter.*
- `l4_uint32_t l4_rdpmc_32 (int nr)`  
*Return the least significant 32 bit of a performance counter.*
- `l4_uint64_t l4_tsc_to_ns (l4_cpu_time_t tsc)`  
*Convert time stamp to ns value.*
- `l4_uint64_t l4_tsc_to_us (l4_cpu_time_t tsc)`  
*Convert time stamp into micro seconds value.*
- `void l4_tsc_to_s_and_ns (l4_cpu_time_t tsc, l4_uint32_t *s, l4_uint32_t *ns)`  
*Convert time stamp to s.ns value.*
- `l4_cpu_time_t l4_ns_to_tsc (l4_uint64_t ns)`  
*Convert nano seconds into CPU ticks.*
- `void l4_busy_wait_ns (l4_uint64_t ns)`  
*Wait busy for a small amount of time.*

- `void l4_busy_wait_us (l4_uint64_t us)`  
*Wait busy for a small amount of time.*
- `l4_uint32_t l4_calibrate_tsc (l4_kernel_info_t *kip)`  
*Calibrate scalers for time stamp calculations.*
- `l4_uint32_t l4_tsc_init (int constraint, l4_kernel_info_t *kip)`  
*Initialize scaler for TSC calicltions.*
- `l4_uint32_t l4_get_hz (void)`  
*Get CPU frequency in Hz.*

## 9.78.1 Function Documentation

### 9.78.1.1 `l4_cpu_time_t l4_rdtsc ( void ) [inline]`

Read current value of CPU-internal time stamp counter.

#### Returns

64-bit time stamp

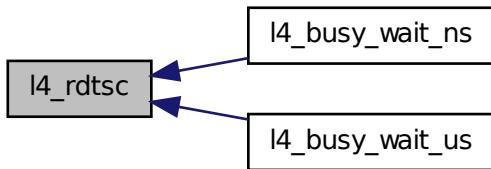
#### Examples:

`examples/sys/aliens/main.c`.

Definition at line 185 of file `rdtsc.h`.

Referenced by `l4_busy_wait_ns()`, and `l4_busy_wait_us()`.

Here is the caller graph for this function:



### 9.78.1.2 `l4_uint32_t l4_rdtsc_32 ( void ) [inline]`

Read the lest significant 32 bit of the TSC.

Useful for smaller differences, needs less cycles.

Definition at line 246 of file `rdtsc.h`.

**9.78.1.3 l4\_cpu\_time\_t l4\_rdpmc( int nr ) [inline]**

Return current value of CPU-internal performance measurement counter.

**Parameters**

*nr* Number of counter (0 or 1)

**Returns**

64-bit PMC

Definition at line 205 of file [rdtsc.h](#).

**9.78.1.4 l4\_uint32\_t l4\_rdpmc\_32( int nr ) [inline]**

Return the least significant 32 bit of a performance counter.

Useful for smaller differences, needs less cycles.

Definition at line 227 of file [rdtsc.h](#).

**9.78.1.5 l4\_uint64\_t l4\_tsc\_to\_ns( l4\_cpu\_time\_t tsc ) [inline]**

Convert time stamp to ns value.

**Parameters**

*tsc* time value in CPU ticks

**Returns**

time value in ns

**Examples:**

[examples/sys/aliens/main.c](#).

Definition at line 260 of file [rdtsc.h](#).

**9.78.1.6 l4\_uint64\_t l4\_tsc\_to\_us( l4\_cpu\_time\_t tsc ) [inline]**

Convert time stamp into micro seconds value.

**Parameters**

*tsc* time value in CPU ticks

**Returns**

time value in micro seconds

Definition at line 274 of file [rdtsc.h](#).

**9.78.1.7 void l4\_tsc\_to\_s\_and\_ns ( l4\_cpu\_time\_t tsc, l4\_uint32\_t \* s, l4\_uint32\_t \* ns ) [inline]**

Convert time stamp to s.ns value.

**Parameters**

*tsc* time value in CPU ticks

**Return values**

*s* seconds

*ns* nano seconds

Definition at line 288 of file [rdtsc.h](#).

**9.78.1.8 l4\_cpu\_time\_t l4\_ns\_to\_tsc ( l4\_uint64\_t ns ) [inline]**

Convert nano seconds into CPU ticks.

**Parameters**

*ns* nano seconds

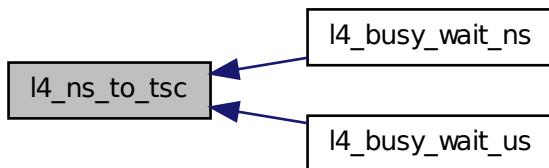
**Returns**

CPU ticks

Definition at line 303 of file [rdtsc.h](#).

Referenced by [l4\\_busy\\_wait\\_ns\(\)](#), and [l4\\_busy\\_wait\\_us\(\)](#).

Here is the caller graph for this function:



**9.78.1.9 void l4\_busy\_wait\_ns ( l4\_uint64\_t ns ) [inline]**

Wait busy for a small amount of time.

**Parameters**

*ns* nano seconds to wait

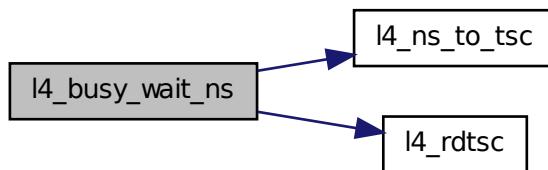
**Attention**

Not intended for any use!

Definition at line 317 of file [rdtsc.h](#).

References [l4\\_ns\\_to\\_tsc\(\)](#), and [l4\\_rdtsc\(\)](#).

Here is the call graph for this function:

**9.78.1.10 void l4\_busy\_wait\_us ( l4\_uint64\_t us ) [inline]**

Wait busy for a small amount of time.

**Parameters**

*us* micro seconds to wait

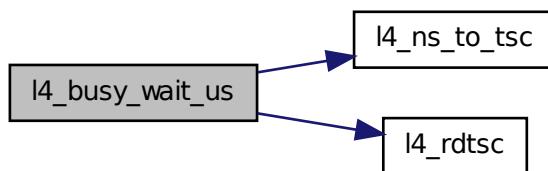
**Attention**

Not intended for any use!

Definition at line 327 of file [rdtsc.h](#).

References [l4\\_ns\\_to\\_tsc\(\)](#), and [l4\\_rdtsc\(\)](#).

Here is the call graph for this function:



### 9.78.1.11 `l4_uint32_t l4_calibrate_tsc ( l4_kernel_info_t * kip ) [inline]`

Calibrate scalers for time stamp calculations.

Determine some scalers to be able to convert between real time and CPU ticks. This test uses channel 0 of the PIT (i8254) or the kernel KIP, depending on availability. Just calls `l4_tsc_init(L4_TSC_INIT_AUTO)`.

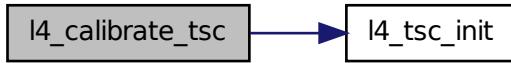
#### Examples:

[examples/sys/aliens/main.c](#).

Definition at line 179 of file [rdtsc.h](#).

References [l4\\_tsc\\_init\(\)](#), and [L4\\_TSC\\_INIT\\_AUTO](#).

Here is the call graph for this function:



### 9.78.1.12 `l4_uint32_t l4_tsc_init ( int constraint, l4_kernel_info_t * kip )`

Initialize scalar for TSC calicatlions.

Initialize the scalers needed by [l4\\_tsc\\_to\\_ns\(\)](#)/[l4\\_ns\\_to\\_tsc\(\)](#) and so on. Current versions of Fiasco export these scalers from kernel into userland. The programmer may decide whether he allows to use these scalers or if an calibration should be performed.

#### Parameters

**constraint** programmers constraint:

- [L4\\_TSC\\_INIT\\_AUTO](#) if the kernel exports the scalers then use them. If not, perform calibration using channel 0 of the PIT (i8254). The latter case may lead into short (unpredictable) periods where interrupts are disabled.
- [L4\\_TSC\\_INIT\\_KERNEL](#) depend on retrieving the scalers from kernel. If the scalers are not available, return 0.
- [L4\\_TSC\\_INIT\\_CALIBRATE](#) Ignore possible scalers exported by the scaler, instead insist on calibration using the PIT.

**kip** KIP pointer

#### Returns

0 on error (no scalers exported by kernel, calibrating failed ...) otherwise returns  $(2^{32} / (\text{tsc per tsec}))$ . This value has the same semantics as the value returned by the `calibrate_delay_loop()` function of the Linux kernel.

Initialize the scalers needed by [l4\\_tsc\\_to\\_ns\(\)](#)/[l4\\_ns\\_to\\_tsc\(\)](#) and so on. Current versions of Fiasco export these scalers from kernel into userland. The programmer may decide whether he allows to use these scalers or if an calibration should be performed.

### Parameters

**constraint** programmers constraint:

- [L4\\_TSC\\_INIT\\_AUTO](#) if the kernel exports the scalers then use them. If not, perform calibration using channel 0 of the PIT (i8254). The latter case may lead into short (unpredictable) periods where interrupts are disabled.
- [L4\\_TSC\\_INIT\\_KERNEL](#) depend on retrieving the scalers from kernel. If the scalers are not available, return 0.
- [L4\\_TSC\\_INIT\\_CALIBRATE](#) Ignore possible scalers exported by the scaler, instead insist on calibration using the PIT.

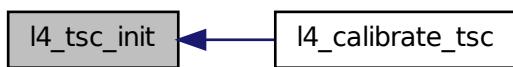
**kip** KIP pointer

### Returns

0 on error (no scalers exported by kernel, calibrating failed ...) otherwise returns  $(2^{32} / (\text{tsc per } \hat{\text{sec}}))$ . This value has the same semantics as the value returned by the `calibrate_delay_loop()` function of the Linux kernel.

Referenced by [l4\\_calibrate\\_tsc\(\)](#).

Here is the caller graph for this function:



### 9.78.1.13 [l4\\_uint32\\_t l4\\_get\\_hz\( void \)](#)

Get CPU frequency in Hz.

### Returns

frequency in Hz

## 9.79 Atomic Instructions

Collaboration diagram for Atomic Instructions:



### Files

- file **atomic.h**  
*atomic operations header and generic implementations*

### Functions

- int **l4util\_cmpxchg64** (volatile **l4\_uint64\_t** \*dest, **l4\_uint64\_t** cmp\_val, **l4\_uint64\_t** new\_val)  
*Atomic compare and exchange (64 bit version).*
- int **l4util\_cmpxchg32** (volatile **l4\_uint32\_t** \*dest, **l4\_uint32\_t** cmp\_val, **l4\_uint32\_t** new\_val)  
*Atomic compare and exchange (32 bit version).*
- int **l4util\_cmpxchg16** (volatile **l4\_uint16\_t** \*dest, **l4\_uint16\_t** cmp\_val, **l4\_uint16\_t** new\_val)  
*Atomic compare and exchange (16 bit version).*
- int **l4util\_cmpxchg8** (volatile **l4\_uint8\_t** \*dest, **l4\_uint8\_t** cmp\_val, **l4\_uint8\_t** new\_val)  
*Atomic compare and exchange (8 bit version).*
- int **l4util\_cmpxchg** (volatile **l4\_umword\_t** \*dest, **l4\_umword\_t** cmp\_val, **l4\_umword\_t** new\_val)  
*Atomic compare and exchange (machine wide fields).*
- **l4\_uint32\_t l4util\_xchg32** (volatile **l4\_uint32\_t** \*dest, **l4\_uint32\_t** val)  
*Atomic exchange (32 bit version).*
- **l4\_uint16\_t l4util\_xchg16** (volatile **l4\_uint16\_t** \*dest, **l4\_uint16\_t** val)  
*Atomic exchange (16 bit version).*
- **l4\_uint8\_t l4util\_xchg8** (volatile **l4\_uint8\_t** \*dest, **l4\_uint8\_t** val)  
*Atomic exchange (8 bit version).*
- **l4\_umword\_t l4util\_xchg** (volatile **l4\_umword\_t** \*dest, **l4\_umword\_t** val)  
*Atomic exchange (machine wide fields).*

- void `l4util_atomic_add` (volatile long \*dest, long val)

*Atomic add.*

- void `l4util_atomic_inc` (volatile long \*dest)

*Atomic increment.*

## Atomic add/sub/and/or (8,16,32 bit version) without result

- void `l4util_add8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- void `l4util_add16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- void `l4util_add32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)
- void `l4util_sub8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- void `l4util_sub16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- void `l4util_sub32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)
- void `l4util_and8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- void `l4util_and16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- void `l4util_and32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)
- void `l4util_or8` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- void `l4util_or16` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- void `l4util_or32` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)

## Atomic add/sub/and/or operations (8,16,32 bit) with result

- `l4_uint8_t l4util_add8_res` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- `l4_uint16_t l4util_add16_res` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- `l4_uint32_t l4util_add32_res` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)
- `l4_uint8_t l4util_sub8_res` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- `l4_uint16_t l4util_sub16_res` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- `l4_uint32_t l4util_sub32_res` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)
- `l4_uint8_t l4util_and8_res` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- `l4_uint16_t l4util_and16_res` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- `l4_uint32_t l4util_and32_res` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)
- `l4_uint8_t l4util_or8_res` (volatile `l4_uint8_t` \*dest, `l4_uint8_t` val)
- `l4_uint16_t l4util_or16_res` (volatile `l4_uint16_t` \*dest, `l4_uint16_t` val)
- `l4_uint32_t l4util_or32_res` (volatile `l4_uint32_t` \*dest, `l4_uint32_t` val)

## Atomic inc/dec (8,16,32 bit) without result

- void `l4util_inc8` (volatile `l4_uint8_t` \*dest)
- void `l4util_inc16` (volatile `l4_uint16_t` \*dest)
- void `l4util_inc32` (volatile `l4_uint32_t` \*dest)
- void `l4util_dec8` (volatile `l4_uint8_t` \*dest)
- void `l4util_dec16` (volatile `l4_uint16_t` \*dest)
- void `l4util_dec32` (volatile `l4_uint32_t` \*dest)

## Atomic inc/dec (8,16,32 bit) with result

- `l4_uint8_t l4util_inc8_res` (volatile `l4_uint8_t` \*dest)
- `l4_uint16_t l4util_inc16_res` (volatile `l4_uint16_t` \*dest)
- `l4_uint32_t l4util_inc32_res` (volatile `l4_uint32_t` \*dest)
- `l4_uint8_t l4util_dec8_res` (volatile `l4_uint8_t` \*dest)
- `l4_uint16_t l4util_dec16_res` (volatile `l4_uint16_t` \*dest)
- `l4_uint32_t l4util_dec32_res` (volatile `l4_uint32_t` \*dest)

### 9.79.1 Function Documentation

#### 9.79.1.1 int l4util\_cmpxchg64 ( volatile l4\_uint64\_t \* dest, l4\_uint64\_t cmp\_val, l4\_uint64\_t new\_val ) [inline]

Atomic compare and exchange (64 bit version).

##### Parameters

*dest* destination operand

*cmp\_val* compare value

*new\_val* new value for dest

##### Returns

0 if comparison failed, 1 otherwise

Compare the value in *dest* with *cmp\_val*, if equal set *dest* to *new\_val*

Definition at line 361 of file [atomic.h](#).

#### 9.79.1.2 int l4util\_cmpxchg32 ( volatile l4\_uint32\_t \* dest, l4\_uint32\_t cmp\_val, l4\_uint32\_t new\_val ) [inline]

Atomic compare and exchange (32 bit version).

##### Parameters

*dest* destination operand

*cmp\_val* compare value

*new\_val* new value for dest

##### Returns

0 if comparison failed, !=0 otherwise

Compare the value in *dest* with *cmp\_val*, if equal set *dest* to *new\_val*

Definition at line 24 of file [atomic\\_arch.h](#).

**9.79.1.3 int l4util\_cmpxchg16 ( volatile l4\_uint16\_t \* dest, l4\_uint16\_t cmp\_val, l4\_uint16\_t new\_val ) [inline]**

Atomic compare and exchange (16 bit version).

**Parameters**

*dest* destination operand

*cmp\_val* compare value

*new\_val* new value for dest

**Returns**

0 if comparison failed, !=0 otherwise

Compare the value in *dest* with *cmp\_val*, if equal set *dest* to *new\_val*

Definition at line 375 of file [atomic.h](#).

**9.79.1.4 int l4util\_cmpxchg8 ( volatile l4\_uint8\_t \* dest, l4\_uint8\_t cmp\_val, l4\_uint8\_t new\_val ) [inline]**

Atomic compare and exchange (8 bit version).

**Parameters**

*dest* destination operand

*cmp\_val* compare value

*new\_val* new value for dest

**Returns**

0 if comparison failed, !=0 otherwise

Compare the value in *dest* with *cmp\_val*, if equal set *dest* to *new\_val*

Definition at line 368 of file [atomic.h](#).

**9.79.1.5 int l4util\_cmpxchg ( volatile l4\_umword\_t \* dest, l4\_umword\_t cmp\_val, l4\_umword\_t new\_val ) [inline]**

Atomic compare and exchange (machine wide fields).

**Parameters**

*dest* destination operand

*cmp\_val* compare value

*new\_val* new value for dest

**Returns**

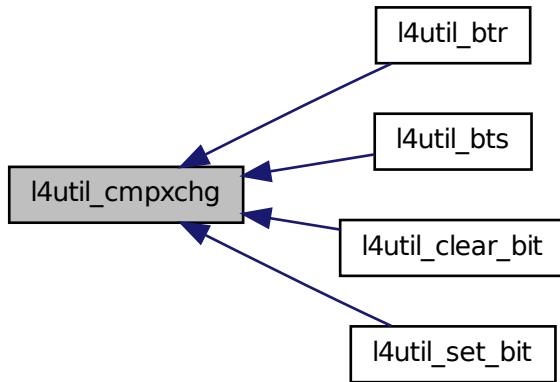
0 if comparison failed, 1 otherwise

Compare the value in *dest* with *cmp\_val*, if equal set *dest* to *new\_val*

Definition at line 382 of file [atomic.h](#).

Referenced by [l4util\\_btr\(\)](#), [l4util\\_bts\(\)](#), [l4util\\_clear\\_bit\(\)](#), and [l4util\\_set\\_bit\(\)](#).

Here is the caller graph for this function:



### **9.79.1.6 l4\_uint32\_t l4util\_xchg32 ( volatile l4\_uint32\_t \* dest, l4\_uint32\_t val ) [inline]**

Atomic exchange (32 bit version).

#### **Parameters**

*dest* destination operand

*val* new value for dest

#### **Returns**

old value at destination

Definition at line 389 of file [atomic.h](#).

### **9.79.1.7 l4\_uint16\_t l4util\_xchg16 ( volatile l4\_uint16\_t \* dest, l4\_uint16\_t val ) [inline]**

Atomic exchange (16 bit version).

#### **Parameters**

*dest* destination operand

*val* new value for dest

#### **Returns**

old value at destination

Definition at line 395 of file atomic.h.

#### 9.79.1.8 `l4_uint8_t l4util_xchg8( volatile l4_uint8_t * dest, l4_uint8_t val ) [inline]`

Atomic exchange (8 bit version).

##### Parameters

*dest* destination operand

*val* new value for dest

##### Returns

old value at destination

Definition at line 401 of file atomic.h.

#### 9.79.1.9 `l4_umword_t l4util_xchg( volatile l4_umword_t * dest, l4_umword_t val ) [inline]`

Atomic exchange (machine wide fields).

##### Parameters

*dest* destination operand

*val* new value for dest

##### Returns

old value at destination

Definition at line 407 of file atomic.h.

#### 9.79.1.10 `void l4util_add8( volatile l4_uint8_t * dest, l4_uint8_t val ) [inline]`

##### Parameters

*dest* destination operand

*val* value to add/sub/and/or

Definition at line 413 of file atomic.h.

#### 9.79.1.11 `l4_uint8_t l4util_add8_res( volatile l4_uint8_t * dest, l4_uint8_t val ) [inline]`

##### Parameters

*dest* destination operand

*val* value to add/sub/and/or

##### Returns

res

Definition at line 486 of file atomic.h.

### 9.79.1.12 void l4util\_inc8 ( volatile l4\_uint8\_t \* dest ) [inline]

#### Parameters

*dest* destination operand

Definition at line 311 of file [atomic.h](#).

### 9.79.1.13 l4\_uint8\_t l4util\_inc8\_res ( volatile l4\_uint8\_t \* dest ) [inline]

#### Parameters

*dest* destination operand

#### Returns

res

Definition at line 337 of file [atomic.h](#).

### 9.79.1.14 void l4util\_atomic\_add ( volatile long \* dest, long val ) [inline]

Atomic add.

#### Parameters

*dest* destination operand

*val* value to add

Definition at line 54 of file [atomic\\_arch.h](#).

### 9.79.1.15 void l4util\_atomic\_inc ( volatile long \* dest ) [inline]

Atomic increment.

#### Parameters

*dest* destination operand

Definition at line 61 of file [atomic\\_arch.h](#).

## 9.80 Internal functions

Collaboration diagram for Internal functions:



## Functions

- void **base64\_encode** (const char \*infile, unsigned int in\_size, char \*\*outfile)  
*base-64-encode string infile*
- void **base64\_decode** (const char \*infile, unsigned int in\_size, char \*\*outfile)  
*decode base-64-encoded string infile*

## 9.81 Bit Manipulation

Collaboration diagram for Bit Manipulation:



## Files

- file **bitops.h**  
*bit manipulation functions*

## Functions

- void **l4util\_set\_bit** (int b, volatile **l4\_umword\_t** \*dest)  
*Set bit in memory.*
- void **l4util\_clear\_bit** (int b, volatile **l4\_umword\_t** \*dest)  
*Clear bit in memory.*
- void **l4util\_complement\_bit** (int b, volatile **l4\_umword\_t** \*dest)  
*Complement bit in memory.*
- int **l4util\_test\_bit** (int b, const volatile **l4\_umword\_t** \*dest)  
*Test bit (return value of bit).*
- int **l4util\_bts** (int b, volatile **l4\_umword\_t** \*dest)  
*Bit test and set.*
- int **l4util\_btr** (int b, volatile **l4\_umword\_t** \*dest)  
*Bit test and reset.*

- int l4util\_btc (int b, volatile l4\_umword\_t \*dest)  
*Bit test and complement.*
- int l4util\_bsr (l4\_umword\_t word)  
*Bit scan reverse.*
- int l4util\_bsf (l4\_umword\_t word)  
*Bit scan forward.*
- int l4util\_find\_first\_set\_bit (const void \*dest, l4\_size\_t size)  
*Find the first set bit in a memory region.*
- int l4util\_find\_first\_zero\_bit (const void \*dest, l4\_size\_t size)  
*Find the first zero bit in a memory region.*
- int l4util\_next\_power2 (const unsigned long val)  
*Find the next power of 2 for a given number.*

### 9.81.1 Function Documentation

#### 9.81.1.1 void l4util\_set\_bit ( int b, volatile l4\_umword\_t \* dest ) [inline]

Set bit in memory.

##### Parameters

- b* bit position  
*dest* destination operand

Definition at line 231 of file [bitops.h](#).

References [l4util\\_cmpxchg\(\)](#).

Here is the call graph for this function:



### 9.81.1.2 void l4util\_clear\_bit( int *b*, volatile l4\_umword\_t \* *dest* ) [inline]

Clear bit in memory.

#### Parameters

*b* bit position

*dest* destination operand

Definition at line 250 of file [bitops.h](#).

References [l4util\\_cmpxchg\(\)](#).

Here is the call graph for this function:



### 9.81.1.3 void l4util\_complement\_bit( int *b*, volatile l4\_umword\_t \* *dest* ) [inline]

Complement bit in memory.

#### Parameters

*b* bit position

*dest* destination operand

Definition at line 394 of file [bitops.h](#).

### 9.81.1.4 int l4util\_test\_bit( int *b*, const volatile l4\_umword\_t \* *dest* ) [inline]

Test bit (return value of bit).

#### Parameters

*b* bit position

*dest* destination operand

#### Returns

Value of bit *b*.

Definition at line 268 of file [bitops.h](#).

### 9.81.1.5 int l4util\_bts ( int *b*, volatile l4\_umword\_t \* *dest* ) [inline]

Bit test and set.

#### Parameters

*b* bit position

*dest* destination operand

#### Returns

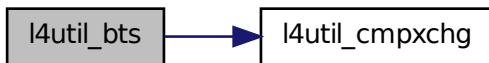
Old value of bit *b*.

Set the *b* bit of *dest* to 1 and return the old value.

Definition at line 291 of file [bitops.h](#).

References [l4util\\_cmpxchg\(\)](#).

Here is the call graph for this function:



### 9.81.1.6 int l4util\_btr ( int *b*, volatile l4\_umword\_t \* *dest* ) [inline]

Bit test and reset.

#### Parameters

*b* bit position

*dest* destination operand

#### Returns

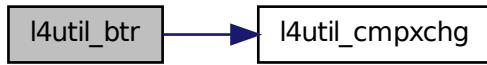
Old value of bit *b*.

Reset bit *b* and return old value.

Definition at line 313 of file [bitops.h](#).

References [l4util\\_cmpxchg\(\)](#).

Here is the call graph for this function:



#### 9.81.1.7 int l4util\_btc ( int *b*, volatile l4\_umword\_t \* *dest* ) [inline]

Bit test and complement.

##### Parameters

*b* bit position

*dest* destination operand

##### Returns

Old value of bit *b*.

Complement bit *b* and return old value.

Definition at line 435 of file [bitops.h](#).

#### 9.81.1.8 int l4util\_bsr ( l4\_umword\_t *word* ) [inline]

Bit scan reverse.

##### Parameters

*word* value (machine size)

##### Returns

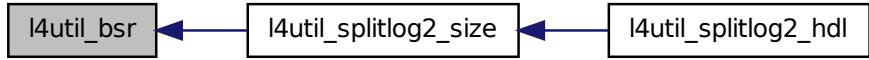
index of most significant set bit in word, -1 if no bit is set (*word* == 0)

"bit scan reverse", find most significant set bit in word (-> LOG2(*word*))

Definition at line 334 of file [bitops.h](#).

Referenced by [l4util\\_splitlog2\\_size\(\)](#).

Here is the caller graph for this function:



### 9.81.1.9 int l4util\_bsf( l4\_umword\_t word ) [inline]

Bit scan forward.

#### Parameters

*word* value (machine size)

#### Returns

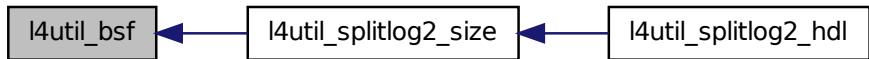
index of least significant bit set in word, -1 if no bit is set (word == 0)

"bit scan forward", find least significant bit set in word.

Definition at line 351 of file [bitops.h](#).

Referenced by [l4util\\_splitlog2\\_size\(\)](#).

Here is the caller graph for this function:



### 9.81.1.10 int l4util\_find\_first\_set\_bit( const void \* dest, l4\_size\_t size ) [inline]

Find the first set bit in a memory region.

#### Parameters

*dest* bit string

*size* size of string in bits (must be a multiple of 32!)

#### Returns

number of the first set bit, >= size if no bit is set

Definition at line 441 of file [bitops.h](#).

**9.81.1.11 int l4util\_find\_first\_zero\_bit ( const void \* dest, l4\_size\_t size ) [inline]**

Find the first zero bit in a memory region.

**Parameters**

*dest* bit string

*size* size of string in bits (must be a multiple of 32!)

**Returns**

number of the first zero bit, >= size if no bit is set

Definition at line 368 of file [bitops.h](#).

**9.81.1.12 int l4util\_next\_power2 ( const unsigned long val ) [inline]**

Find the next power of 2 for a given number.

**Parameters**

*val* initial value

**Returns**

next-highest power of 2

Definition at line 408 of file [bitops.h](#).

## 9.82 ELF binary format

Functions and types related to ELF binaries.

Collaboration diagram for ELF binary format:



## Data Structures

- struct [Elf32\\_Ehdr](#)  
*ELF32 header.*
- struct [Elf64\\_Ehdr](#)

*ELF64 header.*

- struct [Elf32\\_Shdr](#)

*ELF32 section header - figure 1-9, page 1-9.*

- struct [Elf64\\_Shdr](#)

*ELF64 section header.*

- struct [Elf32\\_Phdr](#)

*ELF32 program header.*

- struct [Elf64\\_Phdr](#)

*ELF64 program header.*

- struct [Elf32\\_Dyn](#)

*ELF32 dynamic entry.*

- struct [Elf64\\_Dyn](#)

*ELF64 dynamic entry.*

- struct [Elf32\\_Sym](#)

*ELF32 symbol table entry.*

- struct [Elf64\\_Sym](#)

*ELF64 symbol table entry.*

## Files

- file [elf.h](#)

*ELF definition.*

## Defines

- #define [EI\\_NIDENT](#) 16

*number of characters*

- #define [EI\\_CLASS](#) 4

*ELF class byte index.*

- #define [EI\\_CLASS](#) 4

*ELF class byte index.*

- #define [ELFCLASSNONE](#) 0

*Invalid ELF class.*

- #define [ELFCLASSNONE](#) 0

*Invalid ELF class.*

- #define **ELFCLASS32** 1  
*32-bit objects*
- #define **ELFCLASS64** 2  
*64-bit objects*
- #define **ELFCLASSNUM** 3  
*Mask for 32-bit or 64-bit class.*
- #define **EI\_DATA** 5  
*Data encoding byte index.*
- #define **EI\_DATA** 5  
*Data encoding byte index.*
- #define **ELFDATANONE** 0  
*Invalid data encoding.*
- #define **ELFDATANONE** 0  
*Invalid data encoding.*
- #define **ELFDATA2LSB** 1  
*2's complement, little endian*
- #define **ELFDATA2LSB** 1  
*2's complement, little endian*
- #define **ELFDATA2MSB** 2  
*2's complement, big endian*
- #define **ELFDATA2MSB** 2  
*2's complement, big endian*
- #define **EI\_VERSION** 6  
*File version byte index.*
- #define **EI\_VERSION** 6  
*File version byte index.*
- #define **EI\_OSABI** 7  
*OS ABI identification.*
- #define **EI\_OSABI** 7  
*OS ABI identification.*
- #define **ELFOSABI\_NONE** 0  
*UNIX System V ABI.*
- #define **ELFOSABI\_SYSV** 0

*Alias.*

- #define **ELFOSABI\_SYSV** 0

*Alias.*

- #define **ELFOSABI\_HPUX** 1

*HP-UX.*

- #define **ELFOSABI\_HPUX** 1

*HP-UX.*

- #define **ELFOSABI\_NETBSD** 2

*NetBSD.*

- #define **ELFOSABI\_LINUX** 3

*Linux.*

- #define **ELFOSABI\_SOLARIS** 6

*Sun Solaris.*

- #define **ELFOSABI\_AIX** 7

*IBM AIX.*

- #define **ELFOSABI\_IRIX** 8

*SGI Irix.*

- #define **ELFOSABI\_FREEBSD** 9

*FreeBSD.*

- #define **ELFOSABI\_TRU64** 10

*Compaq TRU64 UNIX.*

- #define **ELFOSABI\_MODESTO** 11

*Novell Modesto.*

- #define **ELFOSABI\_OPENBSD** 12

*OpenBSD.*

- #define **ELFOSABI\_ARM** 97

*ARM.*

- #define **ELFOSABI\_STANDALONE** 255

*Standalone (embedded) application.*

- #define **ELFOSABI\_STANDALONE** 255

*Standalone (embedded) application.*

- #define **EI\_ABIVERSION** 8

*ABI version.*

- #define **EL\_ABIVERSION** 8  
*ABI version.*
- #define **EL\_PAD** 9  
*Byte index of padding bytes.*
- #define **EL\_PAD** 9  
*Byte index of padding bytes.*
- #define **ET\_NONE** 0  
*no file type*
- #define **ET\_REL** 1  
*relocatable file*
- #define **ET\_EXEC** 2  
*executable file*
- #define **ET\_DYN** 3  
*shared object file*
- #define **ET\_CORE** 4  
*core file*
- #define **ET\_LOPROC** 0xff00  
*processor-specific*
- #define **ET\_HIPROC** 0xffff  
*processor-specific*
- #define **EM\_NONE** 0  
*no machine*
- #define **EM\_M32** 1  
*AT&T WE 32100.*
- #define **EM\_SPARC** 2  
*SPARC.*
- #define **EM\_386** 3  
*Intel 80386.*
- #define **EM\_68K** 4  
*Motorola 68000.*
- #define **EM\_88K** 5  
*Motorola 88000.*
- #define **EM\_860** 7  
*Intel 80860.*

- #define **EM\_MIPS** 8  
*MIPS RS3000 big-endian.*
- #define **EM\_MIPS\_RS4\_BE** 10  
*MIPS RS4000 big-endian.*
- #define **EM\_SPARC64** 11  
*SPARC 64-bit.*
- #define **EM\_PARISC** 15  
*HP PA-RISC.*
- #define **EM\_VPP500** 17  
*Fujitsu VPP500.*
- #define **EM\_SPARC32PLUS** 18  
*Sun's V8plus.*
- #define **EM\_960** 19  
*Intel 80960.*
- #define **EM\_PPC** 20  
*PowerPC.*
- #define **EM\_V800** 36  
*NEC V800.*
- #define **EM\_FR20** 37  
*Fujitsu FR20.*
- #define **EM\_RH32** 38  
*TRW RH-32.*
- #define **EM\_RCE** 39  
*Motorola RCE.*
- #define **EM\_ARM** 40  
*Advanced RISC Machines ARM.*
- #define **EM\_ALPHA** 41  
*Digital Alpha.*
- #define **EM\_SH** 42  
*Hitachi SuperH.*
- #define **EM\_SPARCV9** 43  
*SPARC v9 64-bit.*
- #define **EM\_TRICORE** 44

*Siemens Tricore embedded processor.*

- #define **EM\_ARC** 45  
*Argonaut RISC Core, Argonaut Techn Inc.*
- #define **EM\_H8\_300** 46  
*Hitachi H8/300.*
- #define **EM\_H8\_300H** 47  
*Hitachi H8/300H.*
- #define **EM\_H8S** 48  
*Hitachi H8/S.*
- #define **EM\_H8\_500** 49  
*Hitachi H8/500.*
- #define **EM\_IA\_64** 50  
*HP/Intel IA-64.*
- #define **EM\_MIPS\_X** 51  
*Stanford MIPS-X.*
- #define **EM\_COLDFIRE** 52  
*Motorola Coldfire.*
- #define **EM\_68HC12** 53  
*Motorola M68HC12.*
- #define **EV\_NONE** 0  
*Invalid version.*
- #define **EV\_CURRENT** 1  
*Current version.*
- #define **EI\_MAG0** 0  
*file id*
- #define **EI\_MAG1** 1  
*file id*
- #define **EI\_MAG2** 2  
*file id*
- #define **EI\_MAG3** 3  
*file id*
- #define **ELFMAG0** 0x7f  
*e\_ident[EI\_MAG0]*

- #define **ELFMAG1** 'E  
*e\_ident[EI\_MAG1]*
- #define **ELFMAG2** 'L  
*e\_ident[EI\_MAG2]*
- #define **ELFMAG3** 'F  
*e\_ident[EI\_MAG3]*
- #define **ELFCLASS32** 1  
*32-bit object*
- #define **ELFCLASS64** 2  
*64-bit object*
- #define **SHN\_UNDEF** 0  
*undefined section header entry*
- #define **SHN\_LORESERVE** 0xff00  
*lower bound of reserved indexes*
- #define **SHN\_LOPROC** 0xff00  
*lower bound of proc spec entr*
- #define **SHN\_HIPROC** 0xff1f  
*upper bound of proc spec entr*
- #define **SHN\_ABS** 0xffff1  
*absolute values for ref*
- #define **SHN\_COMMON** 0xffff2  
*common symbols*
- #define **SHN\_HIRESERVE** 0xfffff  
*upper bound of reserved indexes*
- #define **SHT\_INIT\_ARRAY** 14  
*Array of constructors.*
- #define **SHT\_FINI\_ARRAY** 15  
*Array of destructors.*
- #define **SHT\_PREINIT\_ARRAY** 16  
*Array of pre-constructors.*
- #define **SHT\_GROUP** 17  
*Section group.*
- #define **SHT\_SYMTAB\_SHNDX** 18  
*Extended section indeces.*

- #define **SHT\_NUM** 19  
*Number of defined types.*
- #define **SHF\_WRITE** 0x1  
*writeable during execution*
- #define **SHF\_ALLOC** 0x2  
*section occupies virt memory*
- #define **SHF\_EXECINSTR** 0x4  
*code section*
- #define **SHF\_MERGE** 0x10  
*Might be merged.*
- #define **SHF\_STRINGS** 0x20  
*Contains nul-terminated strings.*
- #define **SHF\_INFO\_LINK** 0x40  
*'sh\_info' contains SHT index*
- #define **SHF\_LINK\_ORDER** 0x80  
*Preserve order after combining.*
- #define **SHF\_OS\_NONCONFORMING** 0x100  
*Non-standard OS specific handling required.*
- #define **SHF\_GROUP** 0x200  
*Section is member of a group.*
- #define **SHF\_TLS** 0x400  
*Section hold thread-local data.*
- #define **SHF\_MASKOS** 0x0ff000000  
*OS-specific.*
- #define **SHF\_MASKPROC** 0xf0000000  
*proc spec mask*
- #define **PT\_NULL** 0  
*array is unused*
- #define **PT\_LOAD** 1  
*loadable*
- #define **PT\_DYNAMIC** 2  
*dynamic linking information*
- #define **PT\_INTERP** 3

*path to interpreter*

- #define **PT\_NOTE** 4  
*auxiliary information*
- #define **PT\_SHLIB** 5  
*reserved*
- #define **PT\_PHDR** 6  
*location of the pht itself*
- #define **PT\_TLS** 7  
*Thread-local storage segment.*
- #define **PT\_NUM** 8  
*Number of defined types.*
- #define **PT\_LOOS** 0x60000000  
*os spec.*
- #define **PT\_HIOS** 0x6fffffff  
*os spec.*
- #define **PT\_LOPROC** 0x70000000  
*processor spec.*
- #define **PT\_HIPROC** 0x7fffffff  
*processor spec.*
- #define **PT\_GNU\_EH\_FRAME** (PT\_LOOS + 0x474e550)  
*EH frame information.*
- #define **PT\_GNU\_STACK** (PT\_LOOS + 0x474e551)  
*Flags for stack.*
- #define **PT\_GNU\_RELRO** (PT\_LOOS + 0x474e552)  
*Read only after reloc.*
- #define **PT\_L4\_STACK** (PT\_LOOS + 0x12)  
*Address of the stack.*
- #define **PT\_L4\_KIP** (PT\_LOOS + 0x13)  
*Address of the KIP.*
- #define **PT\_L4\_AUX** (PT\_LOOS + 0x14)  
*Address of the AUX strcutures.*
- #define **NT\_PRSTATUS** 1  
*Contains copy of prstatus struct.*

- #define **NT\_FPREGSET** 2  
*Contains copy of fpregset struct.*
- #define **NT\_PRPSINFO** 3  
*Contains copy of prpsinfo struct.*
- #define **NT\_PRXREG** 4  
*Contains copy of prxregset struct.*
- #define **NT\_TASKSTRUCT** 4  
*Contains copy of task structure.*
- #define **NT\_PLATFORM** 5  
*String from sysinfo(SI\_PLATFORM).*
- #define **NT\_AUXV** 6  
*Contains copy of auxv array.*
- #define **NT\_GWINDOWS** 7  
*Contains copy of gwindows struct.*
- #define **NT\_ASRS** 8  
*Contains copy of asrset struct.*
- #define **NT\_PSTATUS** 10  
*Contains copy of pstatus struct.*
- #define **NT\_PSINFO** 13  
*Contains copy of psinfo struct.*
- #define **NT\_PRCRED** 14  
*Contains copy of prcred struct.*
- #define **NT\_UTSNAME** 15  
*Contains copy of utsname struct.*
- #define **NT\_LWPSTATUS** 16  
*Contains copy of lwpstatus struct.*
- #define **NT\_LWPSINFO** 17  
*Contains copy of lwpsinfo struct.*
- #define **NT\_PRFPXREG** 20  
*Contains copy of fprxregset struct.*
- #define **NT\_VERSION** 1  
*Contains a version string.*
- #define **DT\_NULL** 0  
*Dynamic Array Tags, d\_tag - figure 2-10, page 2-12.*

- #define **DT\_NEEDED** 1  
*name of a needed library*
- #define **DT\_PLTRELSZ** 2  
*total size of relocation entry*
- #define **DT\_PLTGOT** 3  
*address assoc with prog link table*
- #define **DT\_HASH** 4  
*address of symbol hash table*
- #define **DT\_STRTAB** 5  
*address of string table*
- #define **DT\_SYMTAB** 6  
*address of symbol table*
- #define **DT\_REL** 7  
*address of relocation table*
- #define **DT\_RELASZ** 8  
*total size of relocation table*
- #define **DT\_RELENT** 9  
*size of DT\_REL relocation entry*
- #define **DT\_STRSZ** 10  
*size of the string table*
- #define **DT\_SYMENT** 11  
*size of a symbol table entry*
- #define **DT\_INIT** 12  
*address of initialization function*
- #define **DT\_FINI** 13  
*address of termination function*
- #define **DT SONAME** 14  
*name of the shared object*
- #define **DT\_RPATH** 15  
*search library path*
- #define **DT\_SYMBOLIC** 16  
*alter symbol resolution algorithm*
- #define **DT\_REL** 17

*address of relocation table*

- #define **DT\_RELSZ** 18  
*total size of DT\_REL relocation table*
- #define **DT\_RELENT** 19  
*size of the DT\_REL relocation entry*
- #define **DT\_PTRREL** 20  
*type of relocation entry*
- #define **DT\_DEBUG** 21  
*for debugging purposes*
- #define **DT\_TEXTREL** 22  
*at least one entry changes r/o section*
- #define **DT\_JMPREL** 23  
*address of relocation entries*
- #define **DT\_BIND\_NOW** 24  
*Process relocations of object.*
- #define **DT\_INIT\_ARRAY** 25  
*Array with addresses of init fct.*
- #define **DT\_FINI\_ARRAY** 26  
*Array with addresses of fini fct.*
- #define **DT\_INIT\_ARRAYSZ** 27  
*Size in bytes of DT\_INIT\_ARRAY.*
- #define **DT\_FINI\_ARRAYSZ** 28  
*Size in bytes of DT\_FINI\_ARRAY.*
- #define **DT\_RUNPATH** 29  
*Library search path.*
- #define **DT\_FLAGS** 30  
*Flags for the object being loaded.*
- #define **DT\_ENCODING** 32  
*Start of encoded range.*
- #define **DT\_PREINIT\_ARRAY** 32  
*Array with addresses of preinit fct.*
- #define **DT\_PREINIT\_ARRAYSZ** 33  
*size in bytes of DT\_PREINIT\_ARRAY*

- #define **DT\_NUM** 34  
*Number used.*
- #define **DT\_LOOS** 0x6000000d  
*Start of OS-specific.*
- #define **DT\_HIOS** 0x6fffff000  
*End of OS-specific.*
- #define **DT\_LOPROC** 0x70000000  
*processor spec.*
- #define **DT\_HIPROC** 0x7fffffff  
*processor spec.*
- #define **DF\_ORIGIN** 0x00000001  
*Object may use DF\_ORIGIN.*
- #define **DF\_SYMBOLIC** 0x00000002  
*Symbol resolutions starts here.*
- #define **DF\_TEXTREL** 0x00000004  
*Object contains text relocations.*
- #define **DF\_BIND\_NOW** 0x00000008  
*No lazy binding for this object.*
- #define **DF\_STATIC\_TLS** 0x00000010  
*Module uses the static TLS model.*
- #define **DF\_1\_NOW** 0x00000001  
*Set RTLD\_NOW for this object.*
- #define **DF\_1\_GLOBAL** 0x00000002  
*Set RTLD\_GLOBAL for this object.*
- #define **DF\_1\_GROUP** 0x00000004  
*Set RTLD\_GROUP for this object.*
- #define **DF\_1\_NODELETE** 0x00000008  
*Set RTLD\_NODELETE for this object.*
- #define **DF\_1\_LOADFLTR** 0x00000010  
*Trigger filtee loading at runtime.*
- #define **DF\_1\_INITFIRST** 0x00000020  
*Set RTLD\_INITFIRST for this object.*
- #define **DF\_1\_NOOPEN** 0x00000040  
*Set RTLD\_NOOPEN for this object.*

- #define **DF\_1\_ORIGIN** 0x00000080  
*\$ORIGIN must be handled.*
- #define **DF\_1\_DIRECT** 0x00000100  
*Direct binding enabled.*
- #define **DF\_1\_INTERPOSE** 0x00000400  
*Object is used to interpose.*
- #define **DF\_1\_NODEFLIB** 0x00000800  
*Ignore default lib search path.*
- #define **DF\_1\_NODUMP** 0x00001000  
*Object can't be dldump'ed.*
- #define **DF\_1\_CONFALT** 0x00002000  
*Configuration alternative created.*
- #define **DF\_1\_ENDFILTEE** 0x00004000  
*Filtee terminates filters search.*
- #define **DF\_1\_DISPRLDNE** 0x00008000  
*Disp reloc applied at build time.*
- #define **DF\_1\_DISPRELPND** 0x00010000  
*Disp reloc applied at run-time.*
- #define **DF\_P1\_LAZYLOAD** 0x00000001  
*Lazyload following object.*
- #define **DF\_P1\_GROUPPERM** 0x00000002  
*Symbols from next object are not generally available.*
- #define **R\_386\_NONE** 0  
*none*
- #define **R\_386\_32** 1  
*S + A.*
- #define **R\_386\_PC32** 2  
*S + A - P.*
- #define **R\_386\_GOT32** 3  
*G + A - P.*
- #define **R\_386\_PLT32** 4  
*L + A - P.*
- #define **R\_386\_COPY** 5

*none*

- #define **R\_386\_GLOB\_DAT** 6  
*S.*
- #define **R\_386 JMP\_SLOT** 7  
*S.*
- #define **R\_386\_RELATIVE** 8  
*B + A.*
- #define **R\_386\_GOTOFF** 9  
*S + A - GOT.*
- #define **R\_386\_GOTPC** 10  
*GOT + A - P.*
- #define **STB\_LOCAL** 0  
*not visible outside object file*
- #define **STB\_GLOBAL** 1  
*visible to all objects being combined*
- #define **STB\_WEAK** 2  
*resemble global symbols*
- #define **STB\_LOOS** 10  
*os specific*
- #define **STB\_HIOS** 12  
*os specific*
- #define **STB\_LOPROC** 13  
*proc specific*
- #define **STB\_HIPROC** 15  
*proc specific*
- #define **STT\_NOTYPE** 0  
*symbol's type not specified*
- #define **STT\_OBJECT** 1  
*associated with a data object*
- #define **STT\_FUNC** 2  
*associated with a function or other code*
- #define **STT\_SECTION** 3  
*associated with a section*

- #define **STT\_FILE** 4  
*source file name associated with object*
- #define **STT\_LOOS** 10  
*os specific*
- #define **STT\_HIOS** 12  
*os specific*
- #define **STT\_LOPROC** 13  
*proc specific*
- #define **STT\_HIPROC** 15  
*proc specific*

## ELF types

- typedef **l4\_uint32\_t Elf32\_Addr**  
*size 4 align 4*
- typedef **l4\_uint32\_t Elf32\_Off**  
*size 4 align 4*
- typedef **l4\_uint16\_t Elf32\_Half**  
*size 2 align 2*
- typedef **l4\_uint32\_t Elf32\_Word**  
*size 4 align 4*
- typedef **l4\_int32\_t Elf32\_Sword**  
*size 4 align 4*
- typedef **l4\_uint64\_t Elf64\_Addr**  
*size 8 align 8*
- typedef **l4\_uint64\_t Elf64\_Off**  
*size 8 align 8*
- typedef **l4\_uint16\_t Elf64\_Half**  
*size 2 align 2*
- typedef **l4\_uint32\_t Elf64\_Word**  
*size 4 align 4*
- typedef **l4\_int32\_t Elf64\_Sword**  
*size 4 align 4*
- typedef **l4\_uint64\_t Elf64\_Xword**

*size 8 align 8*

- `typedef l4_int64_t Elf64_Sxword`  
*size 8 align 8*

### 9.82.1 Detailed Description

Functions and types related to ELF binaries.

### 9.82.2 Define Documentation

#### 9.82.2.1 #define EI\_CLASS 4

ELF class byte index.

file class

Definition at line [254](#) of file `elf.h`.

#### 9.82.2.2 #define EI\_CLASS 4

ELF class byte index.

file class

Definition at line [254](#) of file `elf.h`.

#### 9.82.2.3 #define ELFCLASSNONE 0

Invalid ELF class.

Invalid class.

Definition at line [270](#) of file `elf.h`.

#### 9.82.2.4 #define ELFCLASSNONE 0

Invalid ELF class.

Invalid class.

Definition at line [270](#) of file `elf.h`.

#### 9.82.2.5 #define EI\_DATA 5

Data encoding byte index.

data encoding

Definition at line [255](#) of file `elf.h`.

**9.82.2.6 #define EI\_DATA 5**

Data encoding byte index.

data encoding

Definition at line 255 of file [elf.h](#).

**9.82.2.7 #define ELFDATANONE 0**

Invalid data encoding.

invalid data encoding

Definition at line 276 of file [elf.h](#).

**9.82.2.8 #define ELFDATANONE 0**

Invalid data encoding.

invalid data encoding

Definition at line 276 of file [elf.h](#).

**9.82.2.9 #define ELFDATA2LSB 1**

2's complement, little endian

0x01020304 => [ 0x04|0x03|0x02|0x01 ]

Definition at line 277 of file [elf.h](#).

**9.82.2.10 #define ELFDATA2LSB 1**

2's complement, little endian

0x01020304 => [ 0x04|0x03|0x02|0x01 ]

Definition at line 277 of file [elf.h](#).

**9.82.2.11 #define ELFDATA2MSB 2**

2's complement, big endian

0x01020304 => [ 0x01|0x02|0x03|0x04 ]

Definition at line 278 of file [elf.h](#).

**9.82.2.12 #define ELFDATA2MSB 2**

2's complement, big endian

0x01020304 => [ 0x01|0x02|0x03|0x04 ]

Definition at line 278 of file [elf.h](#).

**9.82.2.13 #define EI\_VERSION 6**

File version byte index.

file version

Value must be EV\_CURRENT

Definition at line [256](#) of file [elf.h](#).

**9.82.2.14 #define EI\_VERSION 6**

File version byte index.

file version

Value must be EV\_CURRENT

Definition at line [256](#) of file [elf.h](#).

**9.82.2.15 #define EI\_OSABI 7**

OS ABI identification.

Operating system / ABI identification.

Definition at line [257](#) of file [elf.h](#).

**9.82.2.16 #define EI\_OSABI 7**

OS ABI identification.

Operating system / ABI identification.

Definition at line [257](#) of file [elf.h](#).

**9.82.2.17 #define ELFOSABI\_SYSV 0**

Alias.

UNIX System V ABI (this specification).

Definition at line [282](#) of file [elf.h](#).

**9.82.2.18 #define ELFOSABI\_SYSV 0**

Alias.

UNIX System V ABI (this specification).

Definition at line [282](#) of file [elf.h](#).

**9.82.2.19 #define ELFOSABI\_HPUX 1**

HP-UX.

HP-UX operating system.

Definition at line 283 of file [elf.h](#).

#### **9.82.2.20 #define ELFOSABI\_HPUX 1**

HP-UX.

HP-UX operating system.

Definition at line 283 of file [elf.h](#).

#### **9.82.2.21 #define ELFOSABI\_NETBSD 2**

NetBSD.

Definition at line 174 of file [elf.h](#).

#### **9.82.2.22 #define ELFOSABI\_LINUX 3**

Linux.

Definition at line 175 of file [elf.h](#).

#### **9.82.2.23 #define ELFOSABI\_SOLARIS 6**

Sun Solaris.

Definition at line 176 of file [elf.h](#).

#### **9.82.2.24 #define ELFOSABI\_AIX 7**

IBM AIX.

Definition at line 177 of file [elf.h](#).

#### **9.82.2.25 #define ELFOSABI\_IRIX 8**

SGI Irix.

Definition at line 178 of file [elf.h](#).

#### **9.82.2.26 #define ELFOSABI\_FREEBSD 9**

FreeBSD.

Definition at line 179 of file [elf.h](#).

#### **9.82.2.27 #define ELFOSABI\_TRU64 10**

Compaq TRU64 UNIX.

Definition at line 180 of file [elf.h](#).

**9.82.2.28 #define ELFOSABI\_MODESTO 11**

Novell Modesto.

Definition at line 181 of file [elf.h](#).

**9.82.2.29 #define ELFOSABI\_OPENBSD 12**

OpenBSD.

Definition at line 182 of file [elf.h](#).

**9.82.2.30 #define EI\_PAD 9**

Byte index of padding bytes.

start of padding bytes

Definition at line 259 of file [elf.h](#).

**9.82.2.31 #define EI\_PAD 9**

Byte index of padding bytes.

start of padding bytes

Definition at line 259 of file [elf.h](#).

**9.82.2.32 #define EM\_ARC 45**

Argonaut RISC Core, Argonaut Techn Inc.

Definition at line 226 of file [elf.h](#).

**9.82.2.33 #define SHT\_NUM 19**

Number of defined types.

Definition at line 348 of file [elf.h](#).

**9.82.2.34 #define SHF\_GROUP 0x200**

Section is member of a group.

Definition at line 367 of file [elf.h](#).

**9.82.2.35 #define SHF\_TLS 0x400**

Section hold thread-local data.

Definition at line 368 of file [elf.h](#).

**9.82.2.36 #define SHF\_MASKOS 0x0ff00000**

OS-specific.

Definition at line 369 of file [elf.h](#).

**9.82.2.37 #define PT\_LOOS 0x60000000**

os spec.

Definition at line 412 of file [elf.h](#).

**9.82.2.38 #define PT\_HIOS 0x6fffffff**

os spec.

Definition at line 413 of file [elf.h](#).

**9.82.2.39 #define PT\_LOPROC 0x70000000**

processor spec.

Definition at line 414 of file [elf.h](#).

**9.82.2.40 #define PT\_HIPROC 0x7fffffff**

processor spec.

Definition at line 415 of file [elf.h](#).

**9.82.2.41 #define PT\_GNU\_EH\_FRAME (PT\_LOOS + 0x474e550)**

EH frame information.

Definition at line 417 of file [elf.h](#).

**9.82.2.42 #define PT\_GNU\_STACK (PT\_LOOS + 0x474e551)**

Flags for stack.

Definition at line 418 of file [elf.h](#).

**9.82.2.43 #define PT\_GNU\_RELRO (PT\_LOOS + 0x474e552)**

Read only after reloc.

Definition at line 419 of file [elf.h](#).

**9.82.2.44 #define PT\_L4\_STACK (PT\_LOOS + 0x12)**

Address of the stack.

Definition at line 421 of file [elf.h](#).

**9.82.2.45 #define PT\_L4\_KIP (PT\_LOOS + 0x13)**

Address of the KIP.

Definition at line [422](#) of file [elf.h](#).

**9.82.2.46 #define PT\_L4\_AUX (PT\_LOOS + 0x14)**

Address of the AUX strcutures.

Definition at line [423](#) of file [elf.h](#).

**9.82.2.47 #define NT\_VERSION 1**

Contains a version string.

Definition at line [454](#) of file [elf.h](#).

**9.82.2.48 #define DT\_NULL 0**

Dynamic Array Tags, d\_tag - figure 2-10, page 2-12.

end of \_DYNAMIC array

Definition at line [478](#) of file [elf.h](#).

**9.82.2.49 #define DT\_LOPROC 0x70000000**

processor spec.

Definition at line [515](#) of file [elf.h](#).

**9.82.2.50 #define DT\_HIPROC 0x7fffffff**

processor spec.

Definition at line [516](#) of file [elf.h](#).

**9.82.2.51 #define DF\_1\_NOW 0x00000001**

Set RTLD\_NOW for this object.

Definition at line [527](#) of file [elf.h](#).

**9.82.2.52 #define DF\_1\_GLOBAL 0x00000002**

Set RTLD\_GLOBAL for this object.

Definition at line [528](#) of file [elf.h](#).

**9.82.2.53 #define DF\_1\_GROUP 0x00000004**

Set RTLD\_GROUP for this object.

Definition at line [529](#) of file `elf.h`.

**9.82.2.54 #define DF\_1\_NODELETE 0x00000008**

Set RTLD\_NODELETE for this object.

Definition at line [530](#) of file `elf.h`.

**9.82.2.55 #define DF\_1\_LOADFLTR 0x00000010**

Trigger filtee loading at runtime.

Definition at line [531](#) of file `elf.h`.

**9.82.2.56 #define DF\_1\_NOOPEN 0x00000040**

Set RTLD\_NOOPEN for this object.

Definition at line [533](#) of file `elf.h`.

**9.82.2.57 #define DF\_1\_ORIGIN 0x00000080**

\$ORIGIN must be handled.

Definition at line [534](#) of file `elf.h`.

**9.82.2.58 #define DF\_1\_DIRECT 0x00000100**

Direct binding enabled.

Definition at line [535](#) of file `elf.h`.

**9.82.2.59 #define DF\_1\_INTERPOSE 0x00000400**

Object is used to interpose.

Definition at line [537](#) of file `elf.h`.

**9.82.2.60 #define DF\_1\_NODEFLIB 0x00000800**

Ignore default lib search path.

Definition at line [538](#) of file `elf.h`.

**9.82.2.61 #define DF\_1\_NODUMP 0x00001000**

Object can't be dldump'ed.

Definition at line [539](#) of file `elf.h`.

**9.82.2.62 #define DF\_1\_CONFALT 0x00002000**

Configuration alternative created.

Definition at line [540](#) of file [elf.h](#).

**9.82.2.63 #define DF\_1\_ENDFILTEE 0x00004000**

Filtee terminates filters search.

Definition at line [541](#) of file [elf.h](#).

**9.82.2.64 #define DF\_1\_DISPRLDNE 0x00008000**

Disp reloc applied at build time.

Definition at line [542](#) of file [elf.h](#).

**9.82.2.65 #define DF\_1\_DISPRELPND 0x00010000**

Disp reloc applied at run-time.

Definition at line [543](#) of file [elf.h](#).

**9.82.2.66 #define DF\_P1\_LAZYLOAD 0x00000001**

Lazyload following object.

Definition at line [550](#) of file [elf.h](#).

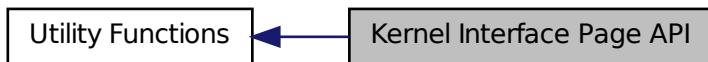
**9.82.2.67 #define DF\_P1\_GROUPPERM 0x00000002**

Symbols from next object are not generally available.

Definition at line [551](#) of file [elf.h](#).

## 9.83 Kernel Interface Page API

Collaboration diagram for Kernel Interface Page API:



## Files

- file [kip.h](#)

## Defines

- `#define l4util_kip_for_each_feature(s) for (s += strlen(s) + 1; *s; s += strlen(s) + 1)`  
*Cycle through kernel features given in the KIP.*

## Functions

- `int l4util_kip_kernel_is_ux (l4_kernel_info_t *)`  
*Return whether the kernel is running native or under UX.*
- `int l4util_kip_kernel_has_feature (l4_kernel_info_t *, const char *str)`  
*Check if kernel supports a feature.*
- `unsigned long l4util_kip_kernel_abi_version (l4_kernel_info_t *)`  
*Return kernel ABI version.*
- `l4_addr_t l4util_memdesc_vm_high (l4_kernel_info_t *kinfo)`  
*Return end of virtual memory.*

### 9.83.1 Define Documentation

#### 9.83.1.1 `#define l4util_kip_for_each_feature( s ) for (s += strlen(s) + 1; *s; s += strlen(s) + 1)`

Cycle through kernel features given in the KIP.

Cycles through all KIP kernel feature strings. s must be a character pointer (char \*) initialized with l4util\_kip\_version\_string().

Definition at line [74](#) of file [kip.h](#).

### 9.83.2 Function Documentation

#### 9.83.2.1 `int l4util_kip_kernel_is_ux ( l4_kernel_info_t * )`

Return whether the kernel is running native or under UX.

Returns whether the kernel is running natively or under UX. The KIP will be mapped if not already mapped. The KIP will not be unmapped again.

#### Returns

1 when running under UX, 0 if not running under UX

### 9.83.2.2 int l4util\_kip\_kernel\_has\_feature ( l4\_kernel\_info\_t \*, const char \* str )

Check if kernel supports a feature.

#### Parameters

*str* Feature name to check.

#### Returns

1 if the kernel supports the feature, 0 if not.

Checks the feature field in the KIP for the given string. The KIP will be mapped if not already mapped. The KIP will not be unmapped again.

### 9.83.2.3 unsigned long l4util\_kip\_kernel\_abi\_version ( l4\_kernel\_info\_t \* )

Return kernel ABI version.

#### Returns

Kernel ABI version.

### 9.83.2.4 l4\_addr\_t l4util\_memdesc\_vm\_high ( l4\_kernel\_info\_t \* kinfo )

Return end of virtual memory.

#### Returns

0 if memory descriptor could not be found, last address of address space otherwise

## 9.84 Comfortable Command Line Parsing

Collaboration diagram for Comfortable Command Line Parsing:



## Typedefs

- `typedef void(* parse_cmd_fn_t)(int)`  
*Function type for PARSE\_CMD\_FN.*

- `typedef void(* parse_cmd_fn_arg_t )(int, const char *, int)`  
*Function type for PARSE\_CMD\_FN\_ARG.*

## Enumerations

- `enum parse_cmd_type`  
*Types for parsing.*

## Functions

- `int parse_cmdline (int *argc, const char ***argv, char arg0,...)`  
*Parse the command-line for specified arguments and store the values into variables.*

### 9.84.1 Function Documentation

#### 9.84.1.1 `int parse_cmdline ( int * argc, const char *** argv, char arg0, ... )`

Parse the command-line for specified arguments and store the values into variables.

This Functions gets the command-line, and a list of command-descriptors. Then, the command-line is parsed according to the given descriptors, storing strings, switches and numeric arguments at given addresses, and possibly calling specified functions. A default help descriptor is added. Its purpose is to present a short command overview in the case the given command-line does not fit to the descriptors.

Each command-descriptor has the following form:

*short option char, long option name, comment, type, val, addr.*

The *short option char* specifies the short form of the described option. The short form will be recognized after a single dash, or in a group of short options preceeded by a single dash. Specify '' if no short form should be used.

The *long option name* specifies the long form of the described option. The long form will be recognized after two dashes. Specify 0 if no long form should be used for this option.

The *comment* is a string that will be used when presenting the short command-line help.

The *type* specifies, if the option should be recognized as

- a number (PARSE\_CMD\_INT),
- a switch (PARSE\_CMD\_SWITCH),
- a string (PARSE\_CMD\_STRING),
- a function call (PARSE\_CMD\_FN, PARSE\_CMD\_FN\_ARG),
- an increment/decrement operator (PARSE\_CMD\_INC, PARSE\_CMD\_DEC).

If type is PARSE\_CMD\_INT, the option requires a second argument on the command-line after the option. This argument is parsed as a number. It can be preceeded by 0x to present a hex-value or by 0 to present an octal form. *addr* is interpreted as an int-pointer. The scanned argument from the command-line is stored in this pointer.

If *type* is PARSE\_CMD\_SWITCH, *addr* must be a pointer to int, and the value from *val* is stored at this pointer.

With PARSE\_CMD\_STRING, an additional argument is expected at the cmdline. *addr* must be a pointer to const char\*, and a pointer to the argument on the command line is stored at this pointer. The value in *val* is a default value, which is stored at *addr* if the corresponding option is not given on the command line.

PARSE\_CMD\_FN\_ARG, *addr* is interpreted as a function pointer of type [parse\\_cmd\\_fn\\_t](#). It will be called with *val* as argument if the corresponding option is found.

If *type* is PARSE\_CMD\_FN\_ARG, *addr* is as a function pointer of type [parse\\_cmd\\_fn\\_arg\\_t](#), and handled similar to PARSE\_CMD\_FN. An additional argument is expected at the command line, however. It is given to the called function as 2nd argument, and parsed as an integer as with PARSE\_CMD\_INT as a third argument.

If *type* is PARSE\_CMD\_INC or PARSE\_CMD\_DEC, *addr* is interpreted as an int-pointer. The value of *val* is stored to this pointer first. For every occurrence of the option in the command line, the integer referenced by *addr* is incremented or decremented, respectively.

The list of command-descriptors is terminated by specifying a binary 0 for the short option char.

Note: The short option char 'h' and the long option name "help" must not be specified. They are used for the default help descriptor and produce a short command-options help when specified on the command-line.

## Parameters

*argc* pointer to number of command line parameters as passed to main

*argv* pointer to array of command line parameters as passed to main

*arg0* format list describing the command line options to parse for

## Returns

0 if the command-line was successfully parsed, otherwise:

- -1 if the given descriptors are somehow wrong.
- -2 if not enough memory was available to hold temporary structs.
- -3 if the given command-line args did not meet the specified set.
- -4 if the help-option was given.

Upon return, *argc* and *argv* point to a list of arguments that were not scanned as arguments. See  [getopt\\_long](#) for details on scanning.

## 9.85 Priority related functions

Collaboration diagram for Priority related functions:



## 9.86 Random number support

Collaboration diagram for Random number support:



### Functions

- `l4_uint32_t l4util_rand (void)`  
*Deliver next random number.*
- `void l4util_srand (l4_uint32_t seed)`  
*Initialize random number generator.*

#### 9.86.1 Function Documentation

##### 9.86.1.1 `l4_uint32_t l4util_rand ( void )`

Deliver next random number.

##### Returns

A new random number

##### 9.86.1.2 `void l4util_srand ( l4_uint32_t seed )`

Initialize random number generator.

##### Parameters

*seed* Value to initialize

## 9.87 Machine Restarting Function

Collaboration diagram for Machine Restarting Function:



### Functions

- void [l4util\\_reboot](#) (void))

*Machine reboot.*

## 9.88 Low-Level Thread Functions

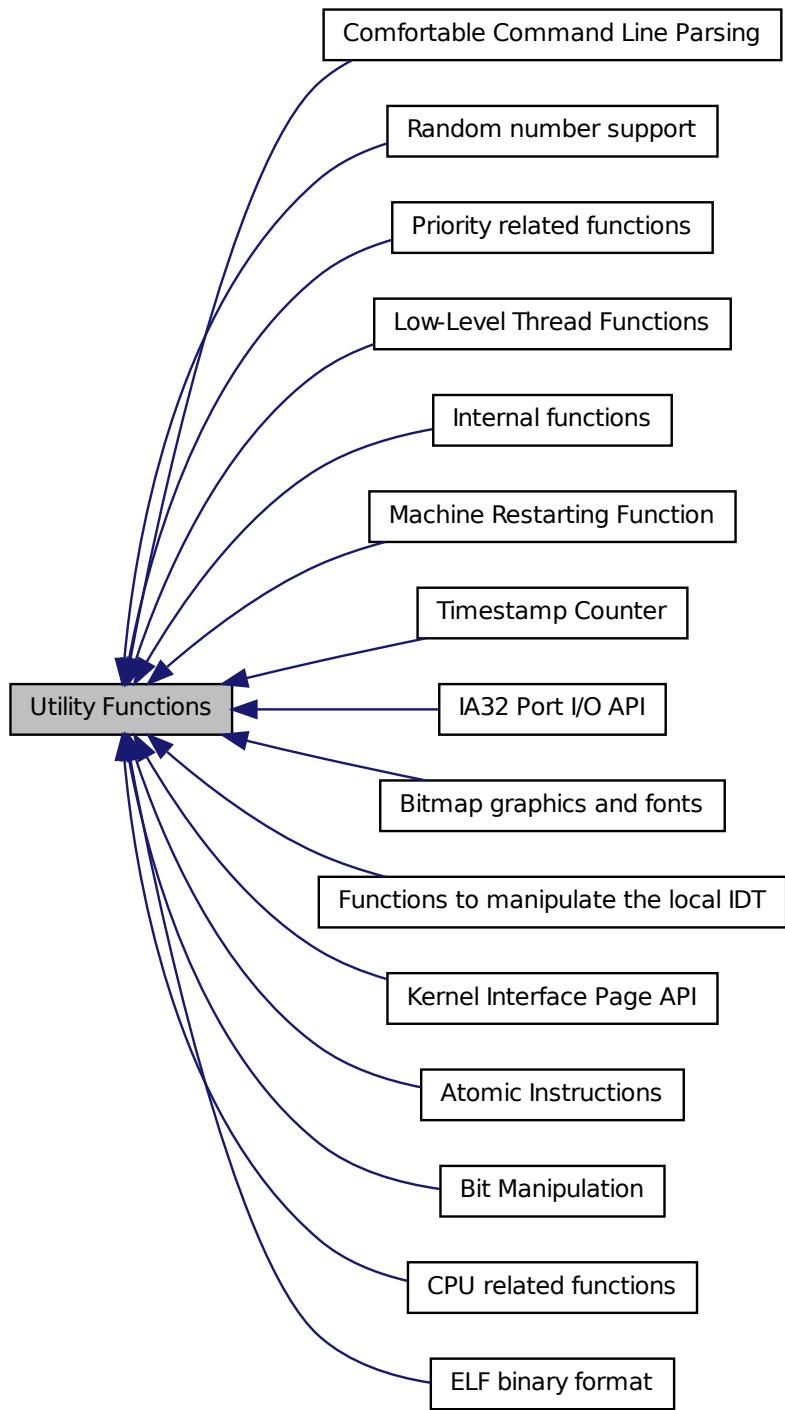
Collaboration diagram for Low-Level Thread Functions:





## 9.89 Utility Functions

Collaboration diagram for Utility Functions:



## Modules

- CPU related functions
- Functions to manipulate the local IDT
- Timestamp Counter
- Atomic Instructions
- Internal functions
- Bit Manipulation
- ELF binary format

*Functions and types related to ELF binaries.*

- Kernel Interface Page API
- Comfortable Command Line Parsing
- Priority related functions
- Random number support
- Machine Restarting Function
- Low-Level Thread Functions
- IA32 Port I/O API
- Bitmap graphics and fonts

*This library provides some functions for bitmap handling in frame buffers.*

## Files

- file `rand.h`

*Simple Pseudo-Random Number Generator.*

## Functions

- `void l4_sleep_forever (void) throw ()`

*Go sleep and never wake up.*

- `long l4util_splitlog2_hdl (l4_addr_t start, l4_addr_t end, long(*handler)(l4_addr_t s, l4_addr_t e, int log2size))`

*Split a range into log2 base and size aligned chunks.*

- `l4_addr_t l4util_splitlog2_size (l4_addr_t start, l4_addr_t end)`

*Return log2 base and size aligned length of a range.*

- `l4_timeout_s l4util_micros2l4to (unsigned int mus) throw ()`

*Calculate l4 timeouts.*

## 9.89.1 Function Documentation

### 9.89.1.1 void l4\_sleep\_forever( void ) throw() [inline]

Go sleep and never wake up.

< never timeout

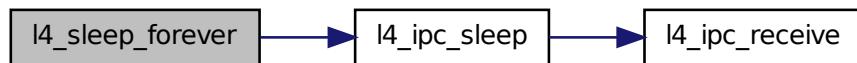
#### Examples:

[examples/libs/libirq/loop.c](#), and [examples/libs/shmc/prodcons.c](#).

Definition at line 47 of file [util.h](#).

References [L4\\_IPC\\_NEVER](#), and [l4\\_ipc\\_sleep\(\)](#).

Here is the call graph for this function:



### 9.89.1.2 long l4util\_splitlog2\_hdl( l4\_addr\_t start, l4\_addr\_t end, long(\*)(l4\_addr\_t s, l4\_addr\_t e, int log2size) handler ) [inline]

Split a range into log2 base and size aligned chunks.

#### Parameters

**start** Start of range

**end** End of range (inclusive) (e.g. 2-4 is len 3)

**handler** Handler function that is called with start and end (both inclusive) of the chunk. On success, the handler must return 0, if it returns !=0 the function will immediately return with the return code of the handler.

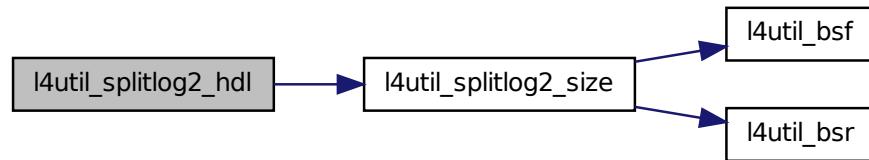
#### Returns

0 on success, != 0 otherwise

Definition at line 53 of file [splitlog2.h](#).

References [l4util\\_splitlog2\\_size\(\)](#).

Here is the call graph for this function:



### 9.89.1.3 `l4_addr_t l4util_splitlog2_size( l4_addr_t start, l4_addr_t end ) [inline]`

Return log2 base and size aligned length of a range.

#### Parameters

*start* Start of range

*end* End of range (inclusive) (e.g. 2-4 is len 3)

#### Returns

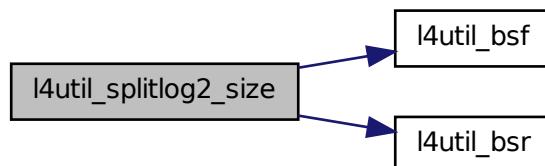
length of elements in log2size (length is  $1 << \text{log2size}$ )

Definition at line 72 of file [splitlog2.h](#).

References [l4util\\_bsfc\(\)](#), and [l4util\\_bsr\(\)](#).

Referenced by [l4util\\_splitlog2\\_hdl\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 9.89.1.4 `l4_timeout_s l4util_micros2l4to ( unsigned int mus ) throw ()`

Calculate l4 timeouts.

##### Parameters

`mus` time in microseconds. Special cases:

- 0 -> timeout 0
- ~0U -> timeout NEVER

##### Returns

the corresponding l4\_timeout value

## 9.90 IA32 Port I/O API

Collaboration diagram for IA32 Port I/O API:



## Functions

- `l4_uint8_t l4util_in8 (l4_uint16_t port)`  
*Read byte from I/O port.*
- `l4_uint16_t l4util_in16 (l4_uint16_t port)`  
*Read 16-bit-value from I/O port.*
- `l4_uint32_t l4util_in32 (l4_uint16_t port)`

*Read 32-bit-value from I/O port.*

- void `l4util_ins8 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`  
*Read a block of 8-bit-value from I/O ports.*
- void `l4util_ins16 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`  
*Read a block of 16-bit-value from I/O ports.*
- void `l4util_ins32 (l4_uint16_t port, l4_umword_t addr, l4_umword_t count)`  
*Read a block of 32-bit-value from I/O ports.*
- void `l4util_out8 (l4_uint8_t value, l4_uint16_t port)`  
*Write byte to I/O port.*
- void `l4util_out16 (l4_uint16_t value, l4_uint16_t port)`  
*Write 16-bit-value to I/O port.*
- void `l4util_out32 (l4_uint32_t value, l4_uint16_t port)`  
*Write 32-bit-value to I/O port.*
- void `l4util_iodelay (void)`  
*delay I/O port access by writing to port 0x80*

## 9.90.1 Function Documentation

### 9.90.1.1 `l4_uint8_t l4util_in8 ( l4_uint16_t port ) [inline]`

Read byte from I/O port.

#### Parameters

*port* I/O port address

#### Returns

*value*

Definition at line 145 of file `port_io.h`.

### 9.90.1.2 `l4_uint16_t l4util_in16 ( l4_uint16_t port ) [inline]`

Read 16-bit-value from I/O port.

#### Parameters

*port* I/O port address

#### Returns

*value*

Definition at line 153 of file `port_io.h`.

**9.90.1.3 l4\_uint32\_t l4util\_in32 ( l4\_uint16\_t port ) [inline]**

Read 32-bit-value from I/O port.

**Parameters**

*port* I/O port address

**Returns**

value

Definition at line 161 of file [port\\_io.h](#).

**9.90.1.4 void l4util\_ins8 ( l4\_uint16\_t port, l4\_umword\_t addr, l4\_umword\_t count ) [inline]**

Read a block of 8-bit-value from I/O ports.

**Parameters**

*port* I/O port address

*addr* address of buffer

*count* number of I/O operations

**Returns**

value

Definition at line 169 of file [port\\_io.h](#).

**9.90.1.5 void l4util\_ins16 ( l4\_uint16\_t port, l4\_umword\_t addr, l4\_umword\_t count ) [inline]**

Read a block of 16-bit-value from I/O ports.

**Parameters**

*port* I/O port address

*addr* address of buffer

*count* number of I/O operations

**Returns**

value

Definition at line 178 of file [port\\_io.h](#).

**9.90.1.6 void l4util\_ins32 ( l4\_uint16\_t port, l4\_umword\_t addr, l4\_umword\_t count ) [inline]**

Read a block of 32-bit-value from I/O ports.

**Parameters**

*port* I/O port address

*addr* address of buffer

*count* number of I/O operations

**Returns**

*value*

Definition at line 187 of file [port\\_io.h](#).

**9.90.1.7 void l4util\_out8( l4\_uint8\_t *value*, l4\_uint16\_t *port* ) [inline]**

Write byte to I/O port.

**Parameters**

*port* I/O port address

*value* value to write

Definition at line 196 of file [port\\_io.h](#).

**9.90.1.8 void l4util\_out16( l4\_uint16\_t *value*, l4\_uint16\_t *port* ) [inline]**

Write 16-bit-value to I/O port.

**Parameters**

*port* I/O port address

*value* value to write

Definition at line 202 of file [port\\_io.h](#).

**9.90.1.9 void l4util\_out32( l4\_uint32\_t *value*, l4\_uint16\_t *port* ) [inline]**

Write 32-bit-value to I/O port.

**Parameters**

*port* I/O port address

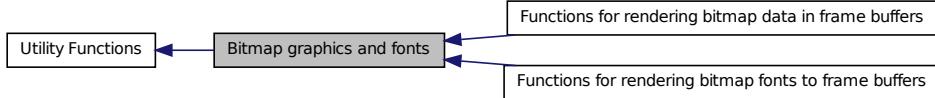
*value* value to write

Definition at line 208 of file [port\\_io.h](#).

## 9.91 Bitmap graphics and fonts

This library provides some functions for bitmap handling in frame buffers.

Collaboration diagram for Bitmap graphics and fonts:



## Modules

- [Functions for rendering bitmap data in frame buffers](#)
- [Functions for rendering bitmap fonts to frame buffers](#)

### 9.91.1 Detailed Description

This library provides some functions for bitmap handling in frame buffers. Includes simple functions like filling or copying an area of the frame buffer going up to rendering text into the frame buffer using bitmap fonts.

## 9.92 Functions for rendering bitmap data in frame buffers

Collaboration diagram for Functions for rendering bitmap data in frame buffers:



## Data Structures

- struct [gfxbitmap\\_offset](#)  
*offsets in pmap[] and bmap[]*

## Typedefs

- typedef unsigned int [gfxbitmap\\_color\\_t](#)  
*Standard color type.*
- typedef unsigned int [gfxbitmap\\_color\\_pix\\_t](#)  
*Specific color type.*

## Functions

- `gfxbitmap_color_pix_t gfxbitmap_convert_color (l4re_video_view_info_t *vi, gfxbitmap_color_t rgb)`  
*Convert a color.*
- `void gfxbitmap_fill (l4_uint8_t *vfb, l4re_video_view_info_t *vi, int x, int y, int w, int h, gfxbitmap_color_pix_t color)`  
*Fill a rectangular area with a color.*
- `void gfxbitmap_bmap (l4_uint8_t *vfb, l4re_video_view_info_t *vi, l4_int16_t x, l4_int16_t y, l4_int32_t w, l4_int32_t h, l4_uint8_t *bmap, gfxbitmap_color_pix_t fgc, gfxbitmap_color_pix_t bgc, struct gfxbitmap_offset *offset, l4_uint8_t mode)`  
*Fill a rectangular area with a bicolor bitmap pattern.*
- `void gfxbitmap_set (l4_uint8_t *vfb, l4re_video_view_info_t *vi, l4_int16_t x, l4_int16_t y, l4_int32_t w, l4_int32_t h, l4_int32_t xoffs, l4_int32_t yoffs, l4_uint8_t *pmap, struct gfxbitmap_offset *offset, l4_int32_t pwidht)`  
*Set area from source area.*
- `void gfxbitmap_copy (l4_uint8_t *dest, l4_uint8_t *src, l4re_video_view_info_t *vi, int x, int y, int w, int h, int dx, int dy)`  
*Copy a rectangular area.*

### 9.92.1 Typedef Documentation

#### 9.92.1.1 `typedef unsigned int gfxbitmap_color_t`

Standard color type.

It's a RGB type with 8bits for each channel, regardless of the framebuffer used.

Definition at line 55 of file `bitmap.h`.

#### 9.92.1.2 `typedef unsigned int gfxbitmap_color_pix_t`

Specific color type.

This color type is specific for a particular framebuffer, it can be use to write pixel on a framebuffer. Use `gfxbitmap_convert_color` to convert from `gfxbitmap_color_t` to `gfxbitmap_color_pix_t`.

Definition at line 64 of file `bitmap.h`.

### 9.92.2 Function Documentation

#### 9.92.2.1 `gfxbitmap_color_pix_t gfxbitmap_convert_color ( l4re_video_view_info_t * vi, gfxbitmap_color_t rgb )`

Convert a color.

Converts a given color in standard format to the format used in the framebuffer.

---

**9.92.2.2 void gfxbitmap\_fill ( l4\_uint8\_t \* *vfb*, l4re\_video\_view\_info\_t \* *vi*, int *x*, int *y*, int *w*, int *h*, gfxbitmap\_color\_pix\_t *color* )**

Fill a rectangular area with a color.

#### Parameters

*vfb* Frame buffer.  
*fbi* Frame buffer information structure.  
*x* X position of area.  
*y* Y position of area.  
*w* Width of area.  
*h* Height of area.  
*color* Color of area.

**9.92.2.3 void gfxbitmap\_bmap ( l4\_uint8\_t \* *vfb*, l4re\_video\_view\_info\_t \* *vi*, l4\_int16\_t *x*, l4\_int16\_t *y*, l4\_uint32\_t *w*, l4\_uint32\_t *h*, l4\_uint8\_t \* *bmap*, gfxbitmap\_color\_pix\_t *fgc*, gfxbitmap\_color\_pix\_t *bgc*, struct gfxbitmap\_offset \* *offset*, l4\_uint8\_t *mode* )**

Fill a rectangular area with a bicolor bitmap pattern.

#### Parameters

*vfb* Frame buffer.  
*fbi* Frame buffer information structure.  
*x* X position of area.  
*y* Y position of area.  
*w* Width of area.  
*h* Height of area.  
*bmap* Bitmap pattern.  
*fgc* Foreground color.  
*bgc* Background color.  
*offset* Offsets.  
*mode* Mode (

#### See also

pSLIM\_BMAP\_START\_MSB and \* #pSLIM\_BMAP\_START\_LSB).

**9.92.2.4 void gfxbitmap\_set ( l4\_uint8\_t \* *vfb*, l4re\_video\_view\_info\_t \* *vi*, l4\_int16\_t *x*, l4\_int16\_t *y*, l4\_uint32\_t *w*, l4\_uint32\_t *h*, l4\_uint32\_t *xoffs*, l4\_uint32\_t *yoffs*, l4\_uint8\_t \* *pmap*, struct gfxbitmap\_offset \* *offset*, l4\_uint32\_t *ewidth* )**

Set area from source area.

#### Parameters

*vfb* Frame buffer.

*fbi* Frame buffer information structure.

*x* X position of area.

*y* Y position of area.

*w* Width of area.

*h* Height of area.

*pmap* Source.

*xoffs* X offset.

*yoffs* Y offset.

*offset* Offsets.

*pwidht* Width of source in bytes.

### 9.92.2.5 void gfxbitmap\_copy( l4\_uint8\_t \* dest, l4\_uint8\_t \* src, l4re\_video\_view\_info\_t \* vi, int x, int y, int w, int h, int dx, int dy )

Copy a rectangular area.

#### Parameters

*dest* Destination frame buffer.

*src* Source frame buffer.

*fbi* Frame buffer information structure.

*x* Source X position of area.

*y* Source Y position of area.

*w* Width of area.

*h* Height of area.

*dx* Source X position of area.

*dy* Source Y position of area.

## 9.93 Functions for rendering bitmap fonts to frame buffers

Collaboration diagram for Functions for rendering bitmap fonts to frame buffers:



## Defines

- #define [GFXBITMAP\\_DEFAULT\\_FONT](#) (void \*)0

*Constant to use for the default font.*

## Typedefs

- `typedef void * gfxbitmap_font_t`  
*Font.*

## Enumerations

- `enum`  
*Constant for length field.*

## Functions

- `int gfxbitmap_font_init (void)`  
*Initialize the library.*
- `gfxbitmap_font_t gfxbitmap_font_get (const char *name)`  
*Get a font descriptor.*
- `unsigned gfxbitmap_font_width (gfxbitmap_font_t font)`  
*Get the font width.*
- `unsigned gfxbitmap_font_height (gfxbitmap_font_t font)`  
*Get the font height.*
- `void * gfxbitmap_font_data (gfxbitmap_font_t font, unsigned c)`  
*Get bitmap font data for a specific character.*
- `void gfxbitmap_font_text (void *fb, l4re_video_view_info_t *vi, gfxbitmap_font_t font, const char *text, unsigned len, unsigned x, unsigned y, gfxbitmap_color_pix_t fg, gfxbitmap_color_pix_t bg)`  
*Render a string to a framebuffer.*
- `void gfxbitmap_font_text_scale (void *fb, l4re_video_view_info_t *vi, gfxbitmap_font_t font, const char *text, unsigned len, unsigned x, unsigned y, gfxbitmap_color_pix_t fg, gfxbitmap_color_pix_t bg, int scale_x, int scale_y)`  
*Render a string to a framebuffer, including scaling.*

### 9.93.1 Enumeration Type Documentation

#### 9.93.1.1 anonymous enum

Constant for length field.

Use this if the function should call `strlen` on the text argument itself.

Definition at line 38 of file `font.h`.

## 9.93.2 Function Documentation

### 9.93.2.1 int gfxbitmap\_font\_init ( void )

Initialize the library.

This function must be called before any other font function of this library.

#### Returns

0 on success, other on error

### 9.93.2.2 gfxbitmap\_font\_t gfxbitmap\_font\_get ( const char \* *name* )

Get a font descriptor.

#### Parameters

*name* Name of the font.

#### Returns

A (opaque) font descriptor, or NULL if font could not be found.

### 9.93.2.3 unsigned gfxbitmap\_font\_width ( gfxbitmap\_font\_t *font* )

Get the font width.

#### Parameters

*font* Font.

#### Returns

Font width, 0 if font width could not be retrieved.

### 9.93.2.4 unsigned gfxbitmap\_font\_height ( gfxbitmap\_font\_t *font* )

Get the font height.

#### Parameters

*font* Font.

#### Returns

Font height, 0 if font height could not be retrieved.

---

### 9.93.2.5 void\* gfxbitmap\_font\_data ( *gfxbitmap\_font\_t font*, *unsigned c* )

Get bitmap font data for a specific character.

#### Parameters

*font* Font.

*c* Character.

#### Returns

Pointer to bmap data, NULL on error.

### 9.93.2.6 void gfxbitmap\_font\_text ( *void \* fb*, *l4re\_video\_view\_info\_t \* vi*, *gfxbitmap\_font\_t font*, *const char \* text*, *unsigned len*, *unsigned x*, *unsigned y*, *gfxbitmap\_color\_pix\_t fg*, *gfxbitmap\_color\_pix\_t bg* )

Render a string to a framebuffer.

#### Parameters

*fb* Pointer to frame buffer.

*fbi* Frame buffer info structure.

*font* Font.

*text* Text string.

*len* Length of the text string.

*x* Horizontal position in the frame buffer.

*y* Vertical position in the frame buffer.

*fg* Foreground color.

*bg* Background color.

### 9.93.2.7 void gfxbitmap\_font\_text\_scale ( *void \* fb*, *l4re\_video\_view\_info\_t \* vi*, *gfxbitmap\_font\_t font*, *const char \* text*, *unsigned len*, *unsigned x*, *unsigned y*, *gfxbitmap\_color\_pix\_t fg*, *gfxbitmap\_color\_pix\_t bg*, *int scale\_x*, *int scale\_y* )

Render a string to a framebuffer, including scaling.

#### Parameters

*fb* Pointer to frame buffer.

*fbi* Frame buffer info structure.

*font* Font.

*text* Text string.

*len* Length of the text string.

*x* Horizontal position in the frame buffer.

*y* Vertical position in the frame buffer.

*fg* Foreground color.

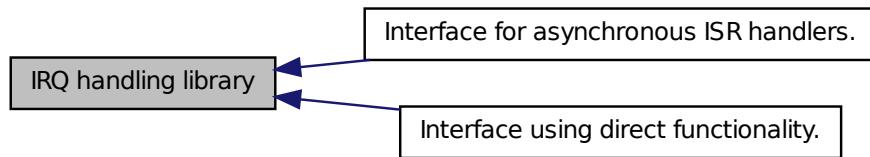
*bg* Background color.

*scale\_x* Horizontal scale factor.

*scale\_y* Vertical scale factor.

## 9.94 IRQ handling library

Collaboration diagram for IRQ handling library:



### Modules

- Interface using direct functionality.
- Interface for asynchronous ISR handlers.

*This interface has just two (main) functions.*

## 9.95 Interface using direct functionality.

Collaboration diagram for Interface using direct functionality.:



### Modules

- Interface using direct functionality.

### Functions

- `l4irq_t * l4irq_attach (int irqnum)`  
*Attach/connect to IRQ.*
- `l4irq_t * l4irq_attach_ft (int irqnum, unsigned flow_type)`  
*Attach/connect to IRQ using given type.*
- `l4irq_t * l4irq_attach_thread (int irqnum, l4_cap_idx_t to_thread)`

*Attach/connect to IRQ.*

- `l4irq_t * l4irq_attach_thread_ft (int irqnum, l4_cap_idx_t to_thread, unsigned flow_type)`  
*Attach/connect to IRQ using given type.*
- `long l4irq_wait (l4irq_t *irq)`  
*Wait for specified IRQ.*
- `long l4irq_unmask_and_wait_any (l4irq_t *unmask_irq, l4irq_t **ret_irq)`  
*Unmask a specific IRQ and wait for any attached IRQ.*
- `long l4irq_wait_any (l4irq_t **irq)`  
*Wait for any attached IRQ.*
- `long l4irq_unmask (l4irq_t *irq)`  
*Unmask a specific IRQ.*
- `long l4irq_detach (l4irq_t *irq)`  
*Detach from IRQ.*

## 9.95.1 Function Documentation

### 9.95.1.1 `l4irq_t* l4irq_attach ( int irqnum )`

Attach/connect to IRQ.

#### Parameters

*irqnum* IRQ number to request

#### Returns

Pointer to l4irq\_t structure, 0 on error

This l4irq\_attach has to be called in the same thread as l4irq\_wait and caller has to be a pthread thread.

#### Examples:

[examples/libs/libirq/loop.c](#).

### 9.95.1.2 `l4irq_t* l4irq_attach_ft ( int irqnum, unsigned flow_type )`

Attach/connect to IRQ using given type.

#### Parameters

*irqnum* IRQ number to request

*flow\_type* Interrupt type,

#### See also

[L4\\_irq\\_flow\\_type](#)

**Returns**

Pointer to l4irq\_t structure, 0 on error

This l4irq\_attach has to be called in the same thread as l4irq\_wait and caller has to be a pthread thread.

**9.95.1.3 l4irq\_t\* l4irq\_attach\_thread ( int irqnum, l4\_cap\_idx\_t to\_thread )**

Attach/connect to IRQ.

**Parameters**

*irqnum* IRQ number to request

*to\_thread* Attach IRQ to this specified thread.

**Returns**

Pointer to l4irq\_t structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

**9.95.1.4 l4irq\_t\* l4irq\_attach\_thread\_ft ( int irqnum, l4\_cap\_idx\_t to\_thread, unsigned flow\_type )**

Attach/connect to IRQ using given type.

**Parameters**

*irqnum* IRQ number to request

*to\_thread* Attach IRQ to this specified thread.

*flow\_type* Interrupt type,

**See also**

[L4\\_irq\\_flow\\_type](#)

**Returns**

Pointer to l4irq\_t structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

**9.95.1.5 long l4irq\_wait ( l4irq\_t \* irq )**

Wait for specified IRQ.

**Parameters**

*irq* IRQ data structure

**Returns**

0 on success, != 0 on error

**Examples:**

[examples/libs/libirq/loop.c](#).

**9.95.1.6 long l4irq\_unmask\_and\_wait\_any ( l4irq\_t \* *unmask\_irq*, l4irq\_t \*\* *ret\_irq* )**

Unmask a specific IRQ and wait for any attached IRQ.

**Parameters**

*unmask\_irq* IRQ data structure for unmask.

**Return values**

*ret\_irq* Received interrupt.

**Returns**

0 on success, != 0 on error

**9.95.1.7 long l4irq\_wait\_any ( l4irq\_t \*\* *irq* )**

Wait for any attached IRQ.

**Return values**

*irq* Received interrupt.

**Returns**

0 on success, != 0 on error

**9.95.1.8 long l4irq\_unmask ( l4irq\_t \* *irq* )**

Unmask a specific IRQ.

**Parameters**

*irq* IRQ data structure

**Returns**

0 on success, != 0 on error

This function is useful if a thread wants to wait for multiple IRQs using l4\_ipc\_wait.

**9.95.1.9 long l4irq\_detach ( l4irq\_t \* *irq* )**

Detach from IRQ.

**Parameters**

*irq* IRQ data structure

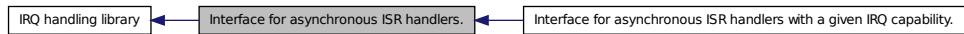
**Returns**

0 on success, != 0 on error

## 9.96 Interface for asynchronous ISR handlers.

This interface has just two (main) functions.

Collaboration diagram for Interface for asynchronous ISR handlers.:



### Modules

- **Interface for asynchronous ISR handlers with a given IRQ capability.**

*This group is just an enhanced version to [l4irq\\_request\(\)](#) which takes a capability object instead of a plain number.*

### Functions

- `l4irq_t * l4irq_request (int irqnum, void(*isr_handler)(void *), void *isr_data, int irq_thread_prio, unsigned flow_type)`

*Attach asynchronous ISR handler to IRQ.*
- `long l4irq_release (l4irq_t *irq)`

*Release asynchronous ISR handler and free resources.*

#### 9.96.1 Detailed Description

This interface has just two (main) functions. `l4irq_request` to install a handler for an interrupt and `l4irq_release` to uninstall the handler again and release all resources associated with it.

#### 9.96.2 Function Documentation

##### 9.96.2.1 `l4irq_t* l4irq_request ( int irqnum, void(*)(void *) isr_handler, void * isr_data, int irq_thread_prio, unsigned flow_type )`

Attach asynchronous ISR handler to IRQ.

#### Parameters

`irqnum` IRQ number to request

`isr_handler` Handler routine that is called when an interrupt triggers

`isr_data` Pointer given as argument to `isr_handler`

`irq_thread_prio` L4 thread priority of the ISR handler. Give -1 for same priority as creator.

`flow_type` Interrupt type,

**See also**

[L4\\_irq\\_flow\\_type](#)

**Returns**

Pointer to l4irq\_t structure, 0 on error

**Examples:**

[examples/libs/libirq/async\\_isr.c](#).

**9.96.2.2 long l4irq\_release ( l4irq\_t \* *irq* )**

Release asynchronous ISR handler and free resources.

**Parameters**

*irq* IRQ data structure

**Returns**

0 sucess, != 0 failure

**Examples:**

[examples/libs/libirq/async\\_isr.c](#).

**9.97 Interface using direct functionality.**

Collaboration diagram for Interface using direct functionality.:  


**Functions**

- `l4irq_t * l4irq_attach_cap (l4_cap_idx_t irqcap)`  
*Attach/connect to IRQ.*
- `l4irq_t * l4irq_attach_cap_ft (l4_cap_idx_t irqcap, unsigned flow_type)`  
*Attach/connect to IRQ using given type.*
- `l4irq_t * l4irq_attach_thread_cap (l4_cap_idx_t irqcap, l4_cap_idx_t to_thread)`  
*Attach/connect to IRQ.*

- `l4irq_t * l4irq_attach_thread_cap_ft (l4_cap_idx_t irqcap, l4_cap_idx_t to_thread, unsigned flow_type)`

*Attach/connect to IRQ using given type.*

## 9.97.1 Function Documentation

### 9.97.1.1 `l4irq_t* l4irq_attach_cap ( l4_cap_idx_t irqcap )`

Attach/connect to IRQ.

#### Parameters

*irqcap* IRQ capability

#### Returns

Pointer to l4irq\_t structure, 0 on error

This l4irq\_attach has to be called in the same thread as l4irq\_wait and caller has to be a pthread thread.

### 9.97.1.2 `l4irq_t* l4irq_attach_cap_ft ( l4_cap_idx_t irqcap, unsigned flow_type )`

Attach/connect to IRQ using given type.

#### Parameters

*irqcap* IRQ capability

*flow\_type* Interrupt type,

#### See also

[L4\\_irq\\_flow\\_type](#)

#### Returns

Pointer to l4irq\_t structure, 0 on error

This l4irq\_attach has to be called in the same thread as l4irq\_wait and caller has to be a pthread thread.

### 9.97.1.3 `l4irq_t* l4irq_attach_thread_cap ( l4_cap_idx_t irqcap, l4_cap_idx_t to_thread )`

Attach/connect to IRQ.

#### Parameters

*irqcap* IRQ capability

*to\_thread* Attach IRQ to this thread.

#### Returns

Pointer to l4irq\_t structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

### 9.97.1.4 l4irq\_t\* l4irq\_attach\_thread\_cap\_ft ( l4\_cap\_idx\_t irqcap, l4\_cap\_idx\_t to\_thread, unsigned flow\_type )

Attach/connect to IRQ using given type.

#### Parameters

*irqcap* IRQ capability

*to\_thread* Attach IRQ to this thread.

*flow\_type* Interrupt type,

#### See also

[L4\\_irq\\_flow\\_type](#)

#### Returns

Pointer to l4irq\_t structure, 0 on error

The pointer to the IRQ structure is used as a label in the IRQ object.

## 9.98 Interface for asynchronous ISR handlers with a given IRQ capability.

This group is just an enhanced version to [l4irq\\_request\(\)](#) which takes a capability object instead of a plain number.

Collaboration diagram for Interface for asynchronous ISR handlers with a given IRQ capability.:  

```

graph LR
    A[Interface for asynchronous ISR handlers.] --> B[Interface for asynchronous ISR handlers with a given IRQ capability.]

```

#### Functions

- l4irq\_t \* [l4irq\\_request\\_cap](#) (l4\_cap\_idx\_t irqcap, void(\*isr\_handler)(void \*), void \*isr\_data, int irq\_thread\_prio, unsigned flow\_type)

*Attach asynchronous ISR handler to IRQ.*

### 9.98.1 Detailed Description

This group is just an enhanced version to [l4irq\\_request\(\)](#) which takes a capability object instead of a plain number.

## 9.98.2 Function Documentation

**9.98.2.1 l4irq\_t\* l4irq\_request\_cap ( l4\_cap\_idx\_t irqcap, void(\*)(void \*) isr\_handler, void \* isr\_data, int irq\_thread\_prio, unsigned flow\_type )**

Attach asynchronous ISR handler to IRQ.

### Parameters

*irqcap* IRQ capability

*isr\_handler* Handler routine that is called when an interrupt triggers

*isr\_data* Pointer given as argument to *isr\_handler*

*irq\_thread\_prio* L4 thread priority of the ISR handler. Give -1 for same priority as creator.

*flow\_type* Interrupt type,

### See also

[L4\\_irq\\_flow\\_type](#)

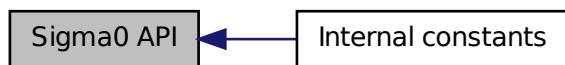
### Returns

Pointer to l4irq\_t structure, 0 on error

## 9.99 Sigma0 API

Sigma0 API bindings.

Collaboration diagram for Sigma0 API:



## Modules

- [Internal constants](#)

*Internal sigma0 definitions.*

## Files

- file [sigma0.h](#)

*Sigma0 interface.*

## Enumerations

- enum l4sigma0\_return\_flags\_t {
 L4SIGMA0\_OK, L4SIGMA0\_NOTALIGNED, L4SIGMA0\_IPCERROR, L4SIGMA0\_NOFPAGE
 ,
 L4SIGMA0\_SMALLERFPAGE
 }
- Return flags of libsigma0 functions.*

## Functions

- l4\_kernel\_info\_t \* l4sigma0\_map\_kip (l4\_cap\_idx\_t sigma0, void \*addr, unsigned log2\_size)  
*Map the kernel info page from pager to addr.*
- int l4sigma0\_map\_mem (l4\_cap\_idx\_t sigma0, l4\_addr\_t phys, l4\_addr\_t virt, l4\_addr\_t size)  
*Request a memory mapping from sigma0.*
- int l4sigma0\_map\_iomem (l4\_cap\_idx\_t sigma0, l4\_addr\_t phys, l4\_addr\_t virt, l4\_addr\_t size, int cached)  
*Request IO memory from sigma0.*
- int l4sigma0\_map\_anypage (l4\_cap\_idx\_t sigma0, l4\_addr\_t map\_area, unsigned log2\_map\_size, l4\_addr\_t \*base, unsigned sz)  
*Request an arbitrary free page of RAM.*
- int l4sigma0\_map\_tbuf (l4\_cap\_idx\_t sigma0, l4\_addr\_t virt)  
*Request Fiasco trace buffer.*
- void l4sigma0\_debug\_dump (l4\_cap\_idx\_t sigma0)  
*Request sigma0 to dump internal debug information.*
- int l4sigma0\_new\_client (l4\_cap\_idx\_t sigma0, l4\_cap\_idx\_t gate)  
*Create a new IPC gate for a new Sigma0 client.*
- char const \* l4sigma0\_map\_errstr (int err)  
*Get a user readable error messages for the return codes.*

### 9.99.1 Detailed Description

Sigma0 API bindings. Convenience bindings for the Sigma0 protocol.

### 9.99.2 Enumeration Type Documentation

#### 9.99.2.1 enum l4sigma0\_return\_flags\_t

Return flags of libsigma0 functions.

**Enumerator:***L4SIGMA0\_OK* Ok.*L4SIGMA0\_NOTALIGNED* Phys, virt or size not aligned.*L4SIGMA0\_IPCERROR* IPC error.*L4SIGMA0\_NOFPAGE* No fpage received.*L4SIGMA0\_SMALLERFPAGE* Superpage requested but smaller flexpage received.

Definition at line 81 of file [sigma0.h](#).

### 9.99.3 Function Documentation

#### 9.99.3.1 `l4_kernel_info_t* l4sigma0_map_kip( l4_cap_idx_t sigma0, void * addr, unsigned log2_size )`

Map the kernel info page from pager to addr.

**Parameters***sigma0* Capability selector for the sigma0 gate.*addr* Start of the receive window to receive KIP in.*log2\_size* Size of the receive window to receive KIP in.**Returns**

Address KIP was mapped to, 0 indicates an error.

#### 9.99.3.2 `int l4sigma0_map_mem( l4_cap_idx_t sigma0, l4_addr_t phys, l4_addr_t virt, l4_addr_t size )`

Request a memory mapping from sigma0.

**Parameters***sigma0* ID of service talking the sigma0 protocol.*phys* the physical address of the requested page (must be at least aligned to the minimum page size).*virt* the virtual address where the paged should be mapped in the local address space (must be at least aligned to the minimum page size).*size* the size of the requested page, this must be a multiple of the minimum page size.**Returns**

0 on success, !0 else (see [l4sigma0\\_map\\_errstr\(\)](#)).

#### 9.99.3.3 `int l4sigma0_map_iomem( l4_cap_idx_t sigma0, l4_addr_t phys, l4_addr_t virt, l4_addr_t size, int cached )`

Request IO memory from sigma0.

This function is similar to [l4sigma0\\_map\\_mem\(\)](#), the difference is that it requests IO memory. IO memory is everything that is not known to be normal RAM. Also ACPI tables or the BIOS memory is treated as IO memory.

**Parameters**

*sigma0* usually the thread id of sigma0.  
*phys* the physical address to be requested (page aligned).  
*virt* the virtual address where the memory should be mapped to (page aligned).  
*size* the size of the IO memory area to be mapped (multiple of page size)  
*cached* requests cacheable IO memory if 1, and uncached if 0.

**Returns**

0 on success, !0 else (see [l4sigma0\\_map\\_errstr\(\)](#)).

### 9.99.3.4 int l4sigma0\_map\_anypage ( l4\_cap\_idx\_t sigma0, l4\_addr\_t map\_area, unsigned log2\_map\_size, l4\_addr\_t \* base, unsigned sz )

Request an arbitrary free page of RAM.

This function requests arbitrary free memory from sigma0. It should be used whenever spare memory is needed, instead of requesting specific physical memory with [l4sigma0\\_map\\_mem\(\)](#).

**Parameters**

*sigma0* usually the thread id of sigma0.  
*map\_area* the base address of the local virtual memory area where the page should be mapped.  
*log2\_map\_size* the size of the requested page log 2 (the size in bytes is  $2^{\log2\_map\_size}$ ). This must be at least the minimal page size. By specifying larger sizes the largest possible hardware page size will be used.

**Return values**

*base* physical address of the page received (i.e., the send base of the received mapping if any).

**Parameters**

*sz* Size to map by the server, in  $2^{\log2\_map\_size}$  bytes.

**Returns**

0 on success, !0 else (see [l4sigma0\\_map\\_errstr\(\)](#)).

### 9.99.3.5 int l4sigma0\_map\_tbuf ( l4\_cap\_idx\_t sigma0, l4\_addr\_t virt )

Request Fiasco trace buffer.

This is a Fiasco specific feature. Where you can request the kernel internal trace buffer for user-level evaluation. This is for special debugging tools, such as Ferret.

**Parameters**

*sigma0* as usual the sigma0 thread id.  
*virt* the virtual address where the trace buffer should be mapped,

**Returns**

0 on success, !0 else (see [l4sigma0\\_map\\_errstr\(\)](#)).

**9.99.3.6 void l4sigma0\_debug\_dump ( l4\_cap\_idx\_t sigma0 )**

Request sigma0 to dump internal debug information.

The debug information, such as internal memory maps, as well as statistics about the internal allocators is dumped to the kernel debugger.

**Parameters**

*sigma0* the sigma0 thread id.

**9.99.3.7 int l4sigma0\_new\_client ( l4\_cap\_idx\_t sigma0, l4\_cap\_idx\_t gate )**

Create a new IPC gate for a new Sigma0 client.

**Parameters**

*sigma0* Capability selector for sigma0 gate.

*gate* Capability selector to use for the new gate.

**9.99.3.8 char const \* l4sigma0\_map\_errstr ( int err ) [inline]**

Get a user readable error messages for the return codes.

**Parameters**

*err* the error code reported by the \*map\* functions.

**Returns**

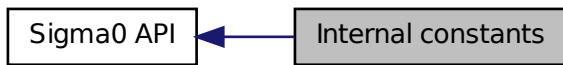
a string containing the error message.

Definition at line 208 of file [sigma0.h](#).

## 9.100 Internal constants

Internal sigma0 definitions.

Collaboration diagram for Internal constants:



## Defines

- #define **SIGMA0\_REQ\_MAGIC** ~0xFFUL  
*Request magic.*
- #define **SIGMA0\_REQ\_MASK** ~0xFFUL  
*Request mask.*
- #define **SIGMA0\_REQ\_ID\_MASK** 0xF0  
*ID mask.*
- #define **SIGMA0\_REQ\_ID\_FPAGE\_RAM** 0x60  
*RAM.*
- #define **SIGMA0\_REQ\_ID\_FPAGE\_IOMEM** 0x70  
*I/O memory.*
- #define **SIGMA0\_REQ\_ID\_FPAGE\_IOMEM\_CACHED** 0x80  
*Cached I/O memory.*
- #define **SIGMA0\_REQ\_ID\_FPAGE\_ANY** 0x90  
*Any.*
- #define **SIGMA0\_REQ\_ID\_KIP** 0xA0  
*KIP.*
- #define **SIGMA0\_REQ\_ID\_TBUF** 0xB0  
*TBUF.*
- #define **SIGMA0\_REQ\_ID\_DEBUG\_DUMP** 0xC0  
*Debug dump.*
- #define **SIGMA0\_REQ\_ID\_NEW\_CLIENT** 0xD0  
*New client.*
- #define **SIGMA0\_IS\_MAGIC\_REQ**(d1) ((d1 & SIGMA0\_REQ\_MASK) == SIGMA0\_REQ\_MAGIC)  
*Check if magic.*
- #define **SIGMA0\_REQ(x)** (SIGMA0\_REQ\_MAGIC + SIGMA0\_REQ\_ID\_## x)  
*Construct.*
- #define **SIGMA0\_REQ\_FPAGE\_RAM** (SIGMA0\_REQ(FPAGE\_RAM))  
*RAM.*
- #define **SIGMA0\_REQ\_FPAGE\_IOMEM** (SIGMA0\_REQ(FPAGE\_IOMEM))  
*I/O memory.*
- #define **SIGMA0\_REQ\_FPAGE\_IOMEM\_CACHED** (SIGMA0\_REQ(FPAGE\_IOMEM\_CACHED))

*Cache I/O memory.*

- #define **SIGMA0\_REQ\_FPAGE\_ANY** (SIGMA0\_REQ(FPAGE\_ANY))  
*Any.*
- #define **SIGMA0\_REQ\_KIP** (SIGMA0\_REQ(KIP))  
*KIP.*
- #define **SIGMA0\_REQ\_TBUF** (SIGMA0\_REQ(TBUF))  
*TBUF.*
- #define **SIGMA0\_REQ\_DEBUG\_DUMP** (SIGMA0\_REQ(DEBUG\_DUMP))  
*Debug dump.*
- #define **SIGMA0\_REQ\_NEW\_CLIENT** (SIGMA0\_REQ(NEW\_CLIENT))  
*New client.*

### 9.100.1 Detailed Description

Internal sigma0 definitions.

## 9.101 vCPU Support Library

vCPU handling functionality.

Collaboration diagram for vCPU Support Library:



## Data Structures

- class **L4vcpu::State**  
*C++ implementation of state word in the vCPU area.*
- class **L4vcpu::Vcpu**  
*C++ implementation of the vCPU save state area.*

## Modules

- [Extended vCPU support](#)  
*extended vCPU handling functionality.*

## Typedefs

- [typedef enum l4vcpu\\_irq\\_state\\_t l4vcpu\\_irq\\_state\\_t](#)  
*IRQ/Event enable and disable flags.*

## Enumerations

- [enum l4vcpu\\_irq\\_state\\_t { L4VCPU\\_IRQ\\_STATE\\_DISABLED = 0, L4VCPU\\_IRQ\\_STATE\\_ENABLED = L4\\_VCPU\\_F\\_IRQ }](#)  
*IRQ/Event enable and disable flags.*

## Functions

- [l4vcpu\\_state\\_t l4vcpu\\_state \(l4\\_vcpu\\_state\\_t const \\*vcpu\) throw \(\)](#)  
*Return the state flags of a vCPU.*
- [void l4vcpu\\_irq\\_disable \(l4\\_vcpu\\_state\\_t \\*vcpu\) throw \(\)](#)  
*Disable a vCPU for event delivery.*
- [l4vcpu\\_irq\\_state\\_t l4vcpu\\_irq\\_disable\\_save \(l4\\_vcpu\\_state\\_t \\*vcpu\) throw \(\)](#)  
*Disable a vCPU for event delivery and return previous state.*
- [void l4vcpu\\_irq\\_enable \(l4\\_vcpu\\_state\\_t \\*vcpu, l4\\_utcb\\_t \\*utcb, l4vcpu\\_event\\_hdl\\_t do\\_event\\_work\\_cb, l4vcpu\\_setup\\_ipc\\_t setup\\_ipc\) throw \(\)](#)  
*Enable a vCPU for event delivery.*
- [void l4vcpu\\_irq\\_restore \(l4\\_vcpu\\_state\\_t \\*vcpu, l4vcpu\\_irq\\_state\\_t s, l4\\_utcb\\_t \\*utcb, l4vcpu\\_event\\_hdl\\_t do\\_event\\_work\\_cb, l4vcpu\\_setup\\_ipc\\_t setup\\_ipc\) throw \(\)](#)  
*Restore a previously saved IRQ/event state.*
- [void l4vcpu\\_halt \(l4\\_vcpu\\_state\\_t \\*vcpu, l4\\_utcb\\_t \\*utcb, l4vcpu\\_event\\_hdl\\_t do\\_event\\_work\\_cb, l4vcpu\\_setup\\_ipc\\_t setup\\_ipc\) throw \(\)](#)  
*Halt the vCPU (sleep).*
- [void l4vcpu\\_print\\_state \(l4\\_vcpu\\_state\\_t \\*vcpu, const char \\*prefix\) throw \(\)](#)  
*Print the state of a vCPU.*
- [int l4vcpu\\_is\\_irq\\_entry \(l4\\_vcpu\\_state\\_t \\*vcpu\) throw \(\)](#)  
*Return whether the entry reason was an IRQ/IPC message.*
- [int l4vcpu\\_is\\_page\\_fault\\_entry \(l4\\_vcpu\\_state\\_t \\*vcpu\) throw \(\)](#)  
*Return whether the entry reason was a page fault.*

### 9.101.1 Detailed Description

vCPU handling functionality. This library provides convenience functionality on top of the l4sys vCPU interface to ease programming. It wraps commonly used code and abstracts architecture depends parts as far as reasonable.

### 9.101.2 Enumeration Type Documentation

#### 9.101.2.1 enum l4vcpu\_irq\_state\_t

IRQ/Event enable and disable flags.

Enumerator:

*L4VCPU\_IRQ\_STATE\_DISABLED* IRQ/Event delivery disabled.

*L4VCPU\_IRQ\_STATE\_ENABLED* IRQ/Event delivery enabled.

Definition at line 44 of file [vcpu.h](#).

### 9.101.3 Function Documentation

#### 9.101.3.1 l4vcpu\_state\_t l4vcpu\_state ( l4\_vcpu\_state\_t const \* *vcpu* ) throw () [inline]

Return the state flags of a vCPU.

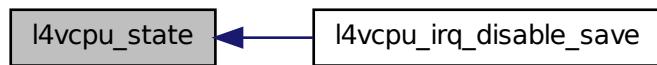
Parameters

*vcpu* Pointer to vCPU area.

Definition at line 225 of file [vcpu.h](#).

Referenced by [l4vcpu\\_irq\\_disable\\_save\(\)](#).

Here is the caller graph for this function:



#### 9.101.3.2 void l4vcpu\_irq\_disable ( l4\_vcpu\_state\_t \* *vcpu* ) throw () [inline]

Disable a vCPU for event delivery.

Parameters

*vcpu* Pointer to vCPU area.

Definition at line 232 of file [vcpu.h](#).

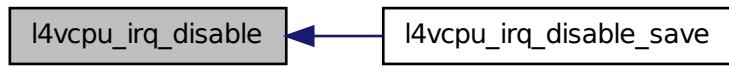
References [l4\\_barrier\(\)](#).

Referenced by [l4vcpu\\_irq\\_disable\\_save\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### **9.101.3.3 l4vcpu\_irq\_state\_t l4vcpu\_irq\_disable\_save ( l4\_vcpu\_state\_t \* vcpu ) throw () [inline]**

Disable a vCPU for event delivery and return previous state.

#### **Parameters**

*vcpu* Pointer to vCPU area.

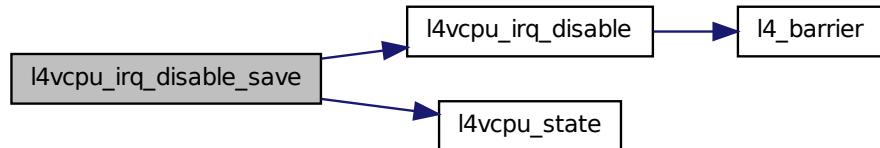
#### **Returns**

IRQ state before disabling IRQs.

Definition at line 240 of file [vcpu.h](#).

References [l4vcpu\\_irq\\_disable\(\)](#), and [l4vcpu\\_state\(\)](#).

Here is the call graph for this function:



**9.101.3.4 void l4vcpu\_irq\_enable ( *l4\_vcpu\_state\_t* \* *vcpu*, *l4\_utcb\_t* \* *utcb*,  
*l4vcpu\_event\_hdl\_t* *do\_event\_work\_cb*, *l4vcpu\_setup\_ipc\_t* *setup\_ipc* ) throw ()  
[*inline*]**

Enable a vCPU for event delivery.

#### Parameters

*vcpu* Pointer to vCPU area.

*utcb* Utcb pointer of the calling vCPU.

*do\_event\_work\_cb* Call-back function that is called in case an event (such as an interrupt) is pending.

*setup\_ipc* Function call-back that is called right before any IPC operation.

< 0 receive and send timeout

Definition at line 263 of file [vcpu.h](#).

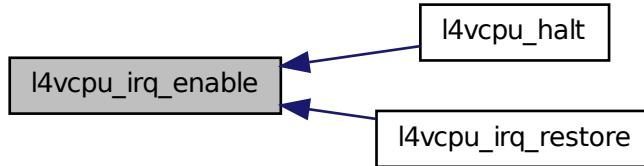
References [EXPECT\\_TRUE](#), [l4\\_barrier\(\)](#), [L4\\_IPC\\_BOTH\\_TIMEOUT\\_0](#), and [L4\\_VCPU\\_SF\\_IRQ\\_PENDING](#).

Referenced by [l4vcpu\\_halt\(\)](#), and [l4vcpu\\_irq\\_restore\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



**9.101.3.5 void l4vcpu\_irq\_restore ( *l4\_vcpu\_state\_t* \* *vcpu*, *l4vcpu\_irq\_state\_t* *s*, *l4\_utcb\_t* \* *utcb*, *l4vcpu\_event\_hdl\_t* *do\_event\_work\_cb*, *l4vcpu\_setup\_ipc\_t* *setup\_ipc* ) throw () [inline]**

Restore a previously saved IRQ/event state.

#### Parameters

*vcpu* Pointer to vCPU area.

*s* IRQ state to be restored.

*utcb* Utcb pointer of the calling vCPU.

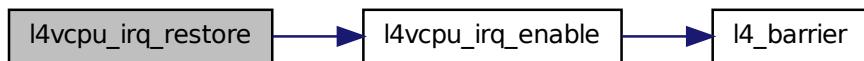
*do\_event\_work\_cb* Call-back function that is called in case an event (such as an interrupt) is pending after enabling.

*setup\_ipc* Function call-back that is called right before any IPC operation.

Definition at line 282 of file [vcpu.h](#).

References [L4\\_VCPU\\_F\\_IRQ](#), and [l4vcpu\\_irq\\_enable\(\)](#).

Here is the call graph for this function:



**9.101.3.6 void l4vcpu\_halt ( *l4\_vcpu\_state\_t* \* *vcpu*, *l4\_utcb\_t* \* *utcb*, *l4vcpu\_event\_hdl\_t* *do\_event\_work\_cb*, *l4vcpu\_setup\_ipc\_t* *setup\_ipc* ) throw () [inline]**

Halt the vCPU (sleep).

### Parameters

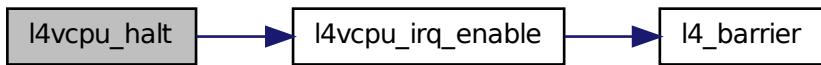
*vcpu* Pointer to vCPU area.  
*utcbl* Utcb pointer of the calling vCPU.  
*do\_event\_work\_cb* Call-back function that is called when the vCPU awakes and needs to handle an event/IRQ.  
*setup\_ipc* Function call-back that is called right before any IPC operation.

< never timeout

Definition at line 293 of file [vcpu.h](#).

References [L4\\_IPC\\_NEVER](#), and [l4vcpu\\_irq\\_enable\(\)](#).

Here is the call graph for this function:



### 9.101.3.7 void l4vcpu\_print\_state ( *l4\_vcpu\_state\_t* \* *vcpu*, *const char* \* *prefix* ) throw ()

Print the state of a vCPU.

### Parameters

*vcpu* Pointer to vCPU area.  
*prefix* A prefix for each line printed.

### 9.101.3.8 int l4vcpu\_is\_irq\_entry ( *l4\_vcpu\_state\_t* \* *vcpu* ) throw () [inline]

Return whether the entry reason was an IRQ/IPC message.

### Parameters

*vcpu* Pointer to vCPU area.

return 0 if not, !=0 otherwise.

### 9.101.3.9 int l4vcpu\_is\_page\_fault\_entry ( *l4\_vcpu\_state\_t* \* *vcpu* ) throw () [inline]

Return whether the entry reason was a page fault.

### Parameters

*vcpu* Pointer to vCPU area.

return 0 if not, !=0 otherwise.

## 9.102 Extended vCPU support

extended vCPU handling functionality.

Collaboration diagram for Extended vCPU support:



### Functions

- `int l4vcpu_ext_alloc (l4_vcpu_state_t **vcpu, l4_addr_t *ext_state, l4_cap_idx_t task, l4_cap_idx_t regmgr) throw ()`

*Allocate state area for an extented vCPU.*

#### 9.102.1 Detailed Description

extended vCPU handling functionality.

#### 9.102.2 Function Documentation

##### 9.102.2.1 `int l4vcpu_ext_alloc ( l4_vcpu_state_t ** vcpu, l4_addr_t * ext_state, l4_cap_idx_t task, l4_cap_idx_t regmgr ) throw ()`

Allocate state area for an extented vCPU.

#### Return values

`vcpu` Allocated vcpu-state area.

`ext_state` Allocated extended vcpu-state area.

#### Parameters

`task` Task to use for allocation.

`regmgr` Region manager to use for allocation.

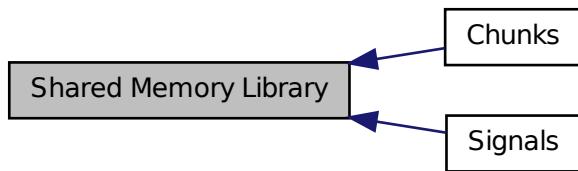
#### Returns

0 for success, error code otherwise

## 9.103 Shared Memory Library

L4SHM provides a shared memory infrastructure that establishes a shared memory area between multiple parties and uses a fast notification mechanism.

Collaboration diagram for Shared Memory Library:



### Modules

- [Chunks](#)
- [Signals](#)

### Functions

- long [l4shmc\\_create](#) (const char \*shmc\_name, [l4\\_umword\\_t](#) shm\_size)  
*Create a shared memory area.*
- long [l4shmc\\_attach](#) (const char \*shmc\_name, l4shmc\_area\_t \*shmarea)  
*Attach to a shared memory area.*
- long [l4shmc\\_attach\\_to](#) (const char \*shmc\_name, [l4\\_umword\\_t](#) timeout\_ms, l4shmc\_area\_t \*shmarea)  
*Attach to a shared memory area, with limited waiting.*
- long [l4shmc\\_connect\\_chunk\\_signal](#) (l4shmc\_chunk\_t \*chunk, l4shmc\_signal\_t \*signal)  
*Connect a signal with a chunk.*
- long [l4shmc\\_area\\_size](#) (l4shmc\_area\_t \*shmarea)  
*Get size of shared memory area.*

#### 9.103.1 Detailed Description

L4SHM provides a shared memory infrastructure that establishes a shared memory area between multiple parties and uses a fast notification mechanism. A shared memory area consists of chunks and signals. A chunk is a defined chunk of memory within the memory area with a maximum size. A chunk is filled

(written) by a producer and read by a consumer. When a producer has finished writing to the chunk it signals a data ready notification to the consumer.

A consumer attaches to a chunk and waits for the producer to fill the chunk. After reading out the chunk it marks the chunk free again.

A shared memory area can have multiple chunks.

The interface is divided in three roles.

- The master role, responsible for setting up a shared memory area.
- A producer, generating data into a chunk
- A consumer, receiving data.

A signal can be connected with a chunk or can be used independently (e.g. for multiple chunks).

### 9.103.2 Function Documentation

#### 9.103.2.1 long l4shmc\_create ( const char \* *shmc\_name*, l4\_umword\_t *shm\_size* )

Create a shared memory area.

##### Parameters

- shmc\_name* Name of the shared memory area.  
*shm\_size* Size of the whole shared memory area.

##### Returns

0 on success, <0 on error

##### Examples:

[examples/libs/shmc/prodcons.c](#).

#### 9.103.2.2 long l4shmc\_attach ( const char \* *shmc\_name*, l4shmc\_area\_t \* *shmarea* ) [inline]

Attach to a shared memory area.

##### Parameters

- shmc\_name* Name of the shared memory area.

##### Return values

- shmarea* Pointer to shared memory area descriptor to be filled with information for the shared memory area.

##### Returns

0 on success, <0 on error

##### Examples:

[examples/libs/shmc/prodcons.c](#).

**9.103.2.3 long l4shmc\_attach\_to ( const char \* *shmc\_name*, l4\_umword\_t *timeout\_ms*, l4shmc\_area\_t \* *shmarea* )**

Attach to a shared memory area, with limited waiting.

#### Parameters

*shmc\_name* Name of the shared memory area.

*timeout\_ms* Timeout to wait for shm area in milliseconds.

#### Return values

*shmarea* Pointer to shared memory area descriptor to be filled with information for the shared memory area.

#### Returns

0 on success, <0 on error

**9.103.2.4 long l4shmc\_connect\_chunk\_signal ( l4shmc\_chunk\_t \* *chunk*, l4shmc\_signal\_t \* *signal* )**

Connect a signal with a chunk.

#### Parameters

*chunk* Chunk to attach the signal to.

*signal* Signal to attach.

#### Returns

0 on success, <0 on error

#### Examples:

[examples/libs/shmc/prodcons.c](#).

**9.103.2.5 long l4shmc\_area\_size ( l4shmc\_area\_t \* *shmarea* ) [inline]**

Get size of shared memory area.

#### Parameters

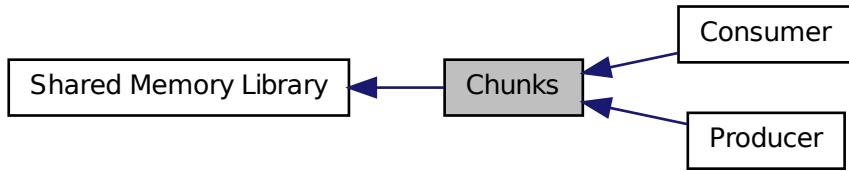
*shmarea* Shared memory area.

#### Returns

<0 on error, otherwise: size of the shared memory area

## 9.104 Chunks

Collaboration diagram for Chunks:



## Modules

- [Producer](#)
- [Consumer](#)

## Functions

- long [`l4shmc\_add\_chunk`](#) (`l4shmc_area_t *shmarea`, `const char *chunk_name`, `l4_umword_t chunk_capacity`, `l4shmc_chunk_t *chunk`)  
*Add a chunk in the shared memory area.*
- long [`l4shmc\_get\_chunk`](#) (`l4shmc_area_t *shmarea`, `const char *chunk_name`, `l4shmc_chunk_t *chunk`)  
*Get chunk out of shared memory area.*
- long [`l4shmc\_get\_chunk\_to`](#) (`l4shmc_area_t *shmarea`, `const char *chunk_name`, `l4_umword_t timeout_ms`, `l4shmc_chunk_t *chunk`)  
*Get chunk out of shared memory area, with timeout.*
- void \* [`l4shmc\_chunk\_ptr`](#) (`l4shmc_chunk_t *chunk`)  
*Get data pointer to chunk.*
- `l4_umword_t l4shmc_chunk_capacity` (`l4shmc_chunk_t *chunk`)  
*Get capacity of a chunk.*
- `l4shmc_signal_t * l4shmc_chunk_signal` (`l4shmc_chunk_t *chunk`)  
*Get the signal of a chunk.*

### 9.104.1 Function Documentation

#### 9.104.1.1 long l4shmc\_add\_chunk ( *l4shmc\_area\_t* \* *shmarea*, *const char* \* *chunk\_name*, *l4\_umword\_t* *chunk\_capacity*, *l4shmc\_chunk\_t* \* *chunk* )

Add a chunk in the shared memory area.

##### Parameters

*shmarea* The shared memory area to put the chunk in.

*chunk\_name* Name of the chunk.

*chunk\_capacity* Capacity for payload of the chunk in bytes.

##### Return values

*chunk* Chunk structure to fill in.

##### Returns

0 on success, <0 on error

##### Examples:

[examples/libs/shmc/prodcons.c](#).

#### 9.104.1.2 long l4shmc\_get\_chunk ( *l4shmc\_area\_t* \* *shmarea*, *const char* \* *chunk\_name*, *l4shmc\_chunk\_t* \* *chunk* ) [inline]

Get chunk out of shared memory area.

##### Parameters

*shmarea* Shared memory area.

*chunk\_name* Name of the chunk.

##### Return values

*chunk* Chunk data structure to fill.

##### Returns

0 on success, <0 on error

##### Examples:

[examples/libs/shmc/prodcons.c](#).

#### 9.104.1.3 long l4shmc\_get\_chunk\_to ( *l4shmc\_area\_t* \* *shmarea*, *const char* \* *chunk\_name*, *l4\_umword\_t* *timeout\_ms*, *l4shmc\_chunk\_t* \* *chunk* )

Get chunk out of shared memory area, with timeout.

##### Parameters

*shmarea* Shared memory area.

*chunk\_name* Name of the chunk.

*timeout\_ms* Timeout in milliseconds to wait for the chunk to appear in the shared memory area.

#### Return values

*chunk* chunk data structure to fill.

#### Returns

0 on success, <0 on error

### 9.104.1.4 void\* l4shmc\_chunk\_ptr ( l4shmc\_chunk\_t \* *chunk* ) [inline]

Get data pointer to chunk.

#### Parameters

*chunk* Chunk.

#### Returns

0 on success, <0 on error

#### Examples:

[examples/libs/shmc/prodcons.c](#).

### 9.104.1.5 l4\_umword\_t l4shmc\_chunk\_capacity ( l4shmc\_chunk\_t \* *chunk* ) [inline]

Get capacity of a chunk.

#### Parameters

*chunk* Chunk.

#### Returns

0 on success, <0 on error

### 9.104.1.6 l4shmc\_signal\_t\* l4shmc\_chunk\_signal ( l4shmc\_chunk\_t \* *chunk* ) [inline]

Get the signal of a chunk.

#### Parameters

*chunk* Chunk.

#### Returns

0 if no signal has been register with this chunk, signal otherwise

## 9.105 Producer

Collaboration diagram for Producer:



### Functions

- long [l4shmc\\_chunk\\_try\\_to\\_take](#) (l4shmc\_chunk\_t \*chunk)  
*Try to mark chunk busy.*
- long [l4shmc\\_chunk\\_ready](#) (l4shmc\_chunk\_t \*chunk, [l4\\_umword\\_t](#) size)  
*Mark chunk as filled (ready).*
- long [l4shmc\\_chunk\\_ready\\_sig](#) (l4shmc\_chunk\_t \*chunk, [l4\\_umword\\_t](#) size)  
*Mark chunk as filled (ready) and signal consumer.*
- long [l4shmc\\_is\\_chunk\\_clear](#) (l4shmc\_chunk\_t \*chunk)  
*Check whether chunk is free.*

#### 9.105.1 Function Documentation

##### 9.105.1.1 long [l4shmc\\_chunk\\_try\\_to\\_take](#) ( l4shmc\_chunk\_t \* *chunk* ) [inline]

Try to mark chunk busy.

#### Parameters

*chunk* chunk to mark.

#### Returns

0 if chunk could be taken, <0 if not (try again then)

#### Examples:

[examples/libs/shmc/prodcons.c](#).

**9.105.1.2 long l4shmc\_chunk\_ready ( *l4shmc\_chunk\_t* \* *chunk*, *l4\_umword\_t* *size* )  
[inline]**

Mark chunk as filled (ready).

**Parameters**

*chunk* chunk.

*size* Size of data in the chunk, in bytes.

**Returns**

0 on success, <0 on error

**9.105.1.3 long l4shmc\_chunk\_ready\_sig ( *l4shmc\_chunk\_t* \* *chunk*, *l4\_umword\_t* *size* )  
[inline]**

Mark chunk as filled (ready) and signal consumer.

**Parameters**

*chunk* chunk.

*size* Size of data in the chunk, in bytes.

**Returns**

0 on success, <0 on error

**Examples:**

[examples/libs/shmc/prodcons.c](#).

**9.105.1.4 long l4shmc\_is\_chunk\_clear ( *l4shmc\_chunk\_t* \* *chunk* ) [inline]**

Check whether chunk is free.

**Parameters**

*chunk* Chunk to check.

**Returns**

0 on success, <0 on error

## 9.106 Consumer

Collaboration diagram for Consumer:



### Functions

- long `l4shmc_enable_chunk` (`l4shmc_chunk_t *chunk`)  
*Enable a signal connected with a chunk.*
- long `l4shmc_wait_chunk` (`l4shmc_chunk_t *chunk`)  
*Wait on a specific chunk.*
- long `l4shmc_wait_chunk_to` (`l4shmc_chunk_t *chunk, l4_timeout_t timeout`)  
*Check whether a specific chunk has an event pending, with timeout.*
- long `l4shmc_wait_chunk_try` (`l4shmc_chunk_t *chunk`)  
*Check whether a specific chunk has an event pending.*
- long `l4shmc_chunk_consumed` (`l4shmc_chunk_t *chunk`)  
*Mark a chunk as free.*
- long `l4shmc_is_chunk_ready` (`l4shmc_chunk_t *chunk`)  
*Check whether data is available.*
- `l4_umword_t l4shmc_chunk_size` (`l4shmc_chunk_t *chunk`)  
*Get current size of a chunk.*

#### 9.106.1 Function Documentation

##### 9.106.1.1 long `l4shmc_enable_chunk` ( `l4shmc_chunk_t * chunk` )

Enable a signal connected with a chunk.

##### Parameters

`chunk` Chunk to enable.

##### Returns

0 on success, <0 on error

A signal must be enabled before waiting when the consumer waits on any signal. Enabling is not needed if the consumer waits for a specific signal or chunk.

#### 9.106.1.2 long l4shmc\_wait\_chunk ( *l4shmc\_chunk\_t* \* *chunk* ) [inline]

Wait on a specific chunk.

##### Parameters

*chunk* Chunk to wait for.

##### Returns

0 on success, <0 on error

##### Examples:

[examples/libs/shmc/prodcons.c](#).

#### 9.106.1.3 long l4shmc\_wait\_chunk\_to ( *l4shmc\_chunk\_t* \* *chunk*, *l4\_timeout\_t* *timeout* )

Check whether a specific chunk has an event pending, with timeout.

##### Parameters

*chunk* Chunk to check.

*timeout* Timeout.

##### Returns

0 on success, <0 on error

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

#### 9.106.1.4 long l4shmc\_wait\_chunk\_try ( *l4shmc\_chunk\_t* \* *chunk* ) [inline]

Check whether a specific chunk has an event pending.

##### Parameters

*chunk* Chunk to check.

##### Returns

0 on success, <0 on error

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

**9.106.1.5 long l4shmc\_chunk\_consumed ( l4shmc\_chunk\_t \* *chunk* ) [inline]**

Mark a chunk as free.

**Parameters**

*chunk* Chunk to mark as free.

**Returns**

0 on success, <0 on error

**Examples:**

[examples/libs/shmc/prodcons.c](#).

**9.106.1.6 long l4shmc\_is\_chunk\_ready ( l4shmc\_chunk\_t \* *chunk* ) [inline]**

Check whether data is available.

**Parameters**

*chunk* Chunk to check.

**Returns**

0 on success, <0 on error

**9.106.1.7 l4\_umword\_t l4shmc\_chunk\_size ( l4shmc\_chunk\_t \* *chunk* ) [inline]**

Get current size of a chunk.

**Parameters**

*chunk* Chunk.

**Returns**

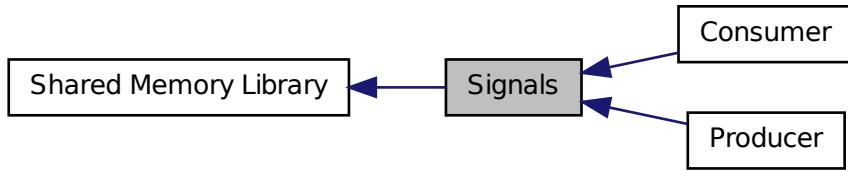
0 on success, <0 on error

**Examples:**

[examples/libs/shmc/prodcons.c](#).

## 9.107 Signals

Collaboration diagram for Signals:



## Modules

- [Producer](#)
- [Consumer](#)

## Functions

- long [`l4shmc\_add\_signal`](#) (`l4shmc_area_t *shmarea`, `const char *signal_name`, `l4shmc_signal_t *signal`)  
*Add a signal for the shared memory area.*
- long [`l4shmc\_attach\_signal`](#) (`l4shmc_area_t *shmarea`, `const char *signal_name`, [`l4\_cap\_idx\_t`](#) `thread`, `l4shmc_signal_t *signal`)  
*Attach to signal.*
- long [`l4shmc\_attach\_signal\_to`](#) (`l4shmc_area_t *shmarea`, `const char *signal_name`, [`l4\_cap\_idx\_t`](#) `thread`, [`l4\_umword\_t`](#) `timeout_ms`, `l4shmc_signal_t *signal`)  
*Attach to signal, with timeout.*
- long [`l4shmc\_get\_signal\_to`](#) (`l4shmc_area_t *shmarea`, `const char *signal_name`, [`l4\_umword\_t`](#) `timeout_ms`, `l4shmc_signal_t *signal`)  
*Get signal object from the shared memory area.*
- [`l4\_cap\_idx\_t l4shmc\_signal\_cap`](#) (`l4shmc_signal_t *signal`)  
*Get the signal capability of a signal.*
- long [`l4shmc\_check\_magic`](#) (`l4shmc_chunk_t *chunk`)  
*Check magic value of a chunk.*

### 9.107.1 Function Documentation

#### 9.107.1.1 long l4shmc\_add\_signal ( *l4shmc\_area\_t* \* *shmarea*, *const char* \* *signal\_name*, *l4shmc\_signal\_t* \* *signal* )

Add a signal for the shared memory area.

##### Parameters

*shmarea* The shared memory area to put the chunk in.

*signal\_name* Name of the signal.

##### Return values

*signal* Signal structure to fill in.

##### Returns

0 on success, <0 on error

##### Examples:

[examples/libs/shmc/prodcons.c](#).

#### 9.107.1.2 long l4shmc\_attach\_signal ( *l4shmc\_area\_t* \* *shmarea*, *const char* \* *signal\_name*, *l4\_cap\_idx\_t* *thread*, *l4shmc\_signal\_t* \* *signal* ) [inline]

Attach to signal.

##### Parameters

*shmarea* Shared memory area.

*signal\_name* Name of the signal.

*thread* Thread capability index to attach the signal to.

##### Return values

*signal* Signal data structure to fill.

##### Returns

0 on success, <0 on error

#### 9.107.1.3 long l4shmc\_attach\_signal\_to ( *l4shmc\_area\_t* \* *shmarea*, *const char* \* *signal\_name*, *l4\_cap\_idx\_t* *thread*, *l4\_umword\_t* *timeout\_ms*, *l4shmc\_signal\_t* \* *signal* )

Attach to signal, with timeout.

##### Parameters

*shmarea* Shared memory area.

*signal\_name* Name of the signal.

*thread* Thread capability index to attach the signal to.

*timeout\_ms* Timeout in milliseconds to wait for the chunk to appear in the shared memory area.

#### Return values

*signal* Signal data structure to fill.

#### Returns

0 on success, <0 on error

#### Examples:

[examples/libs/shmc/prodcons.c](#).

**9.107.1.4 long l4shmc\_get\_signal\_to ( l4shmc\_area\_t \* *shmarea*, const char \* *signal\_name*, l4\_umword\_t *timeout\_ms*, l4shmc\_signal\_t \* *signal* )**

Get signal object from the shared memory area.

#### Parameters

**9.107.1.5 l4\_cap\_idx\_t l4shmc\_signal\_cap ( l4shmc\_signal\_t \* *signal* ) [inline]**

Get the signal capability of a signal.

#### Parameters

*signal* Signal.

#### Returns

Capability of the signal object.

**9.107.1.6 long l4shmc\_check\_magic ( l4shmc\_chunk\_t \* *chunk* ) [inline]**

Check magic value of a chunk.

#### Parameters

*chunk* Chunk.

#### Returns

True if chunk is ok (magic value valid), false if not.

## 9.108 Producer

Collaboration diagram for Producer:



### Functions

- long `l4shmc_trigger` (`l4shmc_signal_t *signal`)

*Trigger a signal.*

#### 9.108.1 Function Documentation

##### 9.108.1.1 long `l4shmc_trigger` (`l4shmc_signal_t * signal`) [inline]

Trigger a signal.

###### Parameters

*signal* Signal to trigger.

###### Returns

0 on success, <0 on error

###### Examples:

[examples/libs/shmc/prodcons.c](#).

## 9.109 Consumer

Collaboration diagram for Consumer:



## Functions

- long `l4shmc_enable_signal` (`l4shmc_signal_t *signal`)  
*Enable a signal.*
- long `l4shmc_wait_any` (`l4shmc_signal_t **retsignal`)  
*Wait on any signal.*
- long `l4shmc_wait_any_try` (`l4shmc_signal_t **retsignal`)  
*Check whether any waited signal has an event pending.*
- long `l4shmc_wait_any_to` (`l4_timeout_t timeout, l4shmc_signal_t **retsignal`)  
*Wait for any signal with timeout.*
- long `l4shmc_wait_signal` (`l4shmc_signal_t *signal`)  
*Wait on a specific signal.*
- long `l4shmc_wait_signal_to` (`l4shmc_signal_t *signal, l4_timeout_t timeout`)  
*Wait on a specific signal, with timeout.*
- long `l4shmc_wait_signal_try` (`l4shmc_signal_t *signal`)  
*Check whether a specific signal has an event pending.*

### 9.109.1 Function Documentation

#### 9.109.1.1 long `l4shmc_enable_signal` ( `l4shmc_signal_t * signal` )

Enable a signal.

##### Parameters

*signal* Signal to enable.

##### Returns

0 on success, <0 on error

A signal must be enabled before waiting when the consumer waits on any signal. Enabling is not needed if the consumer waits for a specific signal or chunk.

#### 9.109.1.2 long `l4shmc_wait_any` ( `l4shmc_signal_t ** retsignal` ) [inline]

Wait on any signal.

##### Return values

*retsignal* Signal received.

##### Returns

0 on success, <0 on error

**9.109.1.3 long l4shmc\_wait\_any\_try( l4shmc\_signal\_t \*\* *retsignal* ) [inline]**

Check whether any waited signal has an event pending.

**Return values**

*retsignal* Signal that has the event pending if any.

**Returns**

0 on success, <0 on error

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

**9.109.1.4 long l4shmc\_wait\_any\_to( l4\_timeout\_t *timeout*, l4shmc\_signal\_t \*\* *retsignal* )**

Wait for any signal with timeout.

**Parameters**

*timeout* Timeout.

**Return values**

*retsignal* Signal that has the event pending if any.

**Returns**

0 on success, <0 on error

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

**9.109.1.5 long l4shmc\_wait\_signal( l4shmc\_signal\_t \* *signal* ) [inline]**

Wait on a specific signal.

**Parameters**

*signal* Signal to wait for.

**Returns**

0 on success, <0 on error

**Examples:**

[examples/libs/shmc/prodcons.c](#).

### 9.109.1.6 long l4shmc\_wait\_signal\_to ( l4shmc\_signal\_t \* signal, l4\_timeout\_t timeout )

Wait on a specific signal, with timeout.

#### Parameters

*signal* Signal to wait for.  
*timeout* Timeout.

#### Returns

0 on success, <0 on error

### 9.109.1.7 long l4shmc\_wait\_signal\_try ( l4shmc\_signal\_t \* signal ) [inline]

Check whether a specific signal has an event pending.

#### Parameters

*signal* Signal to check.

#### Returns

0 on success, <0 on error

The return code indicates whether an event was pending or not. Success means an event was pending, if an receive timeout error is returned no event was pending.

## 9.110 Integer Types

```
#include<l4/sys/l4int.h>
```

Collaboration diagram for Integer Types:



## Files

- file [l4int.h](#)  
*Fixed sized integer types, generic version.*
- file [l4int.h](#)

*Fixed sized integer types, arm version.*

- file [l4int.h](#)

*Fixed sized integer types, amd64 version.*

- file [l4int.h](#)

*Fixed sized integer types, x86 version.*

## Defines

- `#define L4_MWORD_BITS 32`

*Size of machine words in bits.*

- `#define L4_MWORD_BITS 64`

*Size of machine words in bits.*

- `#define L4_MWORD_BITS 32`

*Size of machine words in bits.*

## Typedefs

- `typedef signed char l4_int8_t`

*Signed 8bit value.*

- `typedef unsigned char l4_uint8_t`

*Unsigned 8bit value.*

- `typedef signed short int l4_int16_t`

*Signed 16bit value.*

- `typedef unsigned short int l4_uint16_t`

*Unsigned 16bit value.*

- `typedef signed int l4_int32_t`

*Signed 32bit value.*

- `typedef unsigned int l4_uint32_t`

*Unsigned 32bit value.*

- `typedef signed long long l4_int64_t`

*Signed 64bit value.*

- `typedef unsigned long long l4_uint64_t`

*Unsigned 64bit value.*

- `typedef unsigned long l4_addr_t`

*Address type.*

- **typedef signed long l4\_mword\_t**  
*Signed machine word.*
- **typedef unsigned long l4\_umword\_t**  
*Unsigned machine word.*
- **typedef l4\_uint64\_t l4\_cpu\_time\_t**  
*CPU clock type.*
- **typedef l4\_uint64\_t l4\_kernel\_clock\_t**  
*Kernel clock type.*
- **typedef unsigned int l4\_size\_t**  
*Signed size type.*
- **typedef signed int l4\_ssize\_t**  
*Unsigned size type.*
- **typedef unsigned long l4\_size\_t**  
*Signed size type.*
- **typedef signed long l4\_ssize\_t**  
*Unsigned size type.*
- **typedef unsigned int l4\_size\_t**  
*Signed size type.*
- **typedef signed int l4\_ssize\_t**  
*Unsigned size type.*

### 9.110.1 Detailed Description

```
#include<l4/sys/l4int.h>
```

### 9.110.2 Typedef Documentation

#### 9.110.2.1 **typedef signed char l4\_int8\_t**

Signed 8bit value.

Definition at line 35 of file [l4int.h](#).

#### 9.110.2.2 **typedef unsigned char l4\_uint8\_t**

Unsigned 8bit value.

Definition at line 36 of file [l4int.h](#).

**9.110.2.3 `typedef signed short int l4_int16_t`**

Signed 16bit value.

Definition at line [37](#) of file [l4int.h](#).

**9.110.2.4 `typedef unsigned short int l4_uint16_t`**

Unsigned 16bit value.

Definition at line [38](#) of file [l4int.h](#).

**9.110.2.5 `typedef signed int l4_int32_t`**

Signed 32bit value.

Definition at line [39](#) of file [l4int.h](#).

**9.110.2.6 `typedef unsigned int l4_uint32_t`**

Unsigned 32bit value.

Definition at line [40](#) of file [l4int.h](#).

**9.110.2.7 `typedef signed long long l4_int64_t`**

Signed 64bit value.

Definition at line [41](#) of file [l4int.h](#).

**9.110.2.8 `typedef unsigned long long l4_uint64_t`**

Unsigned 64bit value.

Definition at line [42](#) of file [l4int.h](#).



# Chapter 10

## Namespace Documentation

### 10.1 cxx Namespace Reference

Various kinds of C++ utilities.

#### Namespaces

- namespace [Bits](#)

*Internal helpers for the cxx package.*

#### Data Structures

- class [Auto\\_ptr](#)

*Smart pointer with automatic deletion.*

- class [Avl\\_map](#)

*AVL tree based associative container.*

- class [Avl\\_set](#)

*AVL Tree for simple comparable items.*

- class [Avl\\_tree\\_node](#)

*Node of an AVL tree.*

- class [Avl\\_tree](#)

*A generic AVL tree.*

- class [Bitmap\\_base](#)

*Basic bitmap abstraction.*

- class [Bitmap](#)

*A static bit map.*

- class [List\\_item](#)  
*Basic list item.*
- class [List](#)  
*Doubly linked list, with internal allocation.*
- class [List\\_alloc](#)  
*Standard list-based allocator.*
- struct [Pair](#)  
*Pair of two values.*
- class [Pair\\_first\\_compare](#)  
*Comparison functor for [Pair](#).*
- class [Base\\_slab](#)  
*Basic slab allocator.*
- class [Slab](#)  
*Slab allocator for object of type [Type](#).*
- class [Base\\_slab\\_static](#)  
*Merged slab allocator (allocators for objects of the same size are merged together).*
- class [Slab\\_static](#)  
*Merged slab allocator (allocators for objects of the same size are merged together).*
- class [Nothrow](#)  
*Helper type to distinguish the operator new version that does not throw exceptions.*
- class [New\\_allocator](#)  
*Standard allocator based on operator new () .*
- struct [Lt\\_functor](#)  
*Generic comparator class that defaults to the less-than operator.*

## Functions

- template<typename T1 >  
**T1 min (T1 a, T1 b)**  
*Get the minimum of a and b.*
- template<typename T1 >  
**T1 max (T1 a, T1 b)**  
*Get the maximum of a and b.*

### 10.1.1 Detailed Description

Various kinds of C++ utilities.

## 10.2 cxx::Bits Namespace Reference

Internal helpers for the cxx package.

### Data Structures

- class [Bst](#)  
*Basic binary search tree (BST).*
- struct [Direction](#)  
*The direction to go in a binary search tree.*
- class [Bst\\_node](#)  
*Basic type of a node in a binary search tree (BST).*

### 10.2.1 Detailed Description

Internal helpers for the cxx package.

## 10.3 L4 Namespace Reference

[L4](#) low-level kernel interface.

### Namespaces

- namespace [Ipc\\_svr](#)  
*Helper classes for [L4::Server](#) instantiation.*

### Data Structures

- class [Alloc\\_list](#)  
*A simple list-based allocator.*
- class [IOModifier](#)  
*Modifier class for the IO stream.*
- class [Exception\\_tracer](#)  
*Back-trace support for exceptions.*
- class [Base\\_exception](#)

*Base class for all exceptions, thrown by the L4Re framework.*

- class [Runtime\\_error](#)  
*Exception for an abstract runtime error.*
- class [Out\\_of\\_memory](#)  
*Exception signalling insufficient memory.*
- class [Element\\_already\\_exists](#)  
*Exception for duplicate element insertions.*
- class [Unknown\\_error](#)  
*Exception for an unknown condition.*
- class [Element\\_not\\_found](#)  
*Exception for a failed lookup (element not found).*
- class [Invalid\\_capability](#)  
*Indicates that an invalid object was invoked.*
- class [Com\\_error](#)  
*Error conditions during IPC.*
- class [Bounds\\_error](#)  
*Access out of bounds.*
- class [Server](#)  
*Basic server loop for handling client requests.*
- class [Server\\_object](#)  
*Abstract server object to be used with the L4::Registry\_dispatcher.*
- class [Basic\\_registry](#)  
*This registry returns the corresponding server object based on the label.*
- class [String](#)  
*A null-terminated string container class.*
- struct [Type\\_info](#)  
*Dynamic Type Information for L4Re Interfaces.*
- class [Kobject\\_t](#)  
*Helper class to create an L4Re interface class that is derived from a single base class.*
- class [Kobject\\_2t](#)  
*Helper class to create an L4Re interface class that is derived from two base classes.*
- class [Vm](#)  
*Virtual machine.*

- class [Cap\\_base](#)  
*Base class for all kinds of capabilities.*
- class [Cap](#)  
*Capability Selector a la C++.*
- class [Kobject](#)  
*Base class for all kinds of kernel objects, referred to by capabilities.*
- class [Debugger](#)  
*Debugger interface.*
- class [Factory](#)  
*C++ [L4 Factory](#), to create all kinds of kernel objects.*
- class [Ipc\\_gate](#)  
*[L4 IPC gate](#).*
- class [Irq](#)  
*C++ version of an [L4 IRQ](#).*
- class [Icu](#)  
*C++ version of an interrupt controller.*
- class [Meta](#)  
*Meta interface that shall be implemented by each [L4Re](#) object and gives access to the dynamic type information for [L4Re](#) objects.*
- class [Scheduler](#)  
*Scheduler object.*
- class [Smart\\_cap](#)  
*Smart capability class.*
- class [Task](#)  
*An [L4 Task](#).*
- class [Thread](#)  
*[L4 kernel thread](#).*
- class [Vcon](#)  
*C++ [L4 Vcon](#).*

## Functions

- template<typename T >  
[Type\\_info](#) const \* [kobject\\_typeid](#) ()  
*Get the [L4::Type\\_info](#) for the [L4Re](#) interface given in T.*

- template<typename T , typename F >  
`Cap< T > cap_cast (Cap< F > const &c) throw ()`  
*static\_cast for capabilities.*
- template<typename T , typename F >  
`Cap< T > cap_reinterpret_cast (Cap< F > const &c) throw ()`  
*reinterpret\_cast for capabilities.*
- template<typename T , typename F >  
`Cap< T > cap_dynamic_cast (Cap< F > const &c) throw ()`  
*dynamic\_cast for capabilities.*
- template<typename T , typename F , typename SMART >  
`Smart_cap< T, SMART > cap_cast (Smart_cap< F, SMART > const &c) throw ()`  
*static\_cast for capabilities.*
- template<typename T , typename F , typename SMART >  
`Smart_cap< T, SMART > cap_reinterpret_cast (Smart_cap< F, SMART > const &c) throw ()`  
*reinterpret\_cast for capabilities.*

## Variables

- `IOModifier const hex`  
*Modifies the stream to print numbers as hexadecimal values.*
- `IOModifier const dec`  
*Modifies the stream to print numbers as decimal values.*
- `BasicOStream cout`  
*Standard output stream.*
- `BasicOStream cerr`  
*Standard error stream.*

### 10.3.1 Detailed Description

[L4](#) low-level kernel interface.

### 10.3.2 Function Documentation

#### 10.3.2.1 template<typename T > Type\_info const\* L4::kobject\_typeid( ) [inline]

Get the [L4::Type\\_info](#) for the [L4Re](#) interface given in *T*.

##### Parameters

*T* The type ([L4Re](#) interface) for which the information shall be returned.

### Returns

A pointer to the [L4::Type\\_info](#) structure for  $T$ .

Definition at line 87 of file [\\_\\_typeinfo.h](#).

## 10.4 L4::Ipc\_svr Namespace Reference

Helper classes for [L4::Server](#) instantiation.

### Data Structures

- struct [Ignore\\_errors](#)  
*Mix in for LOOP\_HOOKS to ignore IPC errors.*
- struct [Default\\_timeout](#)  
*Mix in for LOOP\_HOOKS to use a 0 send and a infinite receive timeout.*
- struct [Compound\\_reply](#)  
*Mix in for LOOP\_HOOKS to always use compound reply and wait.*
- struct [Default\\_setup\\_wait](#)  
*Mix in for LOOP\_HOOKS for setup\_wait no op.*
- struct [Default\\_loop\\_hooks](#)  
*Default LOOP\_HOOKS.*

### Enumerations

- enum [Reply\\_mode](#) { [Reply\\_compound](#), [Reply\\_separate](#) }  
*Reply mode for server loop.*

### 10.4.1 Detailed Description

Helper classes for [L4::Server](#) instantiation.

### 10.4.2 Enumeration Type Documentation

#### 10.4.2.1 enum L4::Ipc\_svr::Reply\_mode

Reply mode for server loop.

The reply mode specifies if the server loop shall do a compound reply and wait operation ([Reply\\_compound](#)), which is the most performant method. Note, [setup\\_wait\(\)](#) is called before the reply. The other way is to call [reply](#) and [wait](#) separately and call [setup\\_wait](#) in between.

The actual mode is determined by the return value of the [before\\_reply\(\)](#) hook in the LOOP\_HOOKS of [L4::Server](#).

**Enumerator:**

*Reply\_compound* Server shall use a compound reply and wait (fast).

*Reply\_separate* Server shall call reply and wait separately.

Definition at line 44 of file [ipc\\_server](#).

## 10.5 L4Re Namespace Reference

[L4](#) Runtime Environment.

### Namespaces

- namespace [Vfs](#)

*Virtual file system for interfaces POSIX libc.*

### Data Structures

- class [Cap\\_alloc](#)

*Capability allocator interface.*

- class [Smart\\_cap\\_auto](#)

*Helper for Auto\_cap and Auto\_del\_cap.*

- class [Console](#)

*Console class.*

- class [Dataspace](#)

*This class represents a data space.*

- class [Debug\\_obj](#)

*Debug interface.*

- class [Env](#)

*Initial Environment (C++ version).*

- class [Event](#)

*Event class.*

- class [Event\\_buffer\\_t](#)

*Event buffer class.*

- class [Log](#)

*Log interface class.*

- class [Mem\\_alloc](#)

*Memory allocator.*

- class [Namespace](#)  
*Name-space interface.*
- class [Parent](#)  
*Parent interface.*
- class [Rm](#)  
*Region map.*

### 10.5.1 Detailed Description

[L4](#) Runtime Environment.

## 10.6 L4Re::Vfs Namespace Reference

Virtual file system for interfaces POSIX libc.

### Data Structures

- class [Be\\_file](#)  
*Boiler plate class for implementing an open file for [L4Re::Vfs](#).*
- class [Be\\_file\\_system](#)  
*Boilerplate class for implementing a [L4Re::Vfs::File\\_system](#).*
- class [Generic\\_file](#)  
*The common interface for an open POSIX file.*
- class [Directory](#)  
*Interface for a POSIX file that is a directory.*
- class [Regular\\_file](#)  
*Interface for a POSIX file that provides regular file semantics.*
- class [Special\\_file](#)  
*Interface for a POSIX file that provides special file semantics.*
- class [File](#)  
*The basic interface for an open POSIX file.*
- class [Mman](#)  
*Interface for the POSIX memory management.*
- class [File\\_system](#)  
*Basic interface for an [L4Re::Vfs](#) file system.*

- class **Fs**  
*POSIX File-system related functionality.*
- class **Ops**  
*Interface for the POSIX backends for an application.*

## Functions

- **L4Re::Vfs::Ops** \*vfs\_ops **asm** ("l4re\_env\_posix\_vfs\_ops")  
*Reference to the applications **L4Re::Vfs::Ops** singleton.*

### 10.6.1 Detailed Description

Virtual file system for interfaces POSIX libc.

# Chapter 11

## Data Structure Documentation

### 11.1 L4::Alloc\_list Class Reference

A simple list-based allocator.

```
#include <alloc.h>
```

#### 11.1.1 Detailed Description

A simple list-based allocator.

Definition at line 33 of file [alloc.h](#).

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

- l4/cxx/alloc.h

### 11.2 L4::Thread::Attr Class Reference

[Thread](#) attributes used for control\_commit().

#### Public Member Functions

- **Attr** ([l4\\_utcb\\_t](#) \*utcbl4\_utcb()) throw()  
*Create a thread-attribute object with the given UTCB.*
- void **pager** ([Cap<void>](#) const &pager) throw()  
*Set the pager capability selector.*
- [Cap<void>](#) **pager** () throw()  
*Get the capability selector used for page-fault messages.*
- void **exc\_handler** ([Cap<void>](#) const &exc\_handler) throw()  
*Set the exception-handler capability selector.*

- `Cap< void > exc_handler () throw ()`  
*Get the capability selector used for exception messages.*
- `void bind (l4_utcb_t *thread_utcb, Cap< Task > const &task) throw ()`  
*Bind the thread to a task.*
- `void alien (int on) throw ()`  
*Set the thread to alien mode.*
- `void ux_host_syscall (int on) throw ()`  
*Allow host system calls on Fiasco-UX.*

## Friends

- class [L4::Thread](#)

### 11.2.1 Detailed Description

[Thread](#) attributes used for control\_commit(). This class is responsible for initializing various attributes of a thread in a UTCB for the control\_commit() method.

#### See also

[Thread control](#) for some more details.

Definition at line 70 of file [thread](#).

### 11.2.2 Constructor & Destructor Documentation

#### 11.2.2.1 `L4::Thread::Attr::Attr ( l4_utcb_t * utcb = l4_utcb () ) throw () [inline, explicit]`

Create a thread-attribute object with the given UTCB.

#### Parameters

`utcb` the UTCB to use for the later L4::Thread::control\_commit() function. Usually this is the UTCB of the calling thread.

Definition at line 82 of file [thread](#).

### 11.2.3 Member Function Documentation

#### 11.2.3.1 `void L4::Thread::Attr::pager ( Cap< void > const & pager ) throw () [inline]`

Set the pager capability selector.

**Parameters**

**pager** the capability selector that shall be used for page-fault messages. This capability selector must be valid within the task the thread is bound to.

Definition at line 91 of file [thread](#).

References [pager\(\)](#).

Here is the call graph for this function:

**11.2.3.2 Cap<void> L4::Thread::Attr::pager( ) throw() [inline]**

Get the capability selector used for page-fault messages.

**Returns**

the capability selector used to send page-fault messages. The selector is valid in the task the thread is bound to.

Definition at line 99 of file [thread](#).

References [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [pager\(\)](#).

Here is the caller graph for this function:

**11.2.3.3 void L4::Thread::Attr::exc\_handler( Cap< void > const & exc\_handler ) throw() [inline]**

Set the exception-handler capability selector.

## Parameters

**pager** the capability selector that shall be used for exception messages. This capability selector must be valid within the task the thread is bound to.

Definition at line 108 of file [thread](#).

References [exc\\_handler\(\)](#).

Here is the call graph for this function:



### 11.2.3.4 Cap<void> L4::Thread::Attr::exc\_handler( ) throw() [inline]

Get the capability selector used for exception messages.

## Returns

the capability selector used to send exception messages. The selector is valid in the task the thread is bound to.

Definition at line 116 of file [thread](#).

References [l4\\_msg\\_regs\\_t::mr](#).

Referenced by [exc\\_handler\(\)](#).

Here is the caller graph for this function:



### 11.2.3.5 void L4::Thread::Attr::bind( l4\_utcb\_t \* thread\_utcb, Cap< Task > const & task ) throw() [inline]

Bind the thread to a task.

### Parameters

*thread\_utcb* the UTCB address of the thread within the task specified by *task*.

*task* the capability selector for the task the thread shall be bound to.

Binding a thread to a task means that the thread shall afterwards execute in the given task. To actually start execution you need to use [L4::Thread::ex\\_regs\(\)](#).

Definition at line 130 of file [thread](#).

### 11.2.3.6 void L4::Thread::Attr::ux\_host\_syscall ( int *on* ) throw () [inline]

Allow host system calls on Fiasco-UX.

### Precondition

Running on Fiasco-UX.

Definition at line 143 of file [thread](#).

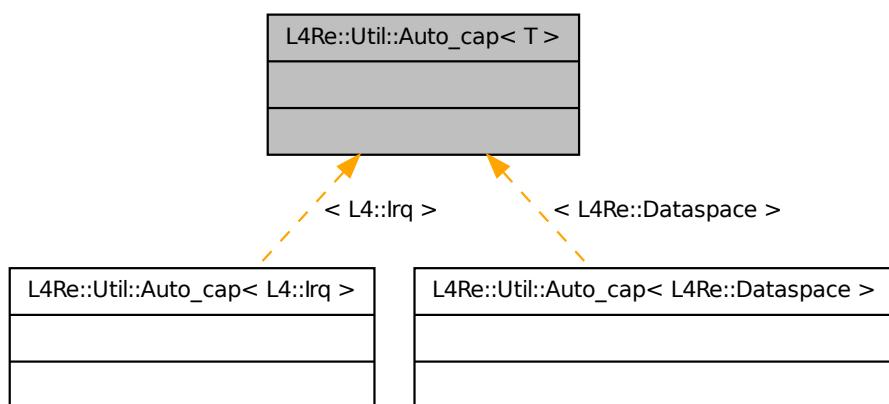
The documentation for this class was generated from the following file:

- [l4/sys/thread](#)

## 11.3 L4Re::Util::Auto\_cap< T > Struct Template Reference

Automatic capability that implements automatic free and unmap of the capability selector.

Inheritance diagram for L4Re::Util::Auto\_cap< T >:



### 11.3.1 Detailed Description

**template<typename T> struct L4Re::Util::Auto\_cap< T >**

Automatic capability that implements automatic free and unmap of the capability selector.

#### Parameters

*T* the type of the object that is referred by the capability.

This kind of automatic capability is useful for capabilities with that shall have a lifetime that is strictly coupled to one C++ scope.

Usage:

```
{
    L4Re::Util::Auto_cap<L4Re::Dataspace>::Cap
    ds_cap(L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>());
    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));
    ...
    // At the end of the scope ds_cap is unmapped and the capability selector
    // is freed.
}
```

Definition at line 155 of file [cap\\_alloc](#).

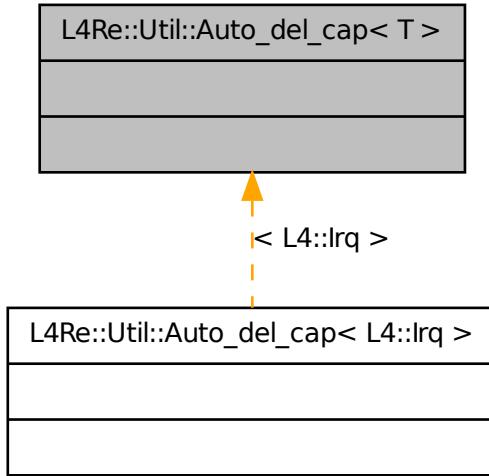
The documentation for this struct was generated from the following file:

- [l4/re/util/cap\\_alloc](#)

## 11.4 L4Re::Util::Auto\_del\_cap< T > Struct Template Reference

Automatic capability that implements automatic free and unmap+delete of the capability selector.

Inheritance diagram for L4Re::Util::Auto\_del\_cap< T >:



### 11.4.1 Detailed Description

**template<typename T> struct L4Re::Util::Auto\_del\_cap< T >**

Automatic capability that implements automatic free and unmap+delete of the capability selector.

#### Parameters

*T* the type of the object that is referred by the capability.

This kind of automatic capability is useful for capabilities with that shall have a lifetime that is strictly coupled to one C++ scope. The main difference to [Auto\\_cap](#) is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Usage:

```

{
    L4Re::Util::Auto_del_cap<L4Re::Dataspace>::Cap
    ds_cap(L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>());

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

    ...
    // At the end of the scope ds_cap is unmapped and the capability selector
    // is freed. Because the deletion flag is set the data space shall be
    // also deleted (even if there are other references to this data space).
}

```

Definition at line 189 of file [cap\\_alloc](#).

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

- l4/re/util/cap\_alloc

## 11.5 cxx::Auto\_ptr< T > Class Template Reference

Smart pointer with automatic deletion.

### Public Types

- `typedef T Ref_type`

*The referenced type.*

### Public Member Functions

- `Auto_ptr (T *p=0) throw ()`  
*Construction by assignment of a normal pointer.*
- `Auto_ptr (Auto_ptr const &o) throw ()`  
*Copy construction, releases the original pointer.*
- `Auto_ptr & operator=(Auto_ptr const &o) throw ()`  
*Assignment from another smart pointer.*
- `~Auto_ptr () throw ()`  
*Destruction, shall delete the object.*
- `T & operator* () const throw ()`  
*Dereference the pointer.*
- `T * operator-> () const throw ()`  
*Member access for the object.*
- `T * get () const throw ()`  
*Get the normal pointer.*
- `T * release () throw ()`  
*Release the object and get the normal pointer back.*
- `void reset (T *p=0) throw ()`  
*Delete the object and reset the smart pointer to NULL.*
- `operator Priv_type * () const throw ()`  
*Operator for if (!ptr) ... <>.*

### 11.5.1 Detailed Description

**template<typename T> class cxx::Auto\_ptr< T >**

Smart pointer with automatic deletion.

#### Parameters

*T* The type of the referenced object.

This smart pointer calls the delete operator when the destructor is called. This has the effect that the object the pointer points to will be deleted when the pointer goes out of scope, or a new value gets assigned. The smart pointer provides a [release\(\)](#) method to get a normal pointer to the object and set the smart pointer to NULL.

Definition at line [36](#) of file [auto\\_ptr](#).

### 11.5.2 Member Typedef Documentation

**11.5.2.1 template<typename T> typedef T cxx::Auto\_ptr< T >::Ref\_type**

The referenced type.

Definition at line [41](#) of file [auto\\_ptr](#).

### 11.5.3 Constructor & Destructor Documentation

**11.5.3.1 template<typename T> cxx::Auto\_ptr< T >::Auto\_ptr ( T \* *p* = *o* ) throw ()  
[inline, explicit]**

Construction by assignment of a normal pointer.

#### Parameters

*p* The pointer to the object

Definition at line [51](#) of file [auto\\_ptr](#).

**11.5.3.2 template<typename T> cxx::Auto\_ptr< T >::Auto\_ptr ( Auto\_ptr< T > const & *o* )  
throw () [inline]**

Copy construction, releases the original pointer.

#### Parameters

*o* The smart pointer, which shall be copied and released.

Definition at line [57](#) of file [auto\\_ptr](#).

**11.5.3.3 template<typename T> cxx::Auto\_ptr< T >::~Auto\_ptr ( ) throw () [inline]**

Destruction, shall delete the object.

Definition at line [76](#) of file [auto\\_ptr](#).

### 11.5.4 Member Function Documentation

**11.5.4.1 template<typename T> Auto\_ptr& cxx::Auto\_ptr< T >::operator=( Auto\_ptr< T > const & *o* ) throw() [inline]**

Assignment from another smart pointer.

#### Parameters

*o* The source for the assignment (will be released).

Definition at line 65 of file [auto\\_ptr](#).

References [cxx::Auto\\_ptr< T >::release\(\)](#).

Here is the call graph for this function:



**11.5.4.2 template<typename T> T& cxx::Auto\_ptr< T >::operator\*( ) const throw() [inline]**

Dereference the pointer.

Definition at line 80 of file [auto\\_ptr](#).

**11.5.4.3 template<typename T> T\* cxx::Auto\_ptr< T >::operator->( ) const throw() [inline]**

Member access for the object.

Definition at line 83 of file [auto\\_ptr](#).

**11.5.4.4 template<typename T> T\* cxx::Auto\_ptr< T >::get( ) const throw() [inline]**

Get the normal pointer.

#### Attention

This function will not release the object, the object will be deleted by the smart pointer.

Definition at line 90 of file [auto\\_ptr](#).

**11.5.4.5 template<typename T> T\* cxx::Auto\_ptr< T >::release( ) throw() [inline]**

Release the object and get the normal pointer back.

After calling this function the smart pointer will point to NULL and the object will not be deleted by the pointer anymore.

Definition at line 98 of file [auto\\_ptr](#).

Referenced by [cxx::Auto\\_ptr< T >::operator=\(\)](#).

Here is the caller graph for this function:

**11.5.4.6 template<typename T> cxx::Auto\_ptr< T >::operator Priv\_type \* ( ) const throw() [inline]**

Operator for if (!ptr) ... < >.

Definition at line 110 of file [auto\\_ptr](#).

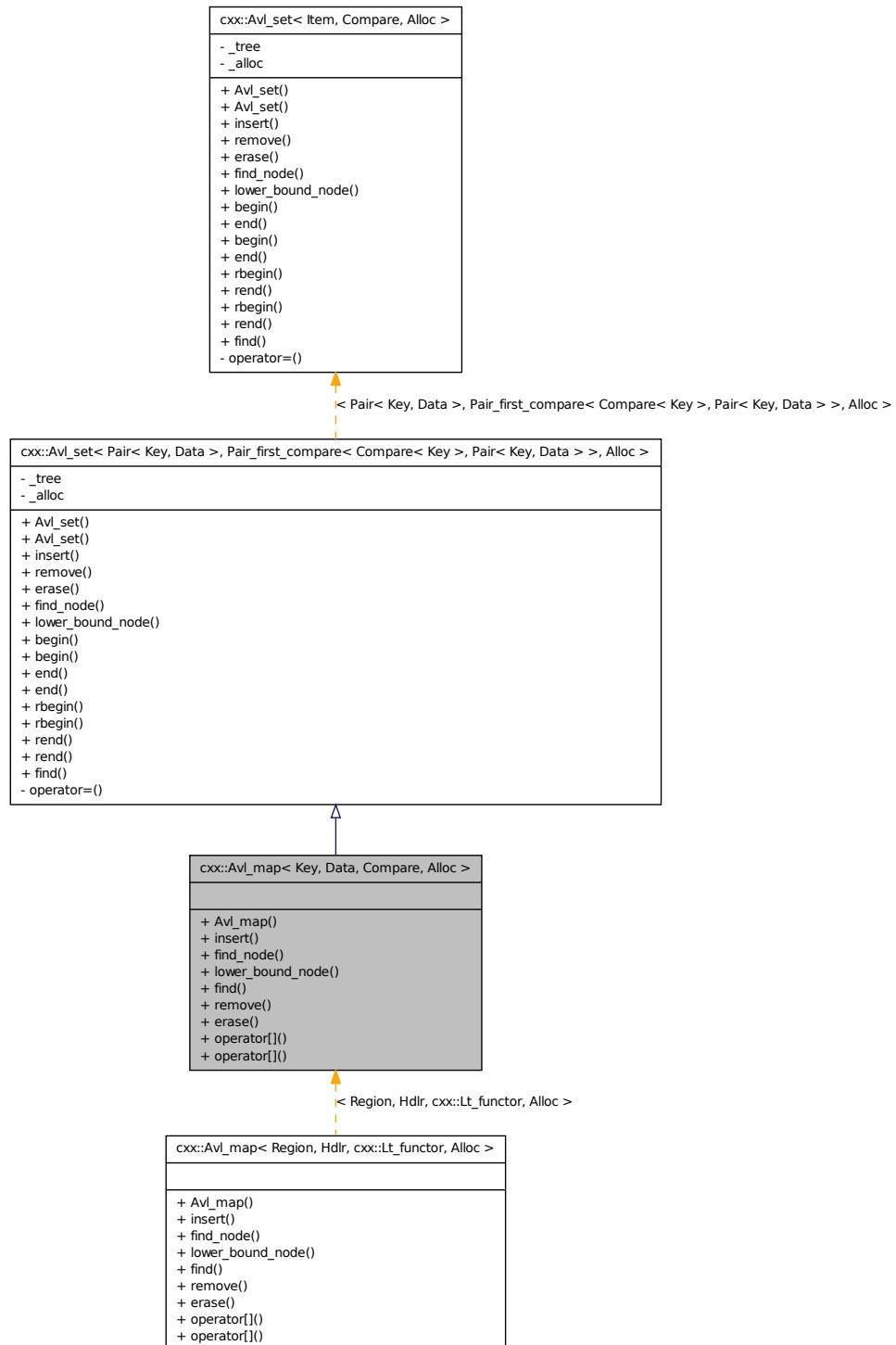
The documentation for this class was generated from the following file:

- l4/cxx/auto\_ptr

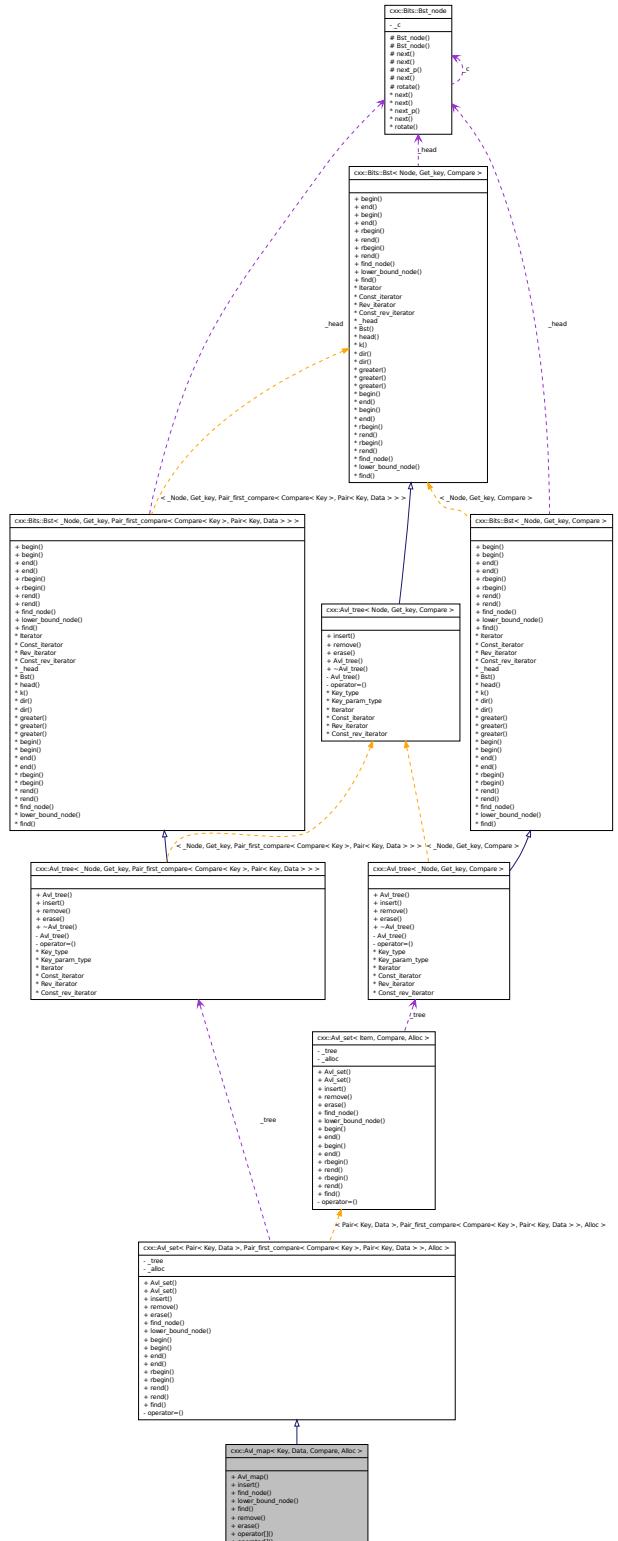
## 11.6 cxx::Avl\_map< Key, Data, Compare, Alloc > Class Template Reference

AVL tree based associative container.

Inheritance diagram for cxx::Avl\_map< Key, Data, Compare, Alloc >:



## Collaboration diagram for cxx::Avl\_map< Key, Data, Compare, Alloc >:



## Public Types

- **typedef Compare< Key > Key\_compare**  
*Type of the comparison functor.*
- **typedef Key Key\_type**  
*Type of the key values.*
- **typedef Data Data\_type**  
*Type of the data values.*
- **typedef Base\_type::Node Node**  
*Return type for find.*
- **typedef Base\_type::Node\_allocator Node\_allocator**  
*Type of the allocator.*
- **typedef Base\_type::Iterator Iterator**  
*Forward iterator for the set.*
- **typedef Base\_type::Const\_iterator Const\_iterator**  
*Constant forward iterator for the set.*
- **typedef Base\_type::Rev\_iterator Rev\_iterator**  
*Backward iterator for the set.*
- **typedef Base\_type::Const\_rev\_iterator Const\_rev\_iterator**  
*Constant backward iterator for the set.*

## Public Member Functions

- **Avl\_map (Node\_allocator const &alloc=Node\_allocator())**  
*Create an empty AVL-tree based map.*
- **cxx::Pair< Iterator, int > insert (Key\_type const &key, Data\_type const &data)**  
*Insert a <key, data> pair into the map.*
- **Node find\_node (Key\_type const &key) const**  
*Find a <key, data> pair for a given key.*
- **Node lower\_bound\_node (Key\_type const &key) const**  
*Find the first node greater or equal to key.*
- **Iterator find (Key\_type const &key) const**  
*Find a <key, data> pair for a given key.*
- **int remove (Key\_type const &key)**  
*Remove the <key, data> pair for the given key.*

- int `erase (Key_type const &key)`  
*Removed the element key.*
- `Data_type const & operator[ ] (Key_type const &key) const`  
*Get the data for the given key.*
- `Data_type & operator[ ] (Key_type const &key)`  
*Get the data for the given key.*

### 11.6.1 Detailed Description

```
template<typename Key, typename Data, template< typename A > class Compare = Lt_functor,
template< typename B > class Alloc = New_allocator> class cxx::Avl_map< Key, Data, Compare,
Alloc >
```

AVL tree based associative container.

#### Parameters

- Key* Type of the key values.  
*Data* Type of the data values.  
*Compare* Type comparison functor for the key values.  
*Alloc* Type of the allocator used for the nodes.

Definition at line 43 of file [avl\\_map](#).

### 11.6.2 Constructor & Destructor Documentation

- 11.6.2.1 `template<typename Key, typename Data, template< typename A > class Compare =
Lt_functor, template< typename B > class Alloc = New_allocator> cxx::Avl_map< Key,
Data, Compare, Alloc >::Avl_map ( Node_allocator const & alloc = Node_allocator () ) [inline]`

Create an empty AVL-tree based map.

#### Parameters

- comp* The comparison functor.  
*alloc* The node allocator.

Definition at line 62 of file [avl\\_map](#).

### 11.6.3 Member Function Documentation

- 11.6.3.1 `template<typename Key, typename Data, template< typename A > class Compare =
Lt_functor, template< typename B > class Alloc = New_allocator> cxx::Pair<Iterator,
int> cxx::Avl_map< Key, Data, Compare, Alloc >::insert ( Key_type const & key,
Data_type const & data ) [inline]`

Insert a <key, data> pair into the map.

**Parameters**

*key* The key value.

*data* The data value to insert.

Definition at line 71 of file [avl\\_map](#).

**11.6.3.2 template<typename Key, typename Data, template< typename A > class Compare = Lt\_functor, template< typename B > class Alloc = New\_allocator> Node cxx::Avl\_map< Key, Data, Compare, Alloc >::find\_node ( Key\_type const & key ) const [inline]**

Find a <key, data> pair for a given key.

**Parameters**

*key* The key value to use for the lookup.

Definition at line 78 of file [avl\\_map](#).

**11.6.3.3 template<typename Key, typename Data, template< typename A > class Compare = Lt\_functor, template< typename B > class Alloc = New\_allocator> Node cxx::Avl\_map< Key, Data, Compare, Alloc >::lower\_bound\_node ( Key\_type const & key ) const [inline]**

Find the first node greater or equal to *key*.

**Parameters**

*key* the key to look for.

**Returns**

The first node greater or equal to *key*.

Definition at line 86 of file [avl\\_map](#).

**11.6.3.4 template<typename Key, typename Data, template< typename A > class Compare = Lt\_functor, template< typename B > class Alloc = New\_allocator> Iterator cxx::Avl\_map< Key, Data, Compare, Alloc >::find ( Key\_type const & key ) const [inline]**

Find a <key, data> pair for a given key.

**Parameters**

*key* The key value to use for the lookup.

Definition at line 93 of file [avl\\_map](#).

**11.6.3.5 template<typename Key, typename Data, template< typename A > class Compare = Lt\_functor, template< typename B > class Alloc = New\_allocator> int cxx::Avl\_map< Key, Data, Compare, Alloc >::remove ( Key\_type const & key ) [inline]**

Remove the <key, data> pair for the given key.

#### Parameters

*key* The key value of the pair that shall be removed.

Definition at line 100 of file [avl\\_map](#).

**11.6.3.6 template<typename Key, typename Data, template< typename A > class Compare = Lt\_functor, template< typename B > class Alloc = New\_allocator> int cxx::Avl\_map< Key, Data, Compare, Alloc >::erase ( Key\_type const & key ) [inline]**

Removed the element *key*.

#### See also

[remove\(\)](#)

Definition at line 107 of file [avl\\_map](#).

**11.6.3.7 template<typename Key, typename Data, template< typename A > class Compare = Lt\_functor, template< typename B > class Alloc = New\_allocator> Data\_type const& cxx::Avl\_map< Key, Data, Compare, Alloc >::operator[] ( Key\_type const & key ) const [inline]**

Get the data for the given key.

#### Parameters

*key* The key value to use for lookup.

#### Precondition

A <key, data> pair for the given key value must exist.

Definition at line 115 of file [avl\\_map](#).

**11.6.3.8 template<typename Key, typename Data, template< typename A > class Compare = Lt\_functor, template< typename B > class Alloc = New\_allocator> Data\_type& cxx::Avl\_map< Key, Data, Compare, Alloc >::operator[] ( Key\_type const & key ) [inline]**

Get the data for the given key.

#### Parameters

*key* The key value to use for lookup.

**Precondition**

A <key, data> pair for the given key value must exist.

Definition at line 123 of file [avl\\_map](#).

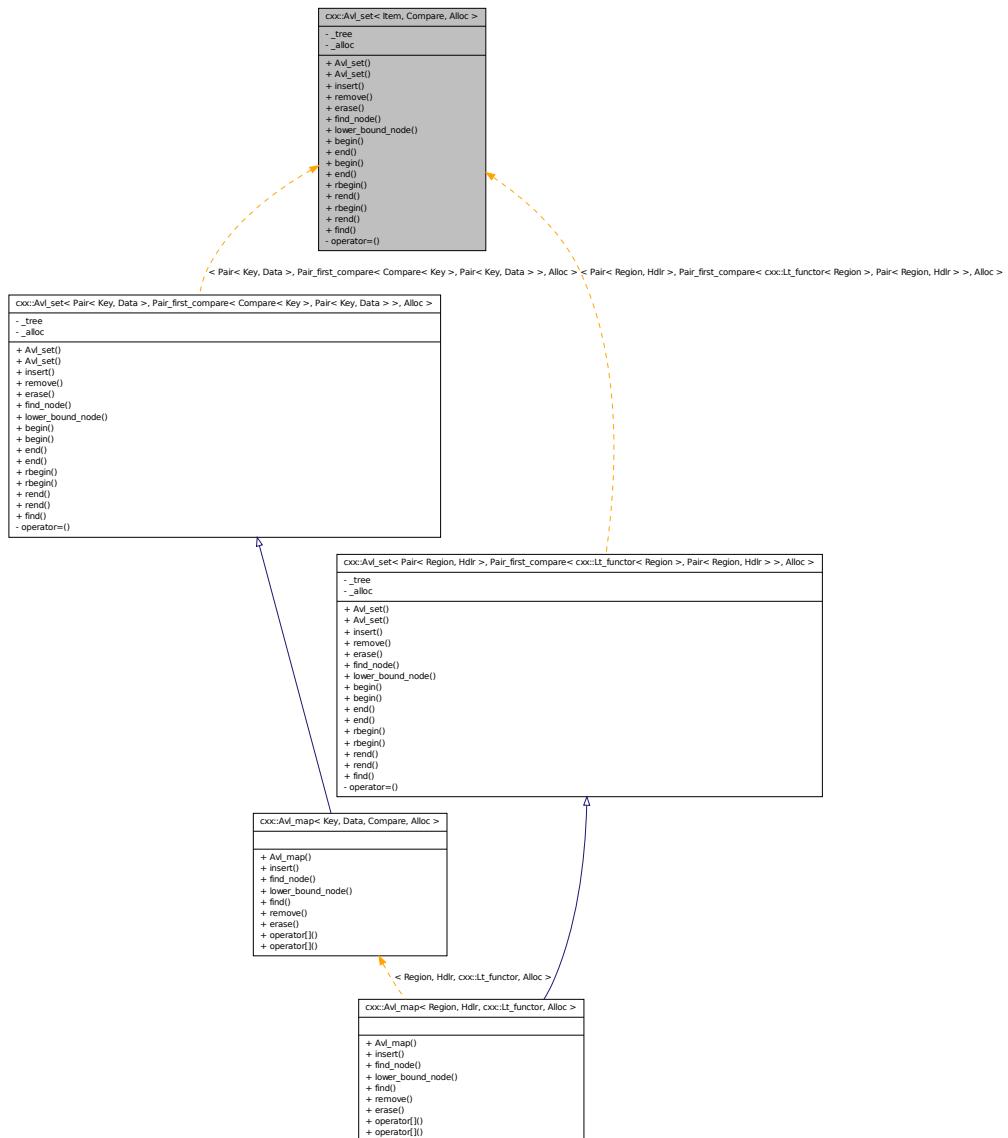
The documentation for this class was generated from the following file:

- [l4/cxx/avl\\_map](#)

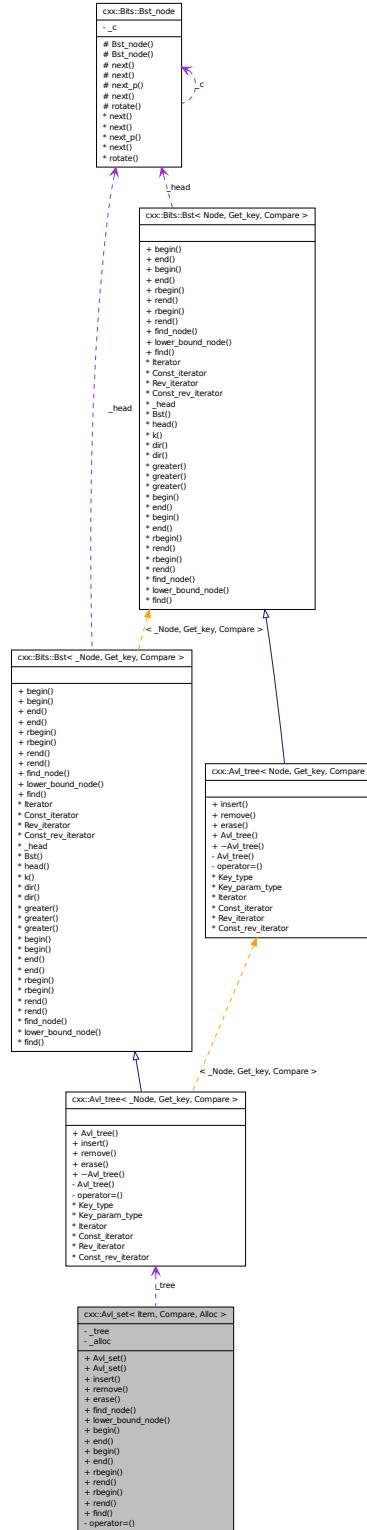
## **11.7 cxx::Avl\_set< Item, Compare, Alloc > Class Template Reference**

AVL Tree for simple comapreable items.

Inheritance diagram for cxx::Avl\_set< Item, Compare, Alloc >:



Collaboration diagram for cxx::Avl\_set< Item, Compare, Alloc >:



## Data Structures

- class [Node](#)  
*A smart pointer to a tree item.*

## Public Types

- [typedef Item Item\\_type](#)  
*Type for the items contained in the tree.*
- [typedef Compare Item\\_compare](#)  
*Type for the comparison functor.*
- [typedef Alloc< \\_Node > Node\\_allocator](#)  
*Type for the node allocator.*
- [typedef \\_\\_Avl\\_set\\_iter< \\_Node, Item\\_type, Fwd > Iterator](#)  
*Forward iterator for the set.*
- [typedef \\_\\_Avl\\_set\\_iter< \\_Node, Const\\_item\\_type, Fwd > Const\\_iterator](#)  
*Constant forward iterator for the set.*
- [typedef \\_\\_Avl\\_set\\_iter< \\_Node, Item\\_type, Rev > Rev\\_iterator](#)  
*Backward iterator for the set.*
- [typedef \\_\\_Avl\\_set\\_iter< \\_Node, Const\\_item\\_type, Rev > Const\\_rev\\_iterator](#)  
*Constant backward iterator for the set.*

## Public Member Functions

- [Avl\\_set \(Node\\_allocator const &alloc=Node\\_allocator\(\)\)](#)  
*Create a AVL-tree based set.*
- [Avl\\_set \(Avl\\_set const &o\)](#)  
*Create a copy of an AVL-tree based set.*
- [cxx::Pair< Iterator, int > insert \(Item\\_type const &item\)](#)  
*Insert an item into the set.*
- [int remove \(Item\\_type const &item\)](#)  
*Remove an item from the set.*
- [Node find\\_node \(Item\\_type const &item\) const](#)  
*Lookup a node equal to item.*
- [Node lower\\_bound\\_node \(Item\\_type const &key\) const](#)  
*Find the first node greater or equal to key.*

- **Const\_iterator begin () const**  
*Get the constant forward iterator for the first element in the set.*
- **Const\_iterator end () const**  
*Get the end marker for the constant forward iterator.*
- **Iterator begin ()**  
*Get the mutable forward iterator for the first element of the set.*
- **Iterator end ()**  
*Get the end marker for the mutable forward iterator.*
- **Const\_rev\_iterator rbegin () const**  
*Get the constant backward iterator for the last element in the set.*
- **Const\_rev\_iterator rend () const**  
*Get the end marker for the constant backward iterator.*
- **Rev\_iterator rbegin ()**  
*Get the mutable backward iterator for the last element of the set.*
- **Rev\_iterator rend ()**  
*Get the end marker for the mutable backward iterator.*

### 11.7.1 Detailed Description

**template<typename Item, class Compare = Lt\_functor<Item>, template< typename A > class Alloc = New\_allocator> class cxx::Avl\_set< Item, Compare, Alloc >**

AVL Tree for simple comapreable items. The AVL tree can store any kind of items where a partial order is defined. The default relation is defined by the '<' operator.

#### Parameters

- Item** The type of the items to be stored in the tree.  
**Compare** The relation to define the partial order, default is to use operator '<'.  
**Alloc** The allocator to use for the nodes of the AVL tree.

Definition at line 106 of file [avl\\_set](#).

### 11.7.2 Constructor & Destructor Documentation

**11.7.2.1 template<typename Item, class Compare = Lt\_functor<Item>, template< typename A > class Alloc = New\_allocator> cxx::Avl\_set< Item, Compare, Alloc >::Avl\_set ( Node\_allocator const & alloc = Node\_allocator () ) [inline, explicit]**

Create a AVL-tree based set.

**Parameters**

*comp* Comparison functor.

*alloc* Node allocator.

Create an empty set (AVL-tree based).

Definition at line 215 of file [avl\\_set](#).

**11.7.2.2 template<typename Item , class Compare , template< typename A > class Alloc>**  
**cxx::Avl\_set< Item, Compare, Alloc >::Avl\_set ( Avl\_set< Item, Compare, Alloc > const**  
**& o ) [inline]**

Create a copy of an AVL-tree based set.

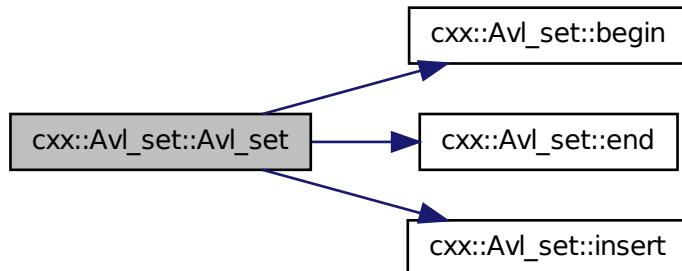
**Parameters**

*o* The set to copy.

Definition at line 335 of file [avl\\_set](#).

References [cxx::Avl\\_set< Item, Compare, Alloc >::begin\(\)](#), [cxx::Avl\\_set< Item, Compare, Alloc >::end\(\)](#), and [cxx::Avl\\_set< Item, Compare, Alloc >::insert\(\)](#).

Here is the call graph for this function:

**11.7.3 Member Function Documentation**

**11.7.3.1 template<typename Item, class Compare = Lt\_functor<Item>, template< typename A > class Alloc = New\_allocator> cxx::Pair<Iterator, int> cxx::Avl\_set< Item, Compare, Alloc >::insert ( Item\_type const & item )**

Insert an item into the set.

**Parameters**

*item* The item to insert.

**Returns**

0 on success, -1 on out of memory, and -2 if the element already exists in the set.

Insert a new item into the set, each item can only be once in the set.

Referenced by [cxx::Avl\\_set< Item, Compare, Alloc >::Avl\\_set\(\)](#).

Here is the caller graph for this function:



**11.7.3.2 template<typename Item, class Compare = Lt\_functor<Item>, template< typename A > class Alloc = New\_allocator> int cxx::Avl\_set< Item, Compare, Alloc >::remove ( Item\_type const & item ) [inline]**

Remove an item from the set.

**Parameters**

*item* The item to remove.

**Returns**

0 on success, -3 if the item does not exist, and -4 on internal error.

Definition at line [242](#) of file [avl\\_set](#).

**11.7.3.3 template<typename Item, class Compare = Lt\_functor<Item>, template< typename A > class Alloc = New\_allocator> Node cxx::Avl\_set< Item, Compare, Alloc >::find\_node ( Item\_type const & item ) const [inline]**

Lookup a node equal to *item*.

**Parameters**

*item* The value to search for.

**Returns**

A smart pointer to the element found, if no element found the pointer is NULL.

Definition at line [267](#) of file [avl\\_set](#).

**11.7.3.4 template<typename Item, class Compare = Lt\_functor<Item>, template< typename A > class Alloc = New\_allocator> Node cxx::Avl\_set< Item, Compare, Alloc >::lower\_bound\_node ( Item\_type const & key ) const [inline]**

Find the first node greater or equal to *key*.

#### Parameters

*key* the key to look for.

#### Returns

The first node greater or equal to *key*.

Definition at line 275 of file [avl\\_set](#).

**11.7.3.5 template<typename Item, class Compare = Lt\_functor<Item>, template< typename A > class Alloc = New\_allocator> Const\_iterator cxx::Avl\_set< Item, Compare, Alloc >::begin ( ) const [inline]**

Get the constant forward iterator for the first element in the set.

#### Returns

Constant forward iterator for the first element in the set.

Definition at line 283 of file [avl\\_set](#).

Referenced by [cxx::Avl\\_set< Item, Compare, Alloc >::Avl\\_set\(\)](#).

Here is the caller graph for this function:



**11.7.3.6 template<typename Item, class Compare = Lt\_functor<Item>, template< typename A > class Alloc = New\_allocator> Const\_iterator cxx::Avl\_set< Item, Compare, Alloc >::end ( ) const [inline]**

Get the end marker for the constant forward iterator.

#### Returns

The end marker for the constant forward iterator.

Definition at line 288 of file [avl\\_set](#).

Referenced by [cxx::Avl\\_set< Item, Compare, Alloc >::Avl\\_set\(\)](#).

Here is the caller graph for this function:



**11.7.3.7 template<typename Item, class Compare = Lt\_functor<Item>, template< typename A > class Alloc = New\_allocator> Iterator cxx::Avl\_set< Item, Compare, Alloc >::begin ( ) [inline]**

Get the mutable forward iterator for the first element of the set.

#### Returns

The mutable forward iterator for the first element of the set.

Definition at line 294 of file [avl\\_set](#).

**11.7.3.8 template<typename Item, class Compare = Lt\_functor<Item>, template< typename A > class Alloc = New\_allocator> Iterator cxx::Avl\_set< Item, Compare, Alloc >::end ( ) [inline]**

Get the end marker for the mutable forward iterator.

#### Returns

The end marker for mutable forward iterator.

Definition at line 299 of file [avl\\_set](#).

**11.7.3.9 template<typename Item, class Compare = Lt\_functor<Item>, template< typename A > class Alloc = New\_allocator> Const\_rev\_iterator cxx::Avl\_set< Item, Compare, Alloc >::rbegin ( ) const [inline]**

Get the constant backward iterator for the last element in the set.

#### Returns

The constant backward iterator for the last element in the set.

Definition at line 305 of file [avl\\_set](#).

**11.7.3.10 template<typename Item, class Compare = Lt\_functor<Item>, template< typename A > class Alloc = New\_allocator> Const\_rev\_iterator cxx::Avl\_set< Item, Compare, Alloc >::rend ( ) const [inline]**

Get the end marker for the constant backward iterator.

#### Returns

The end marker for the constant backward iterator.

Definition at line 310 of file [avl\\_set](#).

**11.7.3.11 template<typename Item, class Compare = Lt\_functor<Item>, template< typename A > class Alloc = New\_allocator> Rev\_iterator cxx::Avl\_set< Item, Compare, Alloc >::rbegin ( ) [inline]**

Get the mutable backward iterator for the last element of the set.

#### Returns

The mutable backward iterator for the last element of the set.

Definition at line 316 of file [avl\\_set](#).

**11.7.3.12 template<typename Item, class Compare = Lt\_functor<Item>, template< typename A > class Alloc = New\_allocator> Rev\_iterator cxx::Avl\_set< Item, Compare, Alloc >::rend ( ) [inline]**

Get the end marker for the mutable backward iterator.

#### Returns

The end marker for mutable backward iterator.

Definition at line 321 of file [avl\\_set](#).

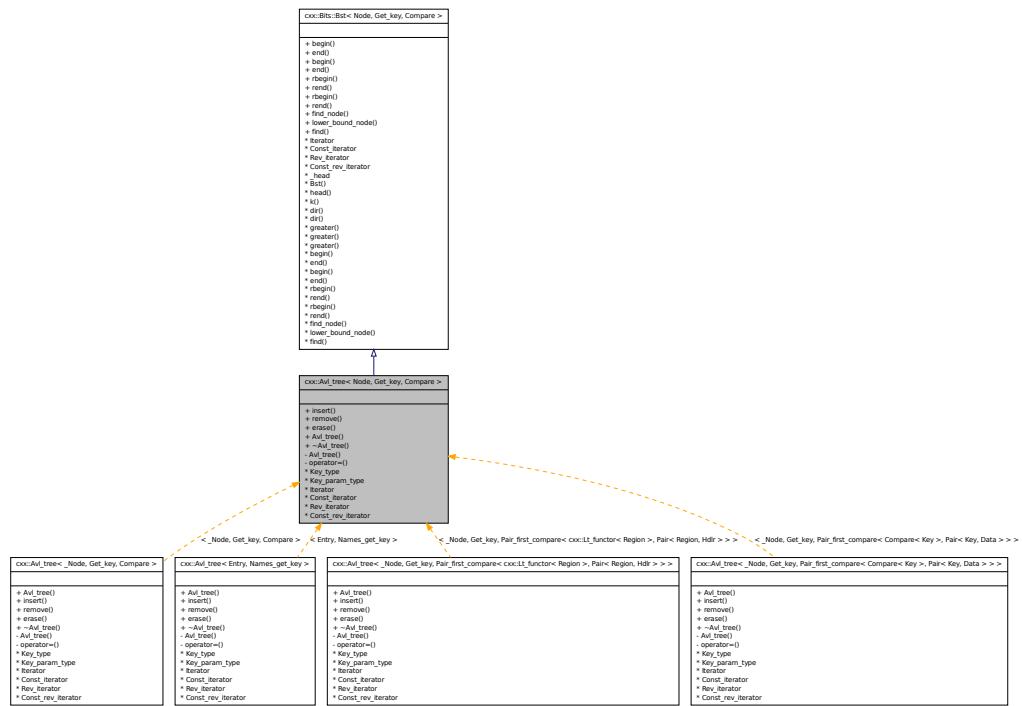
The documentation for this class was generated from the following file:

- 14/cxx/avl\_set

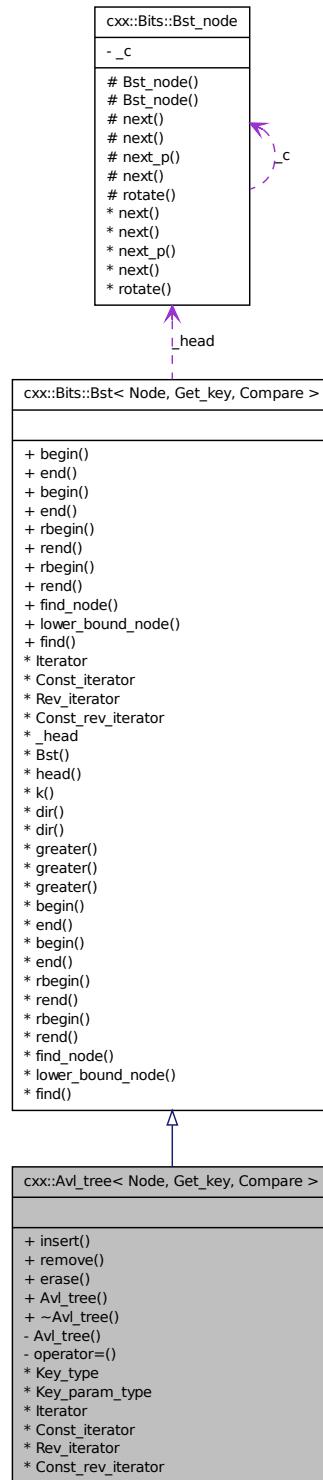
## 11.8 cxx::Avl\_tree< Node, Get\_key, Compare > Class Template Reference

A generic AVL tree.

Inheritance diagram for cxx::Avl\_tree< Node, Get\_key, Compare >:



Collaboration diagram for cxx::Avl\_tree< Node, Get\_key, Compare >:



## Public Types

- **typedef Bst::Key\_type Key\_type**  
*The type of key values used to generate the total order of the elements.*
- **typedef Bst::Key\_param\_type Key\_param\_type**  
*The type for key parameters.*
  
- **typedef Bst::Iterator Iterator**  
*Grab iterator types from Bst.*
- **typedef Bst::Const\_iterator Const\_iterator**  
*Constant forward iterator for the set.*
- **typedef Bst::Rev\_iterator Rev\_iterator**  
*Backward iterator for the set.*
- **typedef Bst::Const\_rev\_iterator Const\_rev\_iterator**  
*Constant backward iterator for the set.*

## Public Member Functions

- **Pair< Node \*, bool > insert (Node \*new\_node)**  
*Insert a new node into this AVL tree.*
- **Node \* remove (Key\_param\_type key)**  
*Remove the node with key from the tree.*
- **Node \* erase (Key\_param\_type key)**  
*An alias for `remove()`.*
- **Avl\_tree ()**  
*Create an empty AVL tree.*
- **~Avl\_tree ()**  
*Destroy, and free the set.*

### 11.8.1 Detailed Description

```
template<typename Node, typename Get_key, typename Compare = Lt_functor<typename Get_-  
key::Key_type>> class cxx::Avl_tree< Node, Get_key, Compare >
```

A generic AVL tree.

#### Parameters

**Node** the data type of the nodes (must inherit from `Avl_tree_noed`).

**Get\_key** the meta function to get the key value from a node. The implementation uses Get\_key::key\_of(ptr\_to\_node).

**Compare** binary relation to establish a total order for the nodes of the tree. Compare()(l, r) must return true if the key *l* is smaller than the key *r*.

### Examples:

[tmpfs/lib/src/fs.cc](#).

Definition at line 102 of file [avl\\_tree](#).

## 11.8.2 Member Typedef Documentation

**11.8.2.1 template<typename Node, typename Get\_key, typename Compare = Lt\_functor<typename Get\_key::Key\_type>> typedef Bst::Iterator cxx::Avl\_tree< Node, Get\_key, Compare >::Iterator**

Grab iterator types from Bst.

Forward iterator for the set.

Reimplemented from [cxx::Bits::Bst< Node, Get\\_key, Compare >](#).

Definition at line 133 of file [avl\\_tree](#).

## 11.8.3 Member Function Documentation

**11.8.3.1 template<typename Node, typename Get\_key , class Compare > Pair< Node \*, bool > cxx::Avl\_tree< Node, Get\_key, Compare >::insert ( Node \* new\_node )**

Insert a new node into this AVL tree.

### Parameters

**new\_node** a pointer to the new node. This node must not already b in an AVL tree.

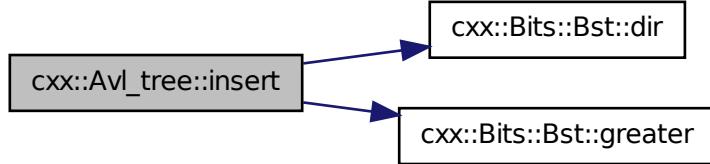
### Returns

A pair, with second set to 'true' and first pointing to *new\_node*, on success. If there is already a node with the same key that first point to this node and second is 'false'.

Definition at line 225 of file [avl\\_tree](#).

References [cxx::Bits::Bst< Node, Get\\_key, Compare >::\\_head](#), [cxx::Bits::Bst< Node, Get\\_key, Compare >::dir\(\)](#), [cxx::Bits::Bst< Node, Get\\_key, Compare >::greater\(\)](#), and [cxx::Bits::Direction::N](#).

Here is the call graph for this function:



### 11.8.3.2 template<typename Node , typename Get\_key , class Compare > Node \* cxx::Avl\_tree< Node, Get\_key, Compare >::remove ( Key\_param\_type key ) [inline]

Remove the node with *key* from the tree.

#### Parameters

*key* The node to remove.

#### Returns

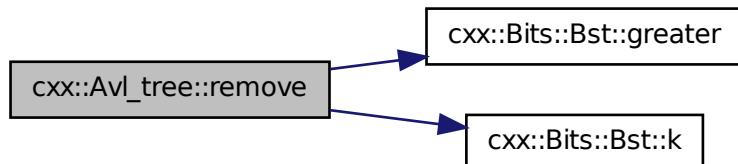
The pointer to the removed node on success, or NULL -3 if no node with the *key* exists.

Definition at line 287 of file [avl\\_tree](#).

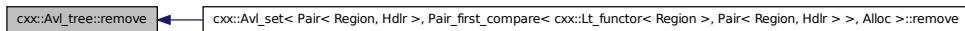
References [cxx::Bits::Bst< Node, Get\\_key, Compare >::\\_head](#), [cxx::Bits::Bst< Node, Get\\_key, Compare >::greater\(\)](#), [cxx::Bits::Bst< Node, Get\\_key, Compare >::k\(\)](#), [cxx::Bits::Direction::L](#), and [cxx::Bits::Direction::N](#).

Referenced by [cxx::Avl\\_set< Pair< Region, Hdlr >, Pair\\_first\\_compare< cxx::Lt\\_functor< Region >, Pair< Region, Hdlr > >, Alloc >::remove\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



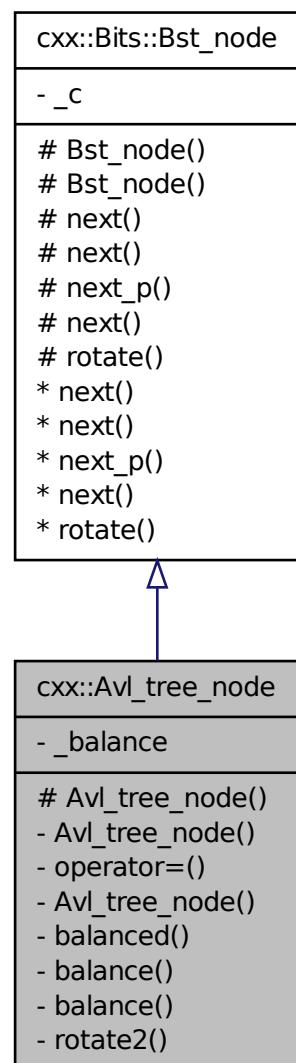
The documentation for this class was generated from the following file:

- l4/cxx/avl\_tree

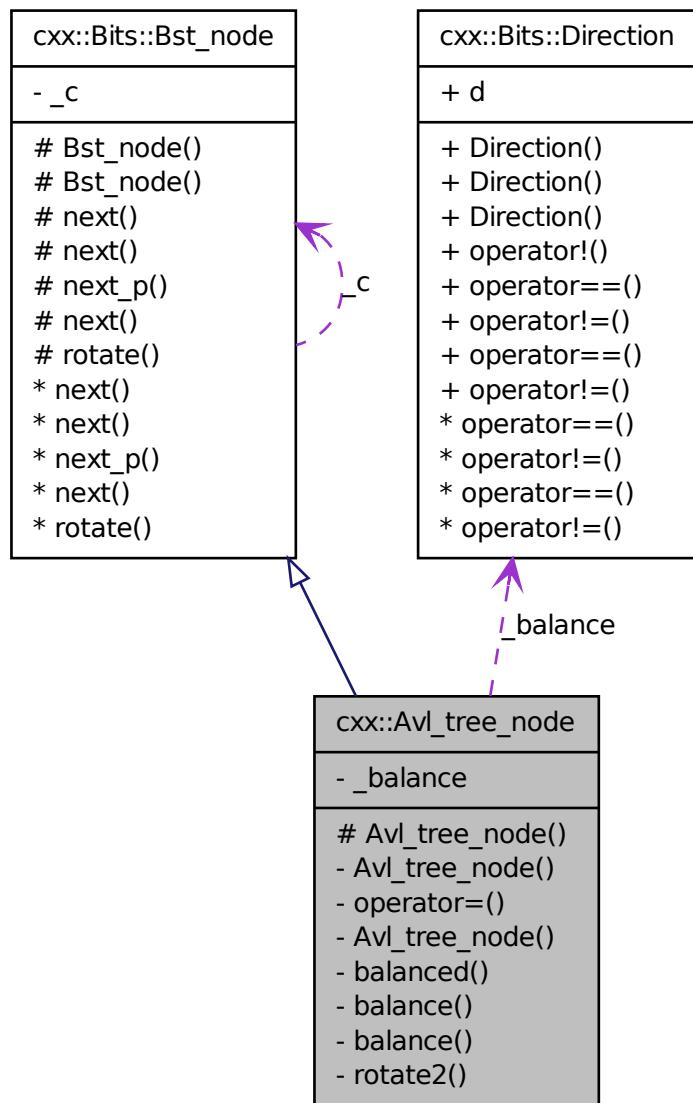
## 11.9 cxx::Avl\_tree\_node Class Reference

Node of an AVL tree.

Inheritance diagram for cxx::Avl\_tree\_node:



Collaboration diagram for cxx::Avl\_tree\_node:



## Protected Member Functions

- `Avl_tree_node()`

*Create an uninitialized node, this is what you shoulkd do.*

### 11.9.1 Detailed Description

Node of an AVL tree.

Definition at line 38 of file [avl\\_tree](#).

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

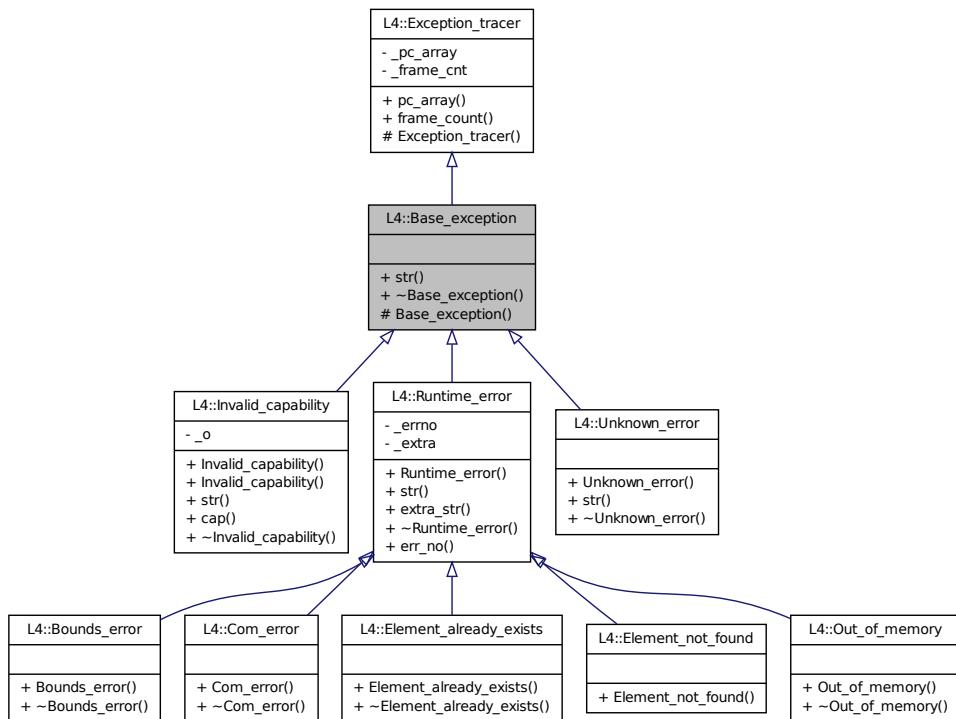
- l4/cxx/avl\_tree

## 11.10 L4::Base\_exception Class Reference

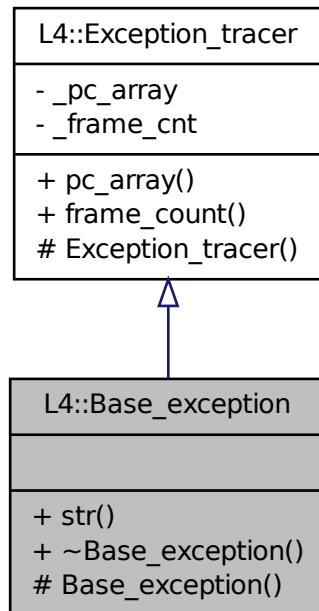
Base class for all exceptions, thrown by the [L4Re](#) framework.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Base\_exception:



Collaboration diagram for L4::Base\_exception:



## Public Member Functions

- virtual char const \* **str** () const =0 throw ()
 

*Should return a human readable string for the exception.*
- virtual **~Base\_exception** () throw ()
 

*Destruction.*

## Protected Member Functions

- **Base\_exception** () throw ()
 

*Create a base exception.*

### 11.10.1 Detailed Description

Base class for all exceptions, thrown by the [L4Re](#) framework. This is the abstract base of all exceptions thrown within the [L4Re](#) framework. It is basically also a good idea to use it as base of all user defined exceptions.

Definition at line 113 of file [exceptions](#).

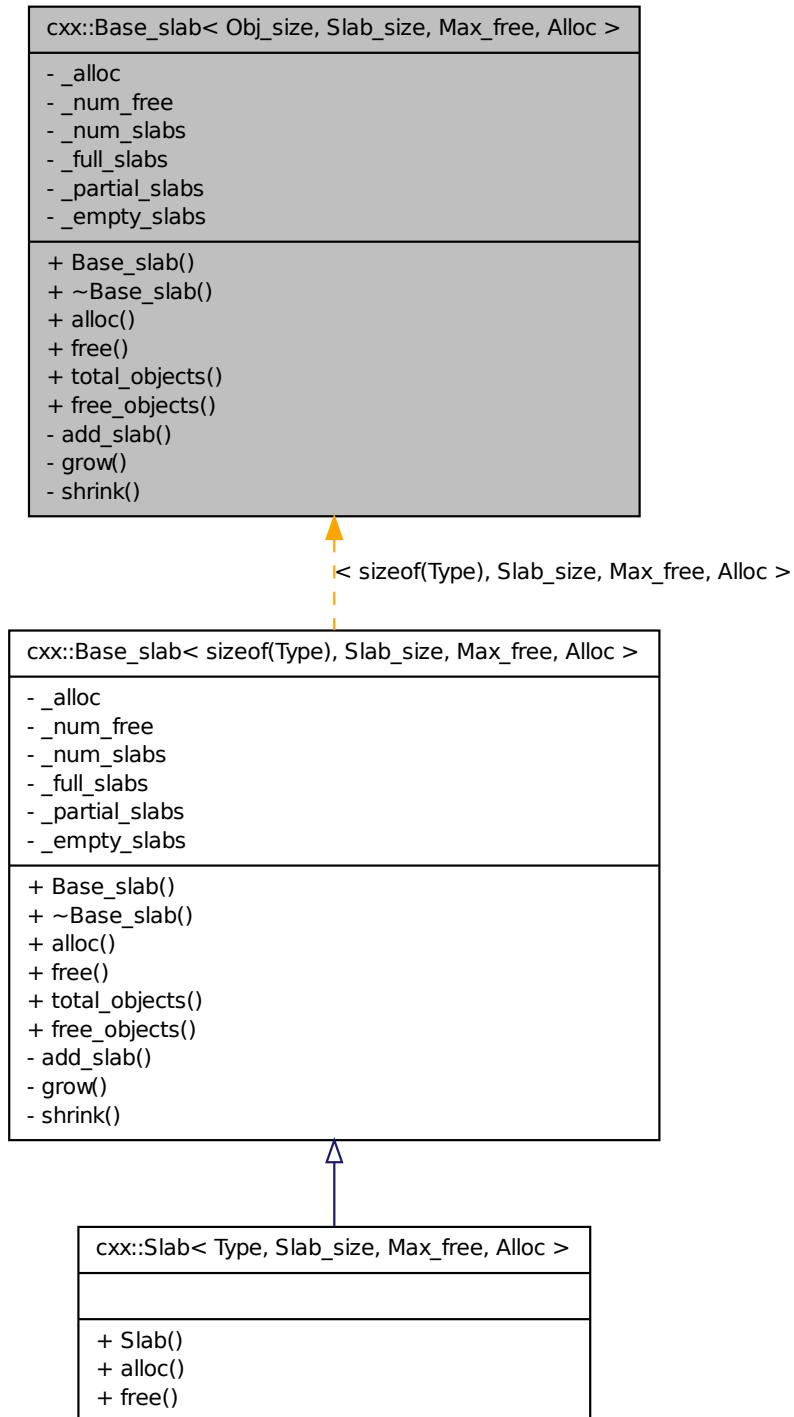
The documentation for this class was generated from the following file:

- l4/cxx/exceptions

## 11.11 **cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >** Class Template Reference

Basic slab allocator.

Inheritance diagram for cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >:



## Public Types

- enum { **object\_size** = Obj\_size, **slab\_size** = Slab\_size, **objects\_per\_slab** = (Slab\_size - sizeof(Slab\_head)) / object\_size, **max\_free\_slabs** = Max\_free }
- typedef Alloc< Slab\_i > **Slab\_alloc**

*Type of the allocator for the slab caches.*

## Public Member Functions

- unsigned **total\_objects** () const throw ()
 

*Get the total number of objects managed by the slab allocator.*
- unsigned **free\_objects** () const throw ()
 

*Get the total number of objects managed by the slab allocator.*

### 11.11.1 Detailed Description

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator> class cxx::Base_slab< Obj_size, Slab_size, Max_free, Alloc >
```

Basic slab allocator.

#### Parameters

- Obj\_size** The size of the objects managed by the allocator (in bytes).
- Slab\_size** The size of a slab cache (in bytes).
- Max\_free** The maximum number of free slab caches. When this limit is reached slab caches are freed.
- Alloc** The allocator that is used to allocate the slab caches.

Definition at line 38 of file [slab\\_alloc](#).

### 11.11.2 Member Enumeration Documentation

#### 11.11.2.1 template<int Obj\_size, int Slab\_size = L4\_PAGESIZE, int Max\_free = 2, template< typename A > class Alloc = New\_allocator> anonymous enum

Enumerator:

- object\_size** size of an object.
- slab\_size** size of a slab cache.
- objects\_per\_slab** objects per slab cache.
- max\_free\_slabs** maximum number of free slab caches.

Definition at line 47 of file [slab\\_alloc](#).

### 11.11.3 Member Function Documentation

**11.11.3.1 template<int Obj\_size, int Slab\_size = L4\_PAGESIZE, int Max\_free = 2, template< typename A > class Alloc = New\_allocator> unsigned cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >::total\_objects( ) const throw() [inline]**

Get the total number of objects managed by the slab allocator.

#### Returns

The number of objects managed by the allocator (including the free objects).

Definition at line 248 of file [slab\\_alloc](#).

**11.11.3.2 template<int Obj\_size, int Slab\_size = L4\_PAGESIZE, int Max\_free = 2, template< typename A > class Alloc = New\_allocator> unsigned cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >::free\_objects( ) const throw() [inline]**

Get the total number of objects managed by the slab allocator.

#### Returns

The number of objects managed by the allocator (including the free objects).

Definition at line 256 of file [slab\\_alloc](#).

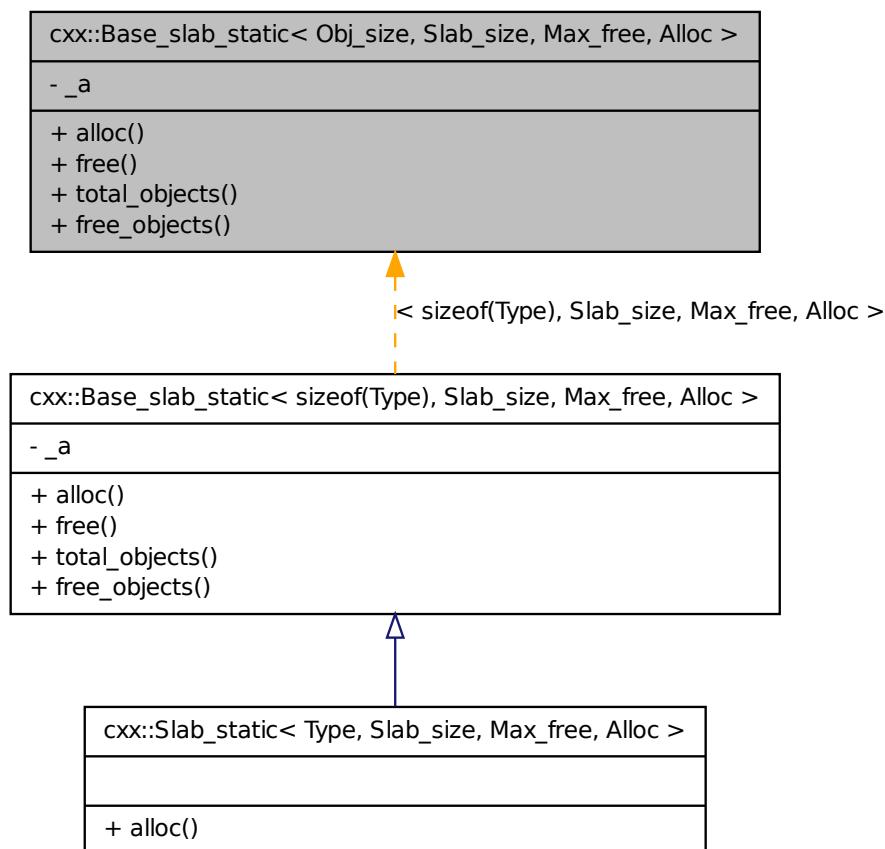
The documentation for this class was generated from the following file:

- l4/cxx/slab\_alloc

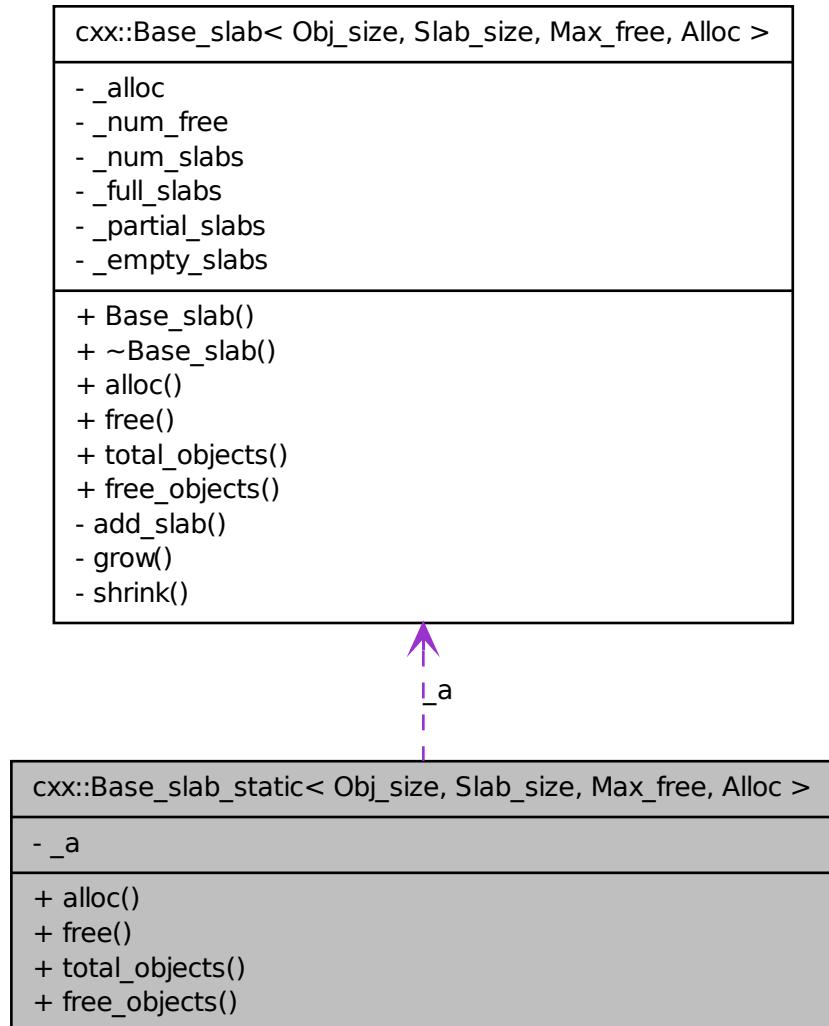
## 11.12 cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc > Class Template Reference

Merged slab allocator (allocators for objects of the same size are merged together).

Inheritance diagram for cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc >:



Collaboration diagram for cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc >:



## Public Types

- enum { **object\_size** = Obj\_size, **slab\_size** = Slab\_size, **objects\_per\_slab** = \_A::objects\_per\_slab, **max\_free\_slabs** = Max\_free }

## Public Member Functions

- **void \* alloc () throw ()**  
*Allocate an object.*

- void [free](#) (void \*p) throw ()
   
*Free the given object (p).*
- unsigned [total\\_objects](#) () const throw ()
   
*Get the total number of objects managed by the slab allocator.*
- unsigned [free\\_objects](#) () const throw ()
   
*Get the number of free objects in the slab allocator.*

### 11.12.1 Detailed Description

```
template<int Obj_size, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A >
class Alloc = New_allocator> class cxx::Base_slab_static< Obj_size, Slab_size, Max_free, Alloc >
```

Merged slab allocator (allocators for objects of the same size are merged together).

#### Parameters

*Obj\_size* The size of an object managed by the slab allocator.

*Slab\_size* The size of a slab cache.

*Max\_free* The maximum number of free slab caches.

*Alloc* The allocator for the slab caches.

This slab allocator class is useful for merging slab allocators with the same parameters (equal *Obj\_size*, *Slab\_size*, *Max\_free*, and *Alloc* parameters) together and share the overhead for the slab caches among all equal-sized objects.

Definition at line 350 of file [slab\\_alloc](#).

### 11.12.2 Member Enumeration Documentation

#### 11.12.2.1 template<int Obj\_size, int Slab\_size = L4\_PAGESIZE, int Max\_free = 2, template< typename A > class Alloc = New\_allocator> anonymous enum

Enumerator:

*object\_size* size of an object.

*slab\_size* size of a slab cache.

*objects\_per\_slab* number of objects per slab cache.

*max\_free\_slabs* maximum number of free slab caches.

Definition at line 357 of file [slab\\_alloc](#).

### 11.12.3 Member Function Documentation

#### 11.12.3.1 template<int Obj\_size, int Slab\_size = L4\_PAGESIZE, int Max\_free = 2, template< typename A > class Alloc = New\_allocator> void\* cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc >::alloc ( ) throw () [inline]

Allocate an object.

Reimplemented in [cxx::Slab\\_static< Type, Slab\\_size, Max\\_free, Alloc >](#).

Definition at line [367](#) of file [slab\\_alloc](#).

**11.12.3.2 template<int Obj\_size, int Slab\_size = L4\_PAGESIZE, int Max\_free = 2, template< typename A > class Alloc = New\_allocator> void cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc >::free ( void \* p ) throw () [inline]**

Free the given object (*p*).

#### Parameters

*p* The pointer to the object to free.

#### Precondition

*p* must be a pointer to an object allocated by this allocator.

Definition at line [373](#) of file [slab\\_alloc](#).

**11.12.3.3 template<int Obj\_size, int Slab\_size = L4\_PAGESIZE, int Max\_free = 2, template< typename A > class Alloc = New\_allocator> unsigned cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc >::total\_objects ( ) const throw () [inline]**

Get the total number of objects managed by the slab allocator.

#### Returns

The number of objects managed by the allocator (including the free objects).

#### Note

The value is the merged value for all equal parameterized [Base\\_slab\\_static](#) instances.

Definition at line [382](#) of file [slab\\_alloc](#).

**11.12.3.4 template<int Obj\_size, int Slab\_size = L4\_PAGESIZE, int Max\_free = 2, template< typename A > class Alloc = New\_allocator> unsigned cxx::Base\_slab\_static< Obj\_size, Slab\_size, Max\_free, Alloc >::free\_objects ( ) const throw () [inline]**

Get the number of free objects in the slab allocator.

#### Returns

The number of free objects in all free and partially used slab caches managed by this allocator.

#### Note

The value is the merged value for all equal parameterized [Base\\_slab\\_static](#) instances.

Definition at line [391](#) of file [slab\\_alloc](#).

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

- l4/cxx/slab\_alloc

## 11.13 L4::Basic\_registry Class Reference

This registry returns the corresponding server object based on the label.

Inherited by L4Re::Util::Object\_registry.

### Static Public Member Functions

- static int [dispatch \(l4\\_umword\\_t obj, L4::Ipc::Iostream &ios\)](#)

*The dispatch function called by the server loop.*

### 11.13.1 Detailed Description

This registry returns the corresponding server object based on the label.

Definition at line [312](#) of file [ipc\\_server](#).

### 11.13.2 Member Function Documentation

#### 11.13.2.1 static int L4::Basic\_registry::dispatch ( l4\_umword\_t obj, L4::Ipc::Iostream & ios ) [inline, static]

The dispatch function called by the server loop.

##### Parameters

*p* The sender [L4](#) UID.

*ios* The [Ipc::Iostream](#) for the request and reply.

Definition at line [324](#) of file [ipc\\_server](#).

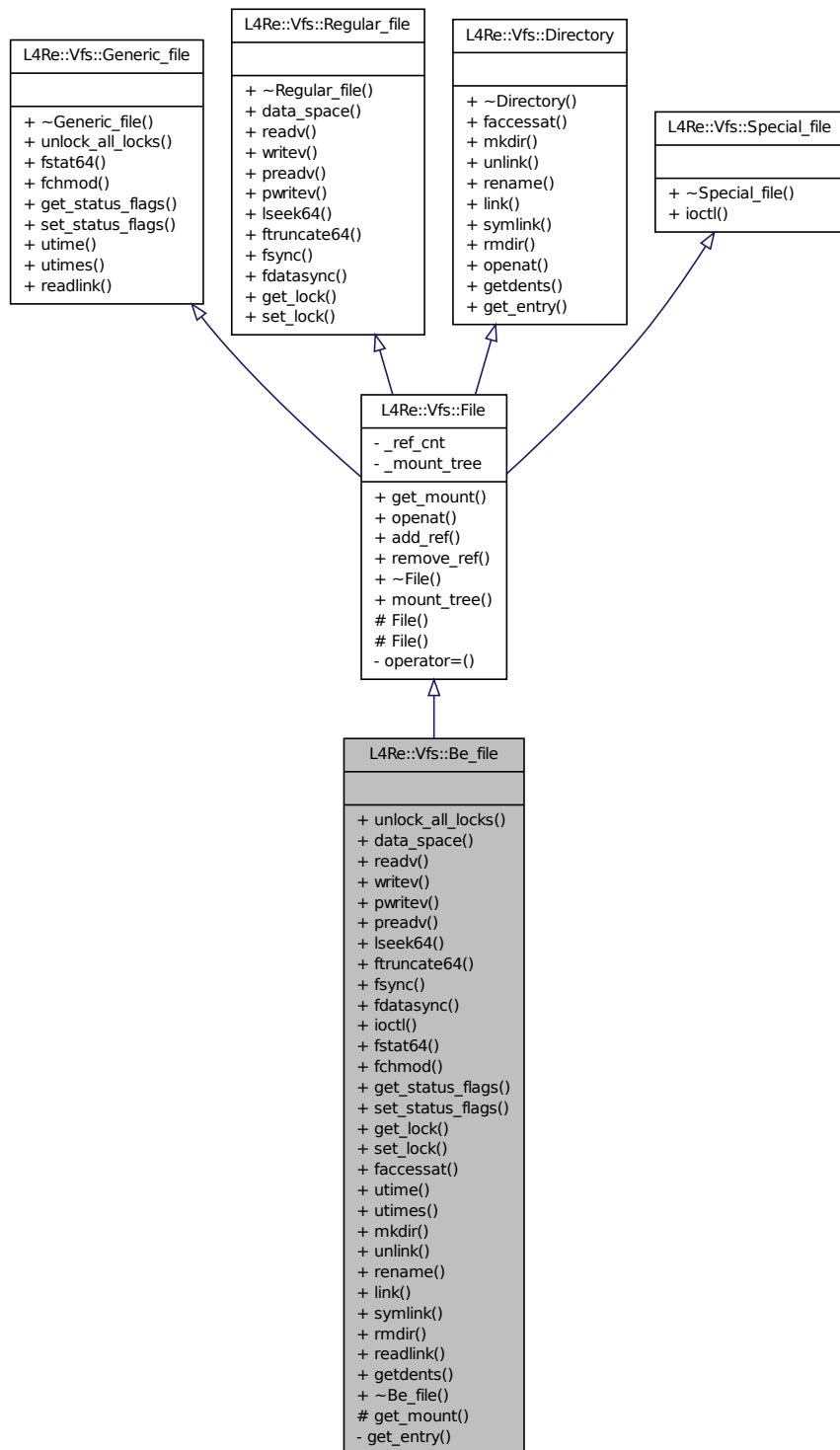
The documentation for this class was generated from the following file:

- [l4/cxx/ipc\\_server](#)

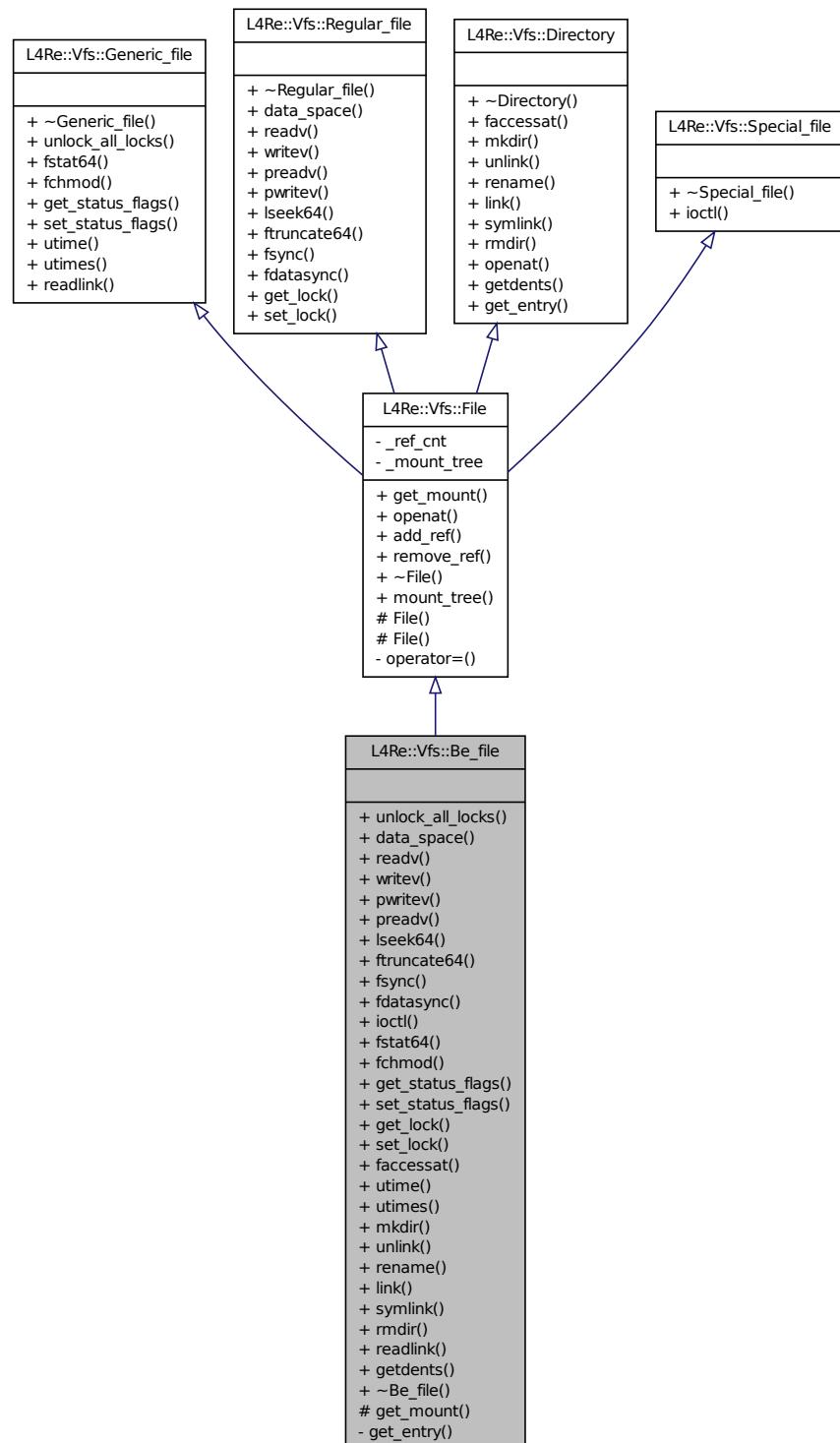
## 11.14 L4Re::Vfs::Be\_file Class Reference

Boiler plate class for implementing an open file for [L4Re::Vfs](#).

Inheritance diagram for L4Re::Vfs::Be\_file:



Collaboration diagram for L4Re::Vfs::Be\_file:



## Public Member Functions

- `ssize_t readyv (const struct iovec *, int) throw ()`  
*Default backend for POSIX read and ready functions.*
- `ssize_t writev (const struct iovec *, int) throw ()`  
*Default backend for POSIX write and writev functions.*
- `ssize_t pwritev (const struct iovec *, int, off64_t) throw ()`  
*Default backend for POSIX pwrite and pwritev functions.*
- `ssize_t preadv (const struct iovec *, int, off64_t) throw ()`  
*Default backend for POSIX pread and preadv functions.*
- `off64_t lseek64 (off64_t, int) throw ()`  
*Default backend for POSIX seek and lseek functions.*
- `int ftruncate64 (off64_t) throw ()`  
*Default backend for the POSIX truncate, ftruncate and similar functions.*
- `int fsync () const throw ()`  
*Default backend for POSIX fsync.*
- `int fdatasync () const throw ()`  
*Default backend for POSIX fdatasync.*
- `int ioctl (unsigned long, va_list) throw ()`  
*Default backend for POSIX ioctl.*
- `int fchmod (mode_t) throw ()`  
*Default backend for POSIX chmod and fchmod.*
- `int get_status_flags () const throw ()`  
*Default backend for POSIX fcntl subfunctions.*
- `int set_status_flags (long) throw ()`  
*Default backend for POSIX fcntl subfunctions.*
- `int get_lock (struct flock64 *) throw ()`  
*Default backend for POSIX fcntl subfunctions.*
- `int set_lock (struct flock64 *, bool) throw ()`  
*Default backend for POSIX fcntl subfunctions.*
- `int faccessat (const char *, int, int) throw ()`  
*Default backend for POSIX access and faccessat functions.*
- `int utime (const struct utimbuf *) throw ()`  
*Default backend for POSIX utime.*

- int [utimes](#) (const struct utimbuf[2]) throw ()
 

*Default backend for POSIX utimes.*
- int [mkdir](#) (const char \*, mode\_t) throw ()
 

*Default backend for POSIX mkdir and mkdirat.*
- int [unlink](#) (const char \*) throw ()
 

*Default backend for POSIX unlink, unlinkat.*
- int [rename](#) (const char \*, const char \*) throw ()
 

*Default backend for POSIX rename, renameat.*
- int [link](#) (const char \*, const char \*) throw ()
 

*Default backend for POSIX link, linkat.*
- int [symlink](#) (const char \*, const char \*) throw ()
 

*Default backend for POSIX symlink, symlinkat.*
- int [rmdir](#) (const char \*) throw ()
 

*Default backend for POSIX rmdir, rmdirat.*
- ssize\_t [readlink](#) (char \*, size\_t)
 

*Default backend for POSIX readlink, readlinkat.*

### 11.14.1 Detailed Description

Boiler plate class for implementing an open file for [L4Re::Vfs](#). This class may be used as a base class for everything that a POSIX file descriptor may point to. This are things such as regular files, directories, special device files, streams, pipes, and so on.

#### Examples:

[tmpfs/lib/src/fs.cc](#).

Definition at line 38 of file [backend](#).

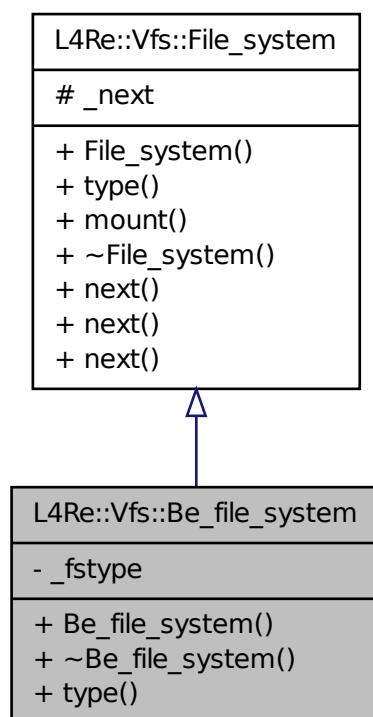
The documentation for this class was generated from the following file:

- l4/l4re\_vfs/backend

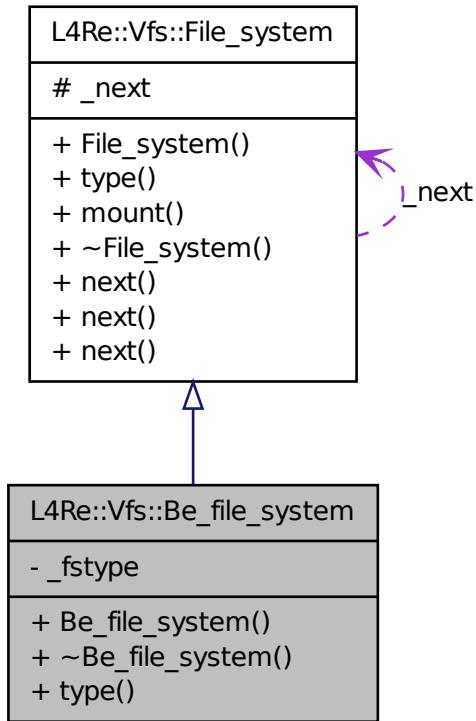
## 11.15 L4Re::Vfs::Be\_file\_system Class Reference

Boilerplate class for implementing a [L4Re::Vfs::File\\_system](#).

Inheritance diagram for L4Re::Vfs::Be\_file\_system:



Collaboration diagram for L4Re::Vfs::Be\_file\_system:



## Public Member Functions

- **Be\_file\_system** (char const \*fstype) throw ()

*Create a file-system object for the given fstype.*

- **~Be\_file\_system** () throw ()

*Destroy a file-system object.*

- char const \* **type** () const throw ()

*Return the file-system type.*

### 11.15.1 Detailed Description

Boilerplate class for implementing a L4Re::Vfs::File\_system. This class already takes care of registering and unregistering the file system in the global registry and implements the **type()** method.

**Examples:**

[tmpfs/lib/src/fs.cc](#).

Definition at line [233](#) of file [backend](#).

## 11.15.2 Constructor & Destructor Documentation

### 11.15.2.1 L4Re::Vfs::Be\_file\_system::Be\_file\_system ( `char const * fstype` ) throw () [inline, explicit]

Create a file-system object for the given *fstype*.

**Parameters**

*fstype* The type that [type\(\)](#) shall return.

This constructor takes care of registering the file system in the registry of L4Re::Vfs::vfs\_ops.

**Examples:**

[tmpfs/lib/src/fs.cc](#).

Definition at line [247](#) of file [backend](#).

### 11.15.2.2 L4Re::Vfs::Be\_file\_system::~Be\_file\_system ( ) throw () [inline]

Destroy a file-system object.

This destructor takes care of removing this file system from the registry of L4Re::Vfs::vfs\_ops.

Definition at line [259](#) of file [backend](#).

## 11.15.3 Member Function Documentation

### 11.15.3.1 char const\* L4Re::Vfs::Be\_file\_system::type ( ) const throw () [inline, virtual]

Return the file-system type.

Returns the file-system type given as *fstype* in the constructor.

Implements [L4Re::Vfs::File\\_system](#).

Definition at line [269](#) of file [backend](#).

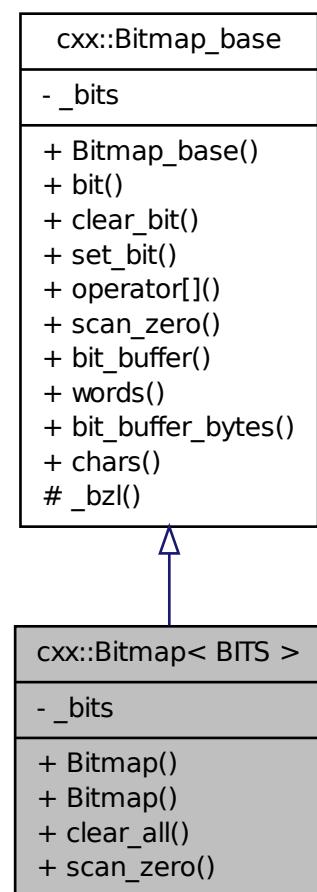
The documentation for this class was generated from the following file:

- [l4/l4re\\_vfs/backend](#)

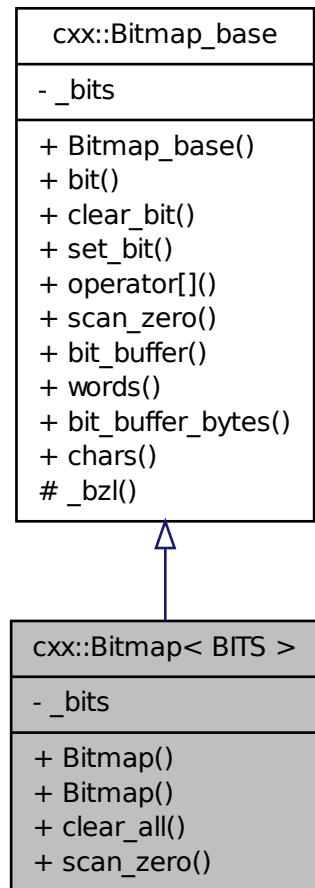
## 11.16 cxx::Bitmap< BITS > Class Template Reference

A static bit map.

Inheritance diagram for cxx::Bitmap< BITS >:



Collaboration diagram for cxx::Bitmap< BITS >:



## Public Member Functions

- [Bitmap \(\) throw \(\)](#)

*Create a bitmap with BITS bits.*

- [void clear\\_all \(\)](#)

*Scan for the first zero bit.*

### 11.16.1 Detailed Description

**template<int BITS> class cxx::Bitmap< BITS >**

A static bit map.

#### Parameters

*BITS* the number of bits that shall be in the bitmap.

Definition at line 120 of file [bitmap](#).

### 11.16.2 Constructor & Destructor Documentation

**11.16.2.1 template<int BITS> cxx::Bitmap< BITS >::Bitmap( ) throw() [inline]**

Create a bitmap with *BITS* bits.

Definition at line 127 of file [bitmap](#).

### 11.16.3 Member Function Documentation

**11.16.3.1 template<int BITS> void cxx::Bitmap< BITS >::clear\_all( ) [inline]**

Scan for the first zero bit.

#### Parameters

*start\_bit* the bit where the scanning shall begin.

Compared to [Bitmap\\_base::scan\\_zero\(\)](#), the upper bound is set to BITS.

Definition at line 137 of file [bitmap](#).

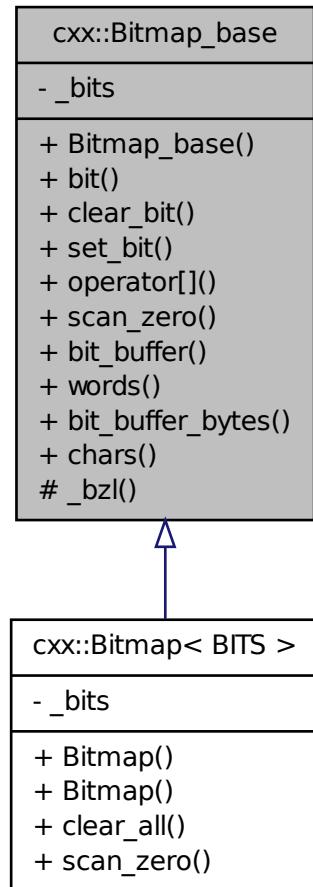
The documentation for this class was generated from the following file:

- 14/cxx(bitmap)

## 11.17 cxx::Bitmap\_base Class Reference

Basic bitmap abstraction.

Inheritance diagram for cxx::Bitmap\_base:



## Data Structures

- class [Char](#)  
*Helper abstraction for a byte contained in the bitmap.*
- class [Word](#)  
*Helper abstraction for a word contained in the bitmap.*

## Public Member Functions

- void [bit](#) (long bit, bool on) throw ()

*Set the value of bit bit to on.*

- void `clear_bit` (long bit) throw ()
 

*Clear bit bit.*
- void `set_bit` (long bit) throw ()
 

*Set bit bit.*
- unsigned long `operator[ ]` (long bit) const throw ()
 

*Get the truth value of a bit.*
- long `scan_zero` (long max\_bit, long start\_bit=0) const throw ()
 

*Scans for the first zero bit.*

## Static Public Member Functions

- static long `words` (long bits) throw ()
 

*Get the number of Words that are used for the bitmap.*
- static long `chars` (long bits) throw ()
 

*Get the number of chars that are used for the bitmap.*

### 11.17.1 Detailed Description

Basic bitmap abstraction. This abstraction keeps a pointer to a memory area that is used as bitmap.  
 Definition at line 30 of file [bitmap](#).

### 11.17.2 Member Function Documentation

#### 11.17.2.1 static long cxx::Bitmap\_base::words ( long bits ) throw () [inline, static]

Get the number of *Words* that are used for the bitmap.  
 Definition at line 44 of file [bitmap](#).

#### 11.17.2.2 static long cxx::Bitmap\_base::chars ( long bits ) throw () [inline, static]

Get the number of chars that are used for the bitmap.  
 Definition at line 61 of file [bitmap](#).

#### 11.17.2.3 void cxx::Bitmap\_base::bit ( long bit, bool on ) throw () [inline]

Set the value of bit *bit* to *on*.

##### Parameters

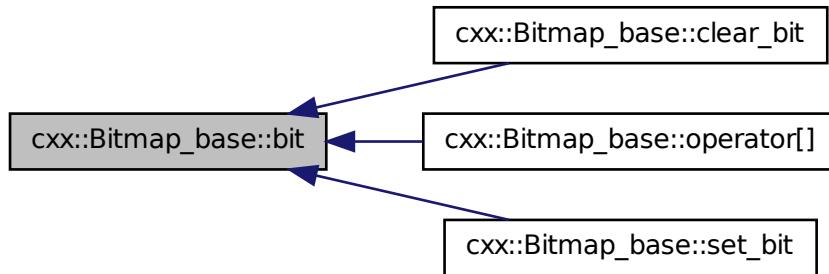
*bit* the number of the bit

*on* the boolean value that shall be assigned to the bit.

Definition at line 146 of file [bitmap](#).

Referenced by [clear\\_bit\(\)](#), [operator\[ \]\(\)](#), and [set\\_bit\(\)](#).

Here is the caller graph for this function:



#### 11.17.2.4 void cxx::Bitmap\_base::clear\_bit ( long *bit* ) throw () [inline]

Clear bit *bit*.

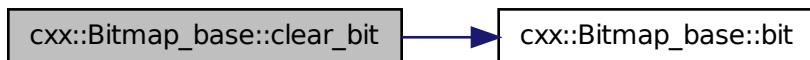
##### Parameters

*bit* the number of the bit to clear.

Definition at line 155 of file [bitmap](#).

References [bit\(\)](#).

Here is the call graph for this function:



#### 11.17.2.5 void cxx::Bitmap\_base::set\_bit ( long *bit* ) throw () [inline]

Set bit *bit*.

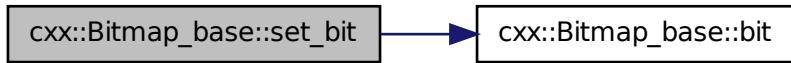
### Parameters

*bit* the number of the bit to set,

Definition at line 164 of file [bitmap](#).

References [bit\(\)](#).

Here is the call graph for this function:



### 11.17.2.6 `unsigned long cxx::Bitmap_base::operator[]( long bit ) const throw() [inline]`

Get the truth value of a bit.

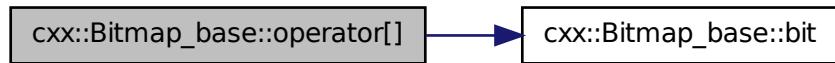
### Parameters

*bit* the number of the bit to read.

Definition at line 173 of file [bitmap](#).

References [bit\(\)](#).

Here is the call graph for this function:



### 11.17.2.7 `long cxx::Bitmap_base::scan_zero( long max_bit, long start_bit = 0 ) const throw() [inline]`

Scans for the first zero bit.

### Parameters

*max\_bit* the upper bound for the scanning operation.

*start\_bit* the number of the first bit to look at.

Definition at line 194 of file [bitmap](#).

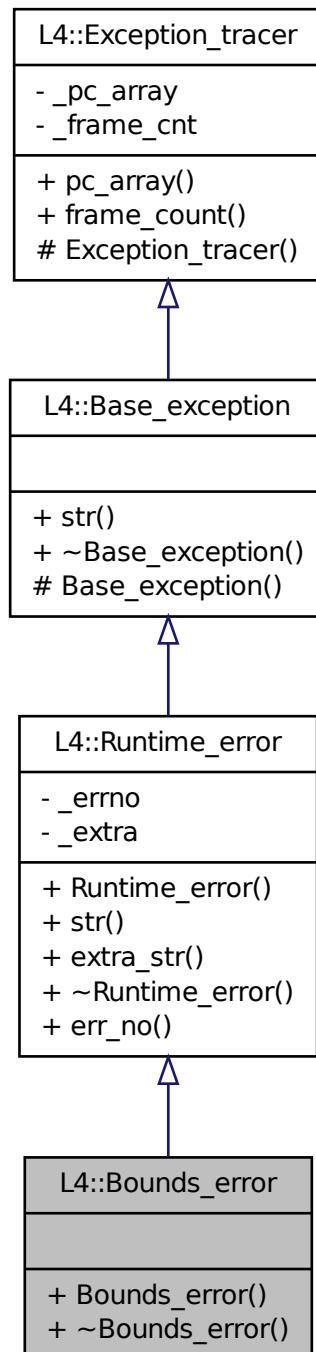
The documentation for this class was generated from the following file:

- l4/cxx(bitmap)

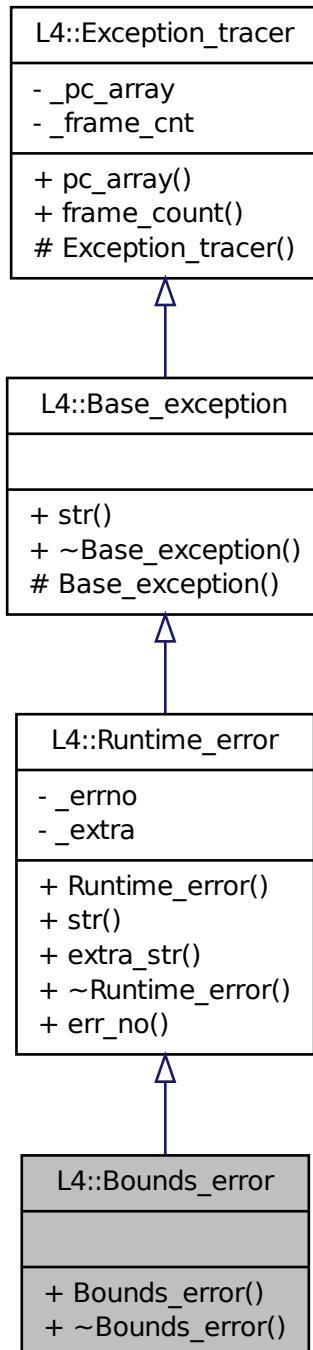
## 11.18 L4::Bounds\_error Class Reference

Access out of bounds.

Inheritance diagram for L4::Bounds\_error:



Collaboration diagram for L4::Bounds\_error:



### 11.18.1 Detailed Description

Access out of bounds.

Definition at line 264 of file [exceptions](#).

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

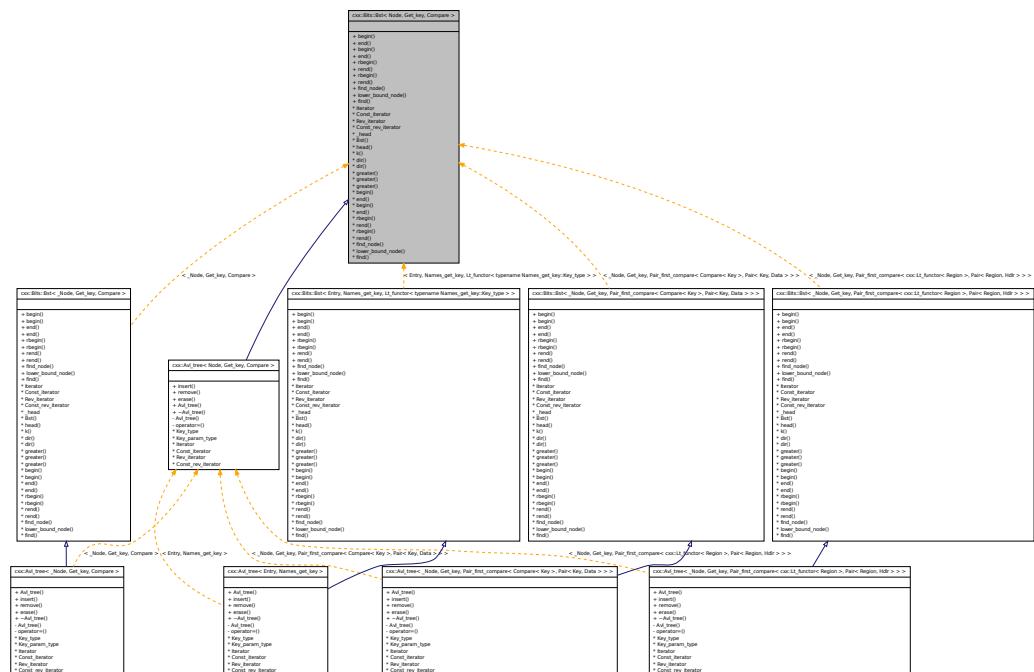
- 14/cxx/exceptions

## 11.19 cxx::Bits::Bst< Node, Get\_key, Compare > Class Template Reference

Basic binary search tree (BST).

```
#include <bst.h>
```

Inheritance diagram for cxx::Bits::Bst< Node, Get\_key, Compare >:



Collaboration diagram for cxx::Bits::Bst< Node, Get\_key, Compare >:



## Public Types

- **typedef Get\_key::Key\_type Key\_type**  
*The type of key values used to generate the total order of the elements.*
- **typedef Type\_traits< Key\_type >::Param\_type Key\_param\_type**  
*The type for key parameters.*
- **typedef Fwd Fwd\_iter\_ops**  
*Helper for building forward iterators for different wrapper classes.*
- **typedef Rev Rev\_iter\_ops**  
*Helper for building reverse iterators for different wrapper classes.*

## Iterators

- **typedef \_\_Bst\_iter< Node, Node, Fwd > Iterator**  
*Forward iterator.*
- **typedef \_\_Bst\_iter< Node, Node const, Fwd > Const\_iterator**  
*Constant forward iterator.*
- **typedef \_\_Bst\_iter< Node, Node, Rev > Rev\_iterator**  
*Backward iterator.*
- **typedef \_\_Bst\_iter< Node, Node const, Rev > Const\_rev\_iterator**  
*Constant backward.*

## Public Member Functions

### Get default iterators for the ordered tree.

- **Const\_iterator begin () const**  
*Get the constant forward iterator for the first element in the set.*
- **Const\_iterator end () const**  
*Get the end marker for the constant forward iterator.*
- **Iterator begin ()**  
*Get the mutable forward iterator for the first element of the set.*
- **Iterator end ()**  
*Get the end marker for the mutable forward iterator.*
- **Const\_rev\_iterator rbegin () const**  
*Get the constant backward iterator for the last element in the set.*
- **Const\_rev\_iterator rend () const**  
*Get the end marker for the constant backward iterator.*
- **Rev\_iterator rbegin ()**

*Get the mutable backward iterator for the last element of the set.*

- **Rev\_iterator rend ()**

*Get the end marker for the mutable backward iterator.*

### Lookup functions.

- **Node \* find\_node (Key\_param\_type key) const**  
*find the node with the given key.*
- **Node \* lower\_bound\_node (Key\_param\_type key) const**  
*find the first node with a key not less than the given key.*
- **Const\_iterator find (Key\_param\_type key) const**  
*find the node with the given key.*

### Interior access for descendants.

As this class is an intended base class we provide protected access to our interior, use 'using' to make this private in concrete implementations.

- **Bst\_node \* \_head**  
*The head pointer of the tree.*
- **Bst ()**  
*Create an empty tree.*
- **Node \* head () const**  
*Access the head node as object of type Node.*
- **static Key\_type k (Bst\_node const \*n)**  
*Get the key value of n.*
- **static Dir dir (Key\_param\_type l, Key\_param\_type r)**  
*Get the direction to go from l to search for r.*
- **static Dir dir (Key\_param\_type l, Bst\_node const \*r)**  
*Get the direction to go from l to search for r.*
- **static bool greater (Key\_param\_type l, Key\_param\_type r)**  
*Is l greater than r.*
- **static bool greater (Key\_param\_type l, Bst\_node const \*r)**  
*Is l greater than r.*
- **static bool greater (Bst\_node const \*l, Bst\_node const \*r)**  
*Is l greater than r.*

### 11.19.1 Detailed Description

```
template<typename Node, typename Get_key, typename Compare> class cxx::Bits::Bst< Node, Get_key, Compare >
```

Basic binary search tree (BST). This class is intended as a base class for concrete binary search trees, such as an AVL tree. This class already provides the basic lookup methods and iterator definitions for a BST.

Definition at line 40 of file [bst.h](#).

### 11.19.2 Member Function Documentation

```
11.19.2.1 template<typename Node, typename Get_key, typename Compare> static Dir
cxx::Bits::Bst< Node, Get_key, Compare >::dir( Key_param_type l, Key_param_type
r ) [inline, static, protected]
```

Get the direction to go from *l* to search for *r*.

#### Parameters

*l* is the key to look for.

*r* is the key at the current position.

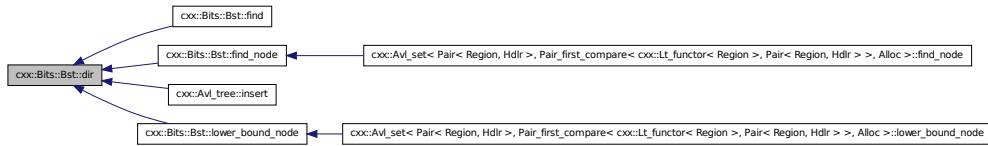
#### Returns

Direction::L for left, Direction::R for right, and Direction::N if *l* is equal to *r*.

Definition at line 117 of file [bst.h](#).

Referenced by [cxx::Bits::Bst< Node, Get\\_key, Compare >::find\(\)](#), [cxx::Bits::Bst< Node, Get\\_key, Compare >::find\\_node\(\)](#), [cxx::Avl\\_tree< Node, Get\\_key, Compare >::insert\(\)](#), and [cxx::Bits::Bst< Node, Get\\_key, Compare >::lower\\_bound\\_node\(\)](#).

Here is the caller graph for this function:



```
11.19.2.2 template<typename Node, typename Get_key, typename Compare> static Dir
cxx::Bits::Bst< Node, Get_key, Compare >::dir( Key_param_type l, Bst_node const *
r ) [inline, static, protected]
```

Get the direction to go from *l* to search for *r*.

#### Parameters

*l* is the key to look for.

*r* is the node at the current position.

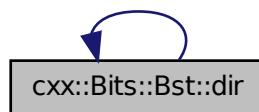
### Returns

Direction::L for left, Direction::R for right, and Direction::N if *l* is equal to *r*.

Definition at line 133 of file bst.h.

Referenced by [cxx::Bits::Bst< \\_Node, Get\\_key, Pair\\_first\\_compare< Compare< Key >, Pair< Key, Data > > >::dir\(\)](#).

Here is the caller graph for this function:



### 11.19.2.3 template<typename Node, typename Get\_key, typename Compare> Const\_iterator cxx::Bits::Bst< Node, Get\_key, Compare >::begin( ) const [inline]

Get the constant forward iterator for the first element in the set.

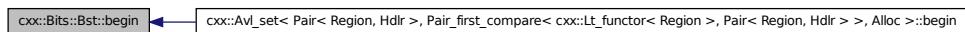
### Returns

Constant forward iterator for the first element in the set.

Definition at line 159 of file bst.h.

Referenced by [cxx::Avl\\_set< Pair< Region, Hdlr >, Pair\\_first\\_compare< cxx::Lt\\_functor< Region >, Pair< Region, Hdlr > >, Alloc >::begin\(\)](#).

Here is the caller graph for this function:



### 11.19.2.4 template<typename Node, typename Get\_key, typename Compare> Const\_iterator cxx::Bits::Bst< Node, Get\_key, Compare >::end( ) const [inline]

Get the end marker for the constant forward iterator.

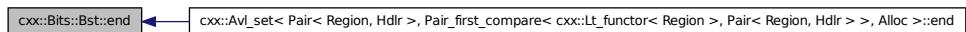
## Returns

The end marker for the constant forward iterator.

Definition at line 164 of file [bst.h](#).

Referenced by [cxx::Avl\\_set< Pair< Region, Hdlr >, Pair\\_first\\_compare< cxx::Lt\\_functor< Region >, Pair< Region, Hdlr > >, Alloc >::end\(\)](#).

Here is the caller graph for this function:



### **11.19.2.5 template<typename Node, typename Get\_key, typename Compare> Iterator cxx::Bits::Bst< Node, Get\_key, Compare >::begin( ) [inline]**

Get the mutable forward iterator for the first element of the set.

## Returns

The mutable forward iterator for the first element of the set.

Definition at line 170 of file [bst.h](#).

### **11.19.2.6 template<typename Node, typename Get\_key, typename Compare> Iterator cxx::Bits::Bst< Node, Get\_key, Compare >::end( ) [inline]**

Get the end marker for the mutable forward iterator.

## Returns

The end marker for mutable forward iterator.

Definition at line 175 of file [bst.h](#).

### **11.19.2.7 template<typename Node, typename Get\_key, typename Compare> Const\_rev\_iterator cxx::Bits::Bst< Node, Get\_key, Compare >::rbegin( ) const [inline]**

Get the constant backward iterator for the last element in the set.

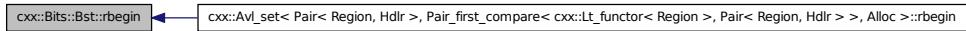
## Returns

The constant backward iterator for the last element in the set.

Definition at line 181 of file [bst.h](#).

Referenced by [cxx::Avl\\_set< Pair< Region, Hdlr >, Pair\\_first\\_compare< cxx::Lt\\_functor< Region >, Pair< Region, Hdlr > >, Alloc >::rbegin\(\)](#).

Here is the caller graph for this function:



### 11.19.2.8 template<typename Node, typename Get\_key, typename Compare> Const\_rev\_iterator cxx::Bits::Bst< Node, Get\_key, Compare >::rend( ) const [inline]

Get the end marker for the constant backward iterator.

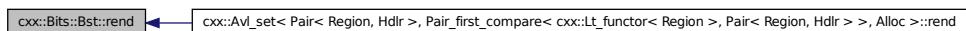
#### Returns

The end marker for the constant backward iterator.

Definition at line 186 of file [bst.h](#).

Referenced by [cxx::Avl\\_set< Pair< Region, Hdr >, Pair\\_first\\_compare< cxx::Lt\\_functor< Region >, Pair< Region, Hdr > >, Alloc >::rend\(\)](#).

Here is the caller graph for this function:



### 11.19.2.9 template<typename Node, typename Get\_key, typename Compare> Rev\_iterator cxx::Bits::Bst< Node, Get\_key, Compare >::rbegin( ) [inline]

Get the mutable backward iterator for the last element of the set.

#### Returns

The mutable backward iterator for the last element of the set.

Definition at line 192 of file [bst.h](#).

### 11.19.2.10 template<typename Node, typename Get\_key, typename Compare> Rev\_iterator cxx::Bits::Bst< Node, Get\_key, Compare >::rend( ) [inline]

Get the end marker for the mutable backward iterator.

#### Returns

The end marker for mutable backward iterator.

Definition at line 197 of file [bst.h](#).

```
11.19.2.11 template<typename Node , typename Get_key , class Compare > Node *
cxx::Bits::Bst< Node, Get_key, Compare >::find_node ( Key_param_type key ) const
[inline]
```

find the node with the given *key*.

#### Parameters

*key* The key value of the element to search.

#### Returns

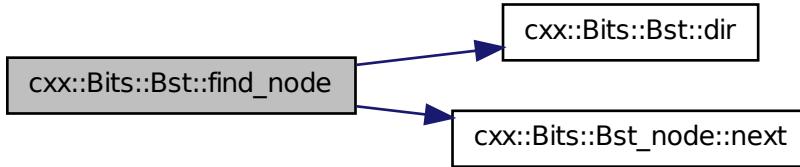
A pointer to the node with the given *key*, or NULL if *key* was not found.

Definition at line 236 of file bst.h.

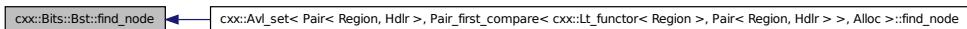
References [cxx::Bits::Bst< Node, Get\\_key, Compare >::\\_head](#), [cxx::Bits::Bst< Node, Get\\_key, Compare >::dir\(\)](#), [cxx::Bits::Direction::N](#), and [cxx::Bits::Bst\\_node::next\(\)](#).

Referenced by [cxx::Avl\\_set< Pair< Region, Hdlr >, Pair\\_first\\_compare< cxx::Lt\\_functor< Region >, Pair< Region, Hdlr > >, Alloc >::find\\_node\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



```
11.19.2.12 template<typename Node , typename Get_key , class Compare > Node *
cxx::Bits::Bst< Node, Get_key, Compare >::lower_bound_node ( Key_param_type
key ) const [inline]
```

find the first node with a key not less than the given *key*.

#### Parameters

*key* The key value of the element to search.

### Returns

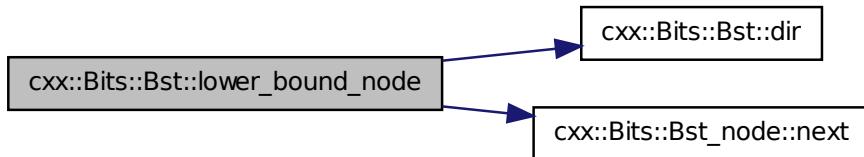
A pointer to the node with the given *key*, or NULL if *key* was not found.

Definition at line 252 of file bst.h.

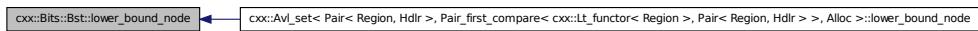
References [cxx::Bits::Bst< Node, Get\\_key, Compare >::\\_head](#), [cxx::Bits::Bst< Node, Get\\_key, Compare >::dir\(\)](#), [cxx::Bits::Direction::L](#), [cxx::Bits::Direction::N](#), and [cxx::Bits::Bst\\_node::next\(\)](#).

Referenced by [cxx::Avl\\_set< Pair< Region, Hdlr >, Pair\\_first\\_compare< cxx::Lt\\_functor< Region >, Pair< Region, Hdlr > >, Alloc >::lower\\_bound\\_node\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 11.19.2.13 template<typename Node , typename Get\_key , class Compare > Bst< Node, Get\_key, Compare >::Const\_iterator cxx::Bits::Bst< Node, Get\_key, Compare >::find ( Key\_param\_type key ) const [inline]

find the node with the given *key*.

### Parameters

*key* The key value of the element to search.

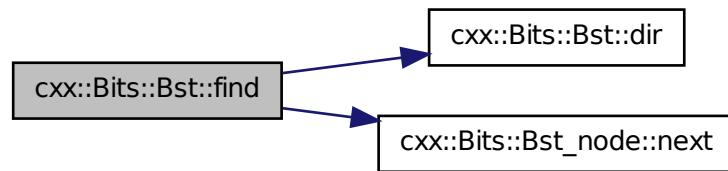
### Returns

A valid iterator for the node with the given *key*, or an invalid iterator if *key* was not found.

Definition at line 272 of file bst.h.

References [cxx::Bits::Bst< Node, Get\\_key, Compare >::\\_head](#), [cxx::Bits::Bst< Node, Get\\_key, Compare >::dir\(\)](#), [cxx::Bits::Direction::L](#), [cxx::Bits::Direction::N](#), and [cxx::Bits::Bst\\_node::next\(\)](#).

Here is the call graph for this function:



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

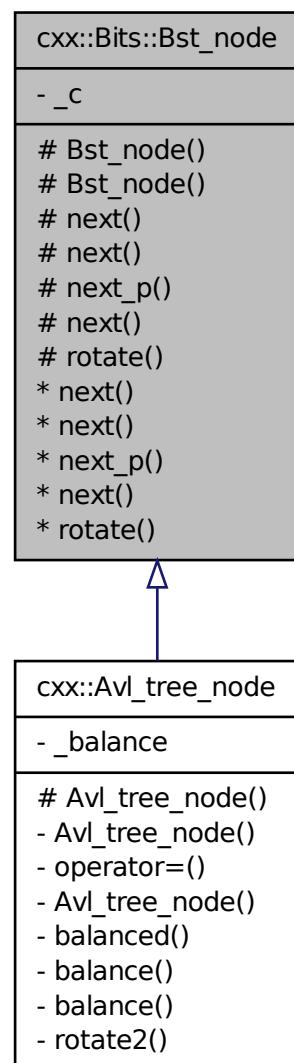
- l4/cxx/bits/bst.h

## 11.20 cxx::Bits::Bst\_node Class Reference

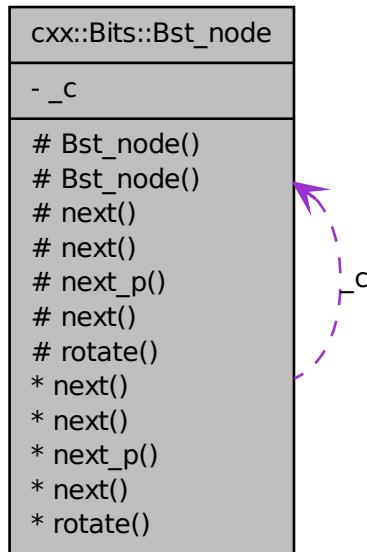
Basic type of a node in a binary search tree (BST).

```
#include <bst_base.h>
```

Inheritance diagram for cxx::Bits::Bst\_node:



Collaboration diagram for cxx::Bits::Bst\_node:



## Protected Member Functions

- `Bst_node ()`  
*Create uninitialized node.*
- `Bst_node (bool)`  
*Create initialized node.*

## Static Protected Member Functions

### Access to BST linkage.

*Provide access to the tree linkage to inherited classes Inherited nodes, such as AVL nodes should make these methods private via 'using'*

- static `Bst_node * next (Bst_node const *p, Direction d)`  
*Get next node in direction d.*
- static void `next (Bst_node *p, Direction d, Bst_node *n)`  
*Set next node of p in direction d to n.*
- static `Bst_node ** next_p (Bst_node *p, Direction d)`  
*Get pointer to link in direction d.*

- template<typename Node >  
static Node \* **next** (**Bst\_node** const \*p, **Direction** d)  
*Get next node in direction d as type Node.*
- static void **rotate** (**Bst\_node** \*\*t, **Direction** idir)  
*Rotate subtree t in the opposite direction of idir.*

### 11.20.1 Detailed Description

Basic type of a node in a binary search tree (BST).

Definition at line 77 of file [bst\\_base.h](#).

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

- 14/cxx/bits/bst\_base.h

## 11.21 L4::Ipc::Buf\_cp\_in< T > Class Template Reference

Abstraction for extracting array from an [Ipc::Istream](#).

### Public Member Functions

- **Buf\_cp\_in** (T \*v, unsigned long &size)  
*Create a buffer for extracting an array from an [Ipc::Istream](#).*

### 11.21.1 Detailed Description

**template<typename T> class L4::Ipc::Buf\_cp\_in< T >**

Abstraction for extracting array from an [Ipc::Istream](#). An instance of **Buf\_cp\_in** can be used to extract an array from an [Ipc::Istream](#). This is the counterpart to the [Buf\\_cp\\_out](#) abstraction. The data from the received message is thereby copied to the given buffer and size is set to the number of elements found in the stream. To avoid the copy operation [Buf\\_in](#) may be used instead.

### See also

[buf\\_cp\\_in\(\)](#), [Buf\\_in](#), [buf\\_in\(\)](#), [Buf\\_cp\\_out](#), and [buf\\_cp\\_out\(\)](#).

Definition at line 117 of file [ipc\\_stream](#).

### 11.21.2 Constructor & Destructor Documentation

#### 11.21.2.1 **template<typename T> L4::Ipc::Buf\_cp\_in< T >::Buf\_cp\_in ( T \* v, unsigned long & size ) [inline]**

Create a buffer for extracting an array from an [Ipc::Istream](#).

## Parameters

- v* The buffer for array (copy in).
- size* Input: the number of elements the array can take at most  
Output: the number of elements found in the stream.

Definition at line 126 of file [ipc\\_stream](#).

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

- l4/cxx/ipc\_stream

## 11.22 L4::Ipc::Buf\_cp\_out< T > Class Template Reference

Abstraction for inserting an array into an [Ipc::Ostream](#).

### Public Member Functions

- [Buf\\_cp\\_out](#) (T \*v, unsigned long size)  
*Create a buffer object for the given array.*
- unsigned long [size](#) () const  
*Get the number of elements in the array.*
- T \* [buf](#) () const  
*Get the pointer to the array.*

### 11.22.1 Detailed Description

**template<typename T> class L4::Ipc::Buf\_cp\_out< T >**

Abstraction for inserting an array into an [Ipc::Ostream](#). An object of [Buf\\_cp\\_out](#) can be used to insert an array of arbitrary values, that can be inserted into an [Ipc::Ostream](#) individually. The array is therefore copied to the message buffer, in contrast to data handled with [Msg\\_out\\_buffer](#) or [Msg\\_io\\_buffer](#).

On insertion into the [Ipc::Ostream](#) exactly the given number of elements of type T are copied to the message buffer, this means the source buffer is no longer referenced after insertion into the stream.

You should use [buf\\_cp\\_out\(\)](#) to create instances of [Buf\\_cp\\_out](#).

The counterpart is either [Buf\\_cp\\_in](#) ([buf\\_cp\\_in\(\)](#)) or [Buf\\_in](#) ([buf\\_in\(\)](#)).

Definition at line 61 of file [ipc\\_stream](#).

### 11.22.2 Constructor & Destructor Documentation

**11.22.2.1 template<typename T> L4::Ipc::Buf\_cp\_out< T >::Buf\_cp\_out ( T \* v, unsigned long size ) [inline]**

Create a buffer object for the given array.

**Parameters**

- v* The pointer to the array with size elements of type T.
- size* the number of elements in the array.

Definition at line 69 of file [ipc\\_stream](#).

### 11.22.3 Member Function Documentation

#### 11.22.3.1 template<typename T> unsigned long L4::Ipc::Buf\_cp\_out< T >::size ( ) const [inline]

Get the number of elements in the array.

**Note**

This function is usually used by the [Ipc::Ostream](#) itself.

Definition at line 75 of file [ipc\\_stream](#).

#### 11.22.3.2 template<typename T> T\* L4::Ipc::Buf\_cp\_out< T >::buf ( ) const [inline]

Get the pointer to the array.

**Note**

This function is usually used by the [Ipc::Ostream](#) itself.

Definition at line 81 of file [ipc\\_stream](#).

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

- l4/cxx/ipc\_stream

## 11.23 L4::Ipc::Buf\_in< T > Class Template Reference

Abstraction to extract an array from an [Ipc::Istream](#).

### Public Member Functions

- [\*\*Buf\\_in\*\*](#) (T \*&*v*, unsigned long &*size*)

*Create an [Buf\\_in](#) to adjust a pointer to the array and the size of the array.*

### 11.23.1 Detailed Description

#### template<typename T> class L4::Ipc::Buf\_in< T >

Abstraction to extract an array from an [Ipc::Istream](#). This wrapper provides a possibility to extract an array from an [Ipc::Istream](#), without extra copy overhead. In contrast to [Buf\\_cp\\_in](#) the data is not copied to a buffer, but a pointer to the array is returned.

The mechanism is comparable to that of `Msg_ptr`, however it handles arrays inserted with `Buf_cp_out`.

See `buf_in()`, `Buf_cp_out`, `buf_cp_out()`, `Buf_cp_in`, and `buf_cp_in()`.

Definition at line 211 of file ipc\_stream.

### 11.23.2 Constructor & Destructor Documentation

**11.23.2.1 template<typename T> L4::Ipc::Buf\_in< T >::Buf\_in ( T \*& v, unsigned long & size ) [inline]**

Create an `Buf_in` to adjust a pointer to the array and the size of the array.

## Parameters

v The pointer to adjust to the first element of the array.

**size** The number of elements found in the stream.

Definition at line 220 of file ipc\_stream.

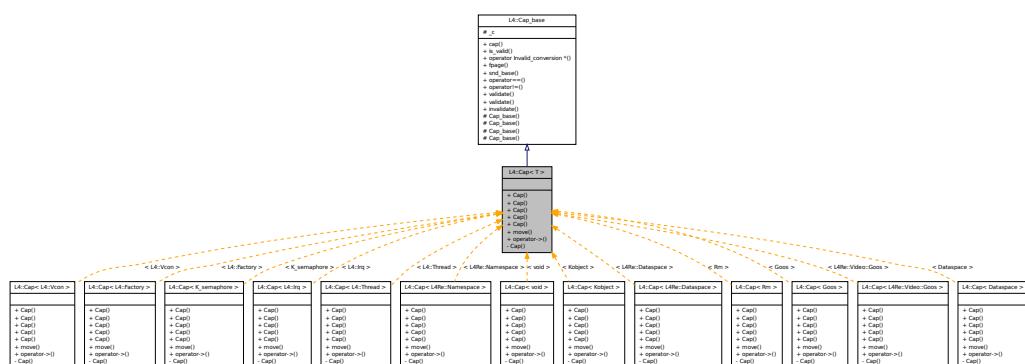
The documentation for this class was generated from the following file:

- 14/cxx/ipc stream

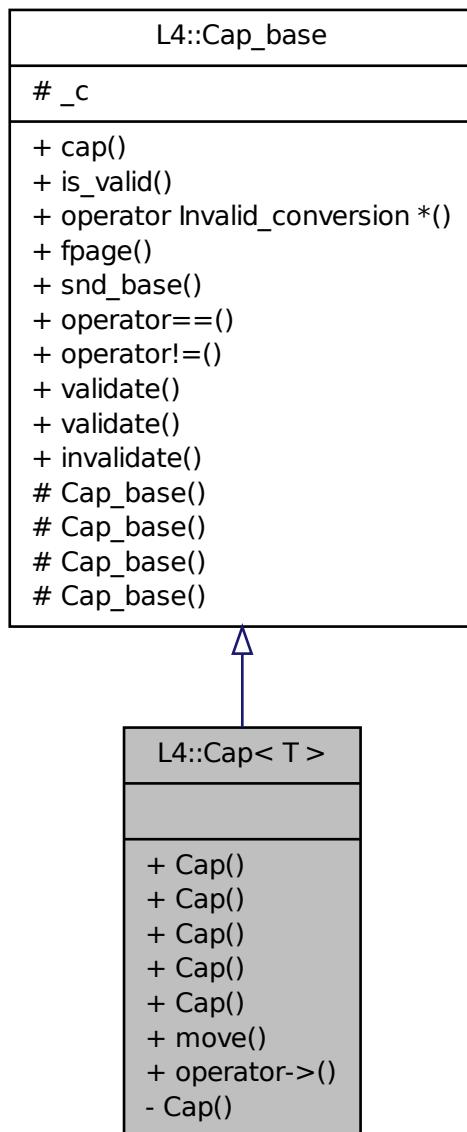
## 11.24 L4::Cap< T > Class Template Reference

Capability Selector a la C++.

## Inheritance diagram for L4::Cap< T >:



Collaboration diagram for L4::Cap< T >:



## Public Member Functions

- template<typename O >  
[Cap](#) ([Cap](#)< O > const &o) throw ()  
*Create a copy from o, supporting implicit type casting.*
- [Cap](#) ([Cap\\_type](#) cap) throw ()

*Constructor to create an invalid capability selector.*

- **Cap (l4\_default\_caps\_t cap) throw ()**  
*Initialize capability with one of the default capability selectors.*
- **Cap (l4\_cap\_idx\_t idx=L4\_INVALID\_CAP) throw ()**  
*Initialize capability, defaults to the invalid capability selector.*
- **Cap (No\_init\_type) throw ()**  
*Create an uninitialized cap selector.*
- **Cap move (Cap const &src) const**  
*Move a capability to this cap slot.*
- **T \* operator-> () const throw ()**  
*Member access of a T.*

## Friends

- class [L4::Kobject](#)

### 11.24.1 Detailed Description

**template<typename T> class L4::Cap< T >**

Capability Selector a la C++.

#### Template Parameters

*T* the type of the object the capability points to

The C++ version of a capability looks just as a pointer, in fact it is a kind of a smart pointer for our kernel objects and the objects derived from the kernel objects ([L4::Kobject](#)).

#### Examples:

[examples/clntsrv/client.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#), and [examples/libs/l4re/streammap/client.cc](#).

Definition at line 200 of file [capability](#).

### 11.24.2 Constructor & Destructor Documentation

#### 11.24.2.1 template<typename T> template<typename O > L4::Cap< T >::Cap ( Cap< O > const & o ) throw () [inline]

Create a copy from *o*, supporting implicit type casting.

#### Parameters

*o* is the source selector that shall be copied (and casted).

Definition at line 225 of file [capability](#).

#### 11.24.2.2 template<typename T> L4::Cap< T >::Cap ( l4\_default\_caps\_t *cap* ) throw () [inline]

Initialize capability with one of the default capability selectors.

##### Parameters

*cap* Capability selector.

Definition at line 237 of file [capability](#).

#### 11.24.2.3 template<typename T> L4::Cap< T >::Cap ( l4\_cap\_idx\_t *idx* = L4\_INVALID\_CAP ) throw () [inline, explicit]

Initialize capability, defaults to the invalid capability selector.

##### Parameters

*idx* Capability selector.

Definition at line 243 of file [capability](#).

### 11.24.3 Member Function Documentation

#### 11.24.3.1 template<typename T> Cap L4::Cap< T >::move ( Cap< T > const & *src* ) const [inline]

Move a capability to this cap slot.

##### Parameters

*src* the source capability slot.

After this operation the source slot is no longer valid.

Definition at line 256 of file [capability](#).

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

- l4/sys/capability

## 11.25 L4Re::Cap\_alloc Class Reference

Capability allocator interface.

### Public Member Functions

- virtual L4::Cap< void > alloc ()=0 throw ()

*Allocate a capability.*

- template<typename T >  
`L4::Cap< T > alloc () throw ()`

*Allocate a capability.*

- virtual void `free (L4::Cap< void > cap)=0 throw ()`

*Free a capability.*

- virtual `~Cap_alloc ()=0`

*Destructor.*

## Static Public Member Functions

- template<typename CAP\_ALLOC >  
`static L4Re::Cap_alloc * get_cap_alloc (CAP_ALLOC &ca)`

*Construct an instance of a capability allocator.*

### 11.25.1 Detailed Description

Capability allocator interface.

Definition at line 39 of file `cap_alloc.h`.

### 11.25.2 Member Function Documentation

#### 11.25.2.1 virtual L4::Cap<void> L4Re::Cap\_alloc::alloc ( ) throw () [pure virtual]

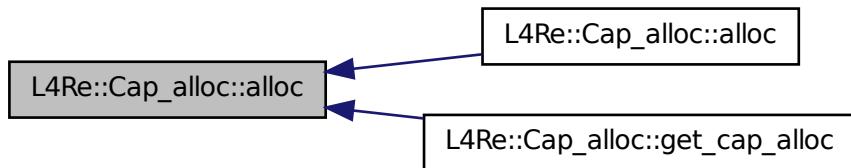
Allocate a capability.

##### Returns

Capability of type void

Referenced by `alloc()`, and `get_cap_alloc()`.

Here is the caller graph for this function:



### 11.25.2.2 template<typename T > L4::Cap<T> L4Re::Cap\_alloc::alloc ( ) throw () [inline]

Allocate a capability.

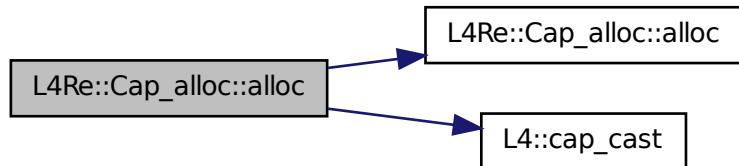
#### Returns

Capability of type T

Definition at line 61 of file [cap\\_alloc](#).

References [alloc\(\)](#), and [L4::cap\\_cast\(\)](#).

Here is the call graph for this function:



### 11.25.2.3 virtual void L4Re::Cap\_alloc::free ( L4::Cap< void > cap ) throw () [pure virtual]

Free a capability.

#### Parameters

*cap* Capability to free.

Referenced by [get\\_cap\\_alloc\(\)](#).

Here is the caller graph for this function:



**11.25.2.4 template<typename CAP\_ALLOC> static L4Re::Cap\_alloc\*  
L4Re::Cap\_alloc::get\_cap\_alloc( CAP\_ALLOC & ca ) [inline, static]**

Construct an instance of a capability allocator.

#### Parameters

*ca* Capabilty allocator

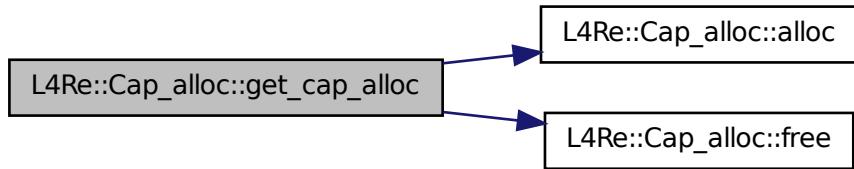
#### Returns

Instance of a capability allocator.

Definition at line 84 of file [cap\\_alloc](#).

References [alloc\(\)](#), and [free\(\)](#).

Here is the call graph for this function:



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

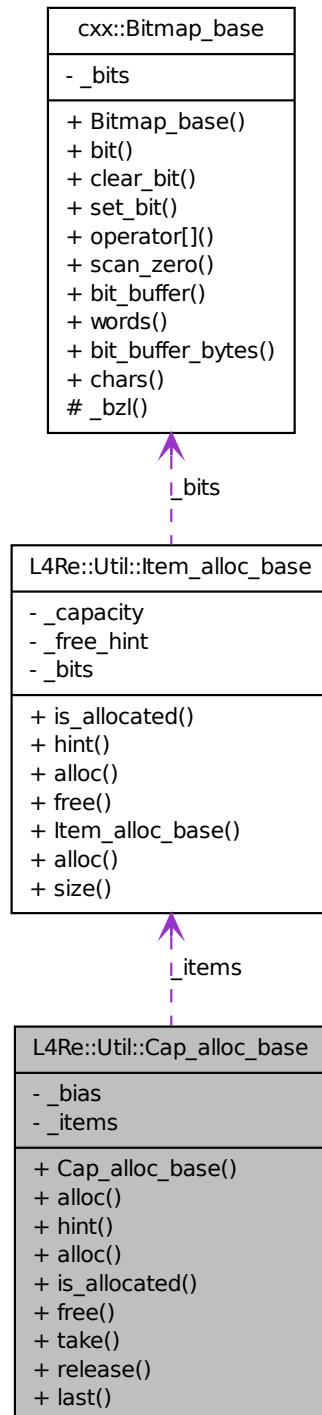
- [l4/re/cap\\_alloc](#)

## 11.26 L4Re::Util::Cap\_alloc\_base Class Reference

Capability allocator.

Inherited by L4Re::Util::Cap\_alloc< Size >.

Collaboration diagram for L4Re::Util::Cap\_alloc\_base:



## Public Member Functions

- template<typename T >  
**L4::Cap< T > alloc () throw ()**  
*Allocate a capability slot.*
- template<typename T >  
**void free (L4::Cap< T > const &cap, l4\_cap\_idx\_t task=-1UL, l4\_umword\_t unmap\_flags=L4\_FTP\_ALL\_SPACES) throw ()**  
*Free a capability slot.*

### 11.26.1 Detailed Description

Capability allocator.

Definition at line 35 of file [bitmap\\_cap\\_alloc](#).

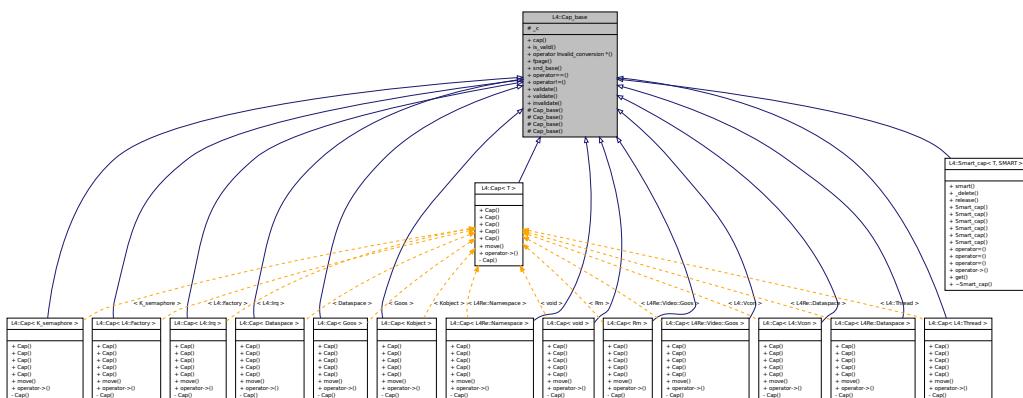
The documentation for this class was generated from the following file:

- l4/re/util/bitmap\_cap\_alloc

## 11.27 L4::Cap\_base Class Reference

Base class for all kinds of capabilities.

Inheritance diagram for L4::Cap\_base:



## Public Types

- enum [No\\_init\\_type](#) { [No\\_init](#) }
  - enum [Cap\\_type](#) { [Invalid](#) = L4\_INVALID\_CAP }
- Invalid capability type.*

## Public Member Functions

- `l4_cap_idx_t cap () const throw ()`  
*Return capability selector.*
- `bool is_valid () const throw ()`  
*Test whether capability selector is not the invalid capability selector.*
- `l4_fpage_t fpage (unsigned rights=L4_FPAGE_RWX) const throw ()`  
*Returns flex-page of the capability selector.*
- `l4_umword_t snd_base (unsigned grant=0, l4_cap_idx_t base=L4_INVALID_CAP) const throw ()`  
*Returns send base.*
- `bool operator==(Cap_base const &o) const throw ()`  
*Test if two capability selectors are equal.*
- `bool operator!=(Cap_base const &o) const throw ()`  
*Test if two capability selectors are not equal.*
- `l4_msgtag_t validate (l4_utcb_t *u=l4_utcb()) const throw ()`  
*Check whether a capability selector points to a valid capability.*
- `l4_msgtag_t validate (Cap< Task > task, l4_utcb_t *u=l4_utcb()) const throw ()`  
*Check whether a capability selector points to a valid capability.*
- `void invalidate () throw ()`  
*Set this selector to the invalid capability (L4\_INVALID\_CAP).*

## Protected Member Functions

- `Cap_base (l4_cap_idx_t c) throw ()`  
*Generate a capability from its C representation.*
- `Cap_base (Cap_type cap) throw ()`  
*Constructor to create an invalid capability selector.*
- `Cap_base (l4_default_caps_t cap) throw ()`  
*Initialize capability with one of the default capability selectors.*
- `Cap_base () throw ()`  
*Create an uninitialized instance.*

## Protected Attributes

- `l4_cap_idx_t _c`  
*The C representation of a capability selector.*

### 11.27.1 Detailed Description

Base class for all kinds of capabilities.

#### Attention

This class is not for direct use, use [L4::Cap](#) instead.

This class contains all the things that are independent of the type of the object referred by the capability.

#### See also

[L4::Cap](#) for typed capabilities.

Definition at line [66](#) of file [capability](#).

### 11.27.2 Member Enumeration Documentation

#### 11.27.2.1 enum L4::Cap\_base::No\_init\_type

##### Enumerator:

*No\_init* Special value for constructing uninitialized [Cap](#) objects.

Definition at line [72](#) of file [capability](#).

#### 11.27.2.2 enum L4::Cap\_base::Cap\_type

Invalid capability type.

##### Enumerator:

*Invalid* Invalid capability selector.

Definition at line [83](#) of file [capability](#).

### 11.27.3 Constructor & Destructor Documentation

#### 11.27.3.1 L4::Cap\_base::Cap\_base ( l4\_cap\_idx\_t c ) throw () [inline, explicit, protected]

Generate a capability from its C representation.

##### Parameters

*c* the C capability selector

Definition at line [168](#) of file [capability](#).

**11.27.3.2 L4::Cap\_base::Cap\_base ( l4\_default\_caps\_t *cap* ) throw () [inline, explicit, protected]**

Initialize capability with one of the default capability selectors.

**Parameters**

*cap* Capability selector.

Definition at line 178 of file [capability](#).

**11.27.4 Member Function Documentation****11.27.4.1 l4\_cap\_idx\_t L4::Cap\_base::cap ( ) const throw () [inline]**

Return capability selector.

**Returns**

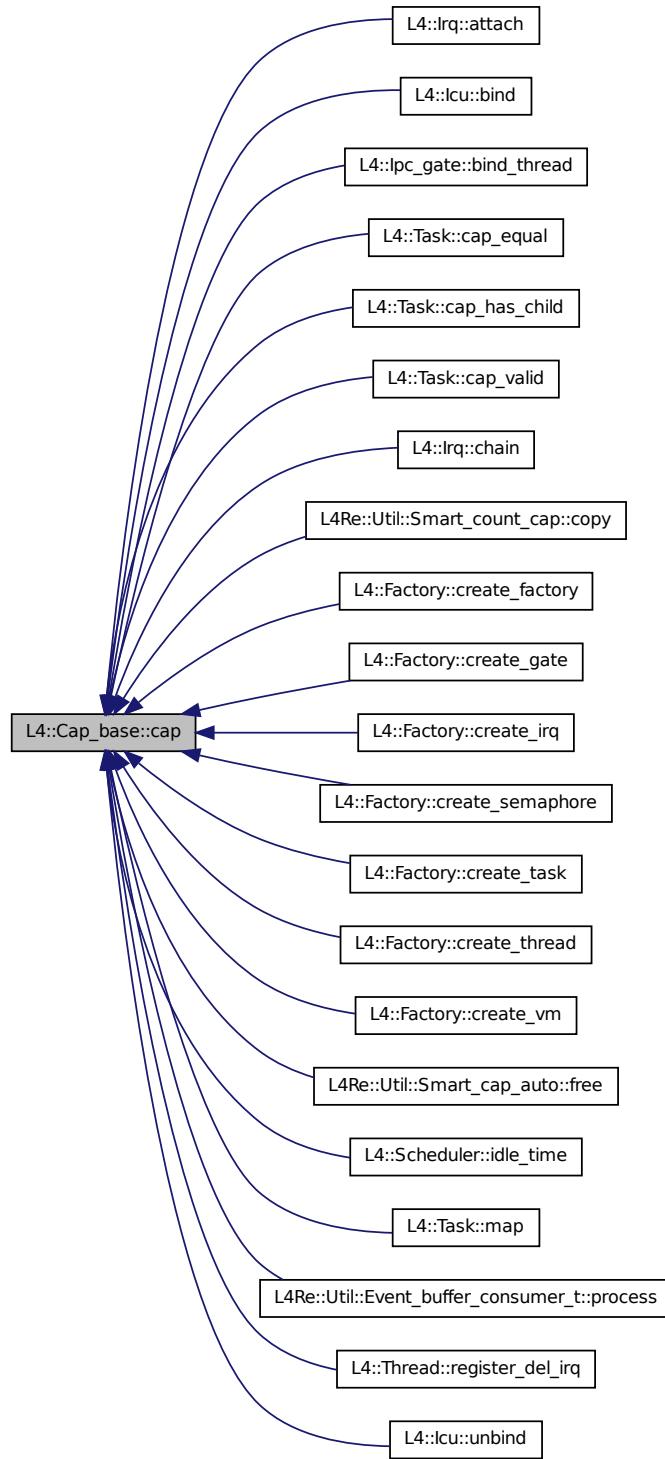
Capability selector.

Definition at line 92 of file [capability](#).

References [\\_c](#).

Referenced by [L4::Irq::attach\(\)](#), [L4::Icu::bind\(\)](#), [L4::Ipc\\_gate::bind\\_thread\(\)](#), [L4::Task::cap\\_equal\(\)](#), [L4::Task::cap\\_has\\_child\(\)](#), [L4::Task::cap\\_valid\(\)](#), [L4::Irq::chain\(\)](#), [L4Re::Util::Smart\\_count\\_cap<Unmap\\_flags>::copy\(\)](#), [L4::Factory::create\\_factory\(\)](#), [L4::Factory::create\\_gate\(\)](#), [L4::Factory::create\\_irq\(\)](#), [L4::Factory::create\\_semaphore\(\)](#), [L4::Factory::create\\_task\(\)](#), [L4::Factory::create\\_thread\(\)](#), [L4::Factory::create\\_vm\(\)](#), [L4Re::Util::Smart\\_cap\\_auto<Unmap\\_flags>::free\(\)](#), [L4::Scheduler::idle\\_time\(\)](#), [L4::Task::map\(\)](#), [L4Re::Util::Event\\_buffer\\_consumer\\_t<PAYLOAD>::process\(\)](#), [L4::Thread::register\\_del\\_irq\(\)](#), and [L4::Icu::unbind\(\)](#).

Here is the caller graph for this function:



### 11.27.4.2 bool L4::Cap\_base::is\_valid( ) const throw() [inline]

Test whether capability selector is not the invalid capability selector.

#### Returns

True if capability is not invalid, false if invalid

#### Examples:

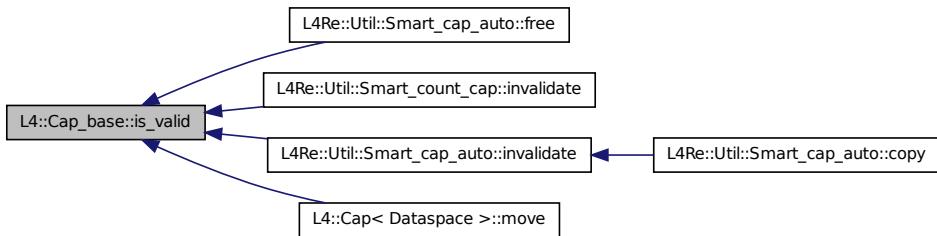
[examples/libs/l4re/c++/mem\\_alloc/ma+rm.cc](#), and [examples/libs/l4re/c++/shared\\_ds/ds\\_clnt.cc](#).

Definition at line 100 of file [capability](#).

References [\\_c](#).

Referenced by [L4Re::Util::Smart\\_cap\\_auto< Unmap\\_flags >::free\(\)](#), [L4Re::Util::Smart\\_count\\_cap< Unmap\\_flags >::invalidate\(\)](#), [L4Re::Util::Smart\\_cap\\_auto< Unmap\\_flags >::invalidate\(\)](#), and [L4::Cap< Dataspace >::move\(\)](#).

Here is the caller graph for this function:



### 11.27.4.3 l4\_fpage\_t L4::Cap\_base::fpage( unsigned rights = L4\_FPAGE\_RWX ) const throw() [inline]

Returns flex-page of the capability selector.

#### Parameters

*rights* Rights, defaults to 'rwx'

#### Returns

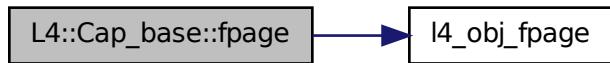
flex-page

Definition at line 110 of file [capability](#).

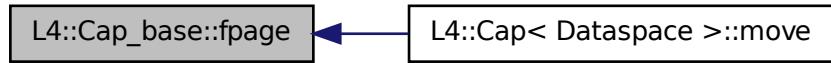
References [\\_c](#), and [l4\\_obj\\_fpage\(\)](#).

Referenced by [L4::Cap< Dataspace >::move\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 11.27.4.4 `l4_umword_t L4::Cap_base::snd_base( unsigned grant = 0, l4_cap_idx_t base = L4_INVALID_CAP ) const throw() [inline]`

Returns send base.

##### Parameters

*grant* True object should be granted.

*base* Base capability selector

##### Returns

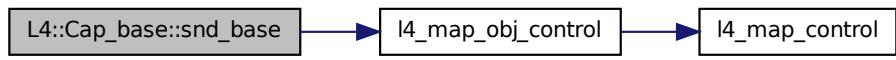
Map object.

Definition at line 119 of file [capability](#).

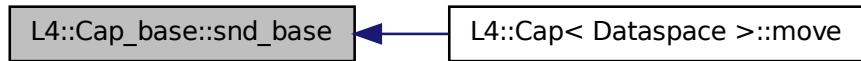
References [\\_c](#), [L4\\_INVALID\\_CAP](#), and [l4\\_map\\_obj\\_control\(\)](#).

Referenced by [L4::Cap< Dataspace >::move\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



#### 11.27.4.5 l4\_msgtag\_t L4::Cap\_base::validate ( l4\_utcb\_t \* u = 14\_utcb() ) const throw () [inline]

Check whether a capability selector points to a valid capability.

##### Parameters

*u* UTCB of the caller

##### Returns

label = 0 valid, label > 0 invalid

Definition at line 514 of file [capability](#).

References [L4\\_BASE\\_TASK\\_CAP](#).

#### 11.27.4.6 l4\_msgtag\_t L4::Cap\_base::validate ( Cap< Task > task, l4\_utcb\_t \* u = 14\_utcb() ) const throw () [inline]

Check whether a capability selector points to a valid capability.

##### Parameters

*u* UTCB of the caller

*task* Task to check the capability in

##### Returns

label = 0 valid, label > 0 invalid

Definition at line 510 of file [capability](#).

## 11.27.5 Field Documentation

#### 11.27.5.1 l4\_cap\_idx\_t L4::Cap\_base::\_c [protected]

The C representation of a capability selector.

Definition at line 187 of file [capability](#).

Referenced by `cap()`, `fpage()`, `invalidate()`, `is_valid()`, `operator!=()`, `L4::Smart_cap< T, SMART >::operator->()`, `L4::Cap< Dataspace >::operator->()`, `operator==()`, and `rnd_base()`.

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

- l4/sys/capability

## 11.28 cxx::Bitmap\_base::Char< BITS > Class Template Reference

Helper abstraction for a byte contained in the bitmap.

### 11.28.1 Detailed Description

`template<long BITS> class cxx::Bitmap_base::Char< BITS >`

Helper abstraction for a byte contained in the bitmap.

Definition at line 66 of file [bitmap](#).

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

- l4/cxx(bitmap

## 11.29 L4Re::Video::Color\_component Class Reference

A color component.

### Public Member Functions

- `Color_component ()`  
*Constructor.*
- `Color_component (unsigned char bits, unsigned char shift)`  
*Constructor.*
- `unsigned char size () const`  
*Return the number of bits used by the component.*
- `unsigned char shift () const`  
*Return the position of the component in the pixel.*
- `bool operator== (Color_component const &o) const`  
*Compare for equality.*
- `int get (unsigned long v) const`  
*Get component from value (normalized to 16bits).*
- `long unsigned set (int v) const`  
*Transform 16bit normalized value to the component in the color space.*

- template<typename STREAM >  
STREAM & **dump** (STREAM &s) const  
*Dump information on the view information to a stream.*

### 11.29.1 Detailed Description

A color component.

Definition at line 32 of file [colors](#).

### 11.29.2 Constructor & Destructor Documentation

#### 11.29.2.1 L4Re::Video::Color\_component::Color\_component ( unsigned char bits, unsigned char shift ) [inline]

Constructor.

##### Parameters

*bits* Number of bits used by the component

*shift* Position in bits of the component in the pixel

Definition at line 47 of file [colors](#).

### 11.29.3 Member Function Documentation

#### 11.29.3.1 unsigned char L4Re::Video::Color\_component::size ( ) const [inline]

Return the number of bits used by the component.

##### Returns

Number of bits used by the component

Definition at line 54 of file [colors](#).

#### 11.29.3.2 unsigned char L4Re::Video::Color\_component::shift ( ) const [inline]

Return the position of the component in the pixel.

##### Returns

Position in bits of the component in the pixel

Definition at line 60 of file [colors](#).

**11.29.3.3 bool L4Re::Video::Color\_component::operator== ( Color\_component const & *o* ) const [inline]**

Compare for equality.

#### Returns

True if the same components are described, false if not.

Definition at line 66 of file [colors](#).

**11.29.3.4 int L4Re::Video::Color\_component::get ( unsigned long *v* ) const [inline]**

Get component from value (normalized to 16bits).

#### Parameters

*v* Value

#### Returns

Converted value

Definition at line 74 of file [colors](#).

**11.29.3.5 long unsigned L4Re::Video::Color\_component::set ( int *v* ) const [inline]**

Transform 16bit normalized value to the component in the color space.

#### Parameters

*v* Value return Converted value.

Definition at line 85 of file [colors](#).

**11.29.3.6 template<typename STREAM> STREAM& L4Re::Video::Color\_component::dump ( STREAM & *s* ) const [inline]**

Dump information on the view information to a stream.

#### Parameters

*s* Stream

#### Returns

The stream

Definition at line 94 of file [colors](#).

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

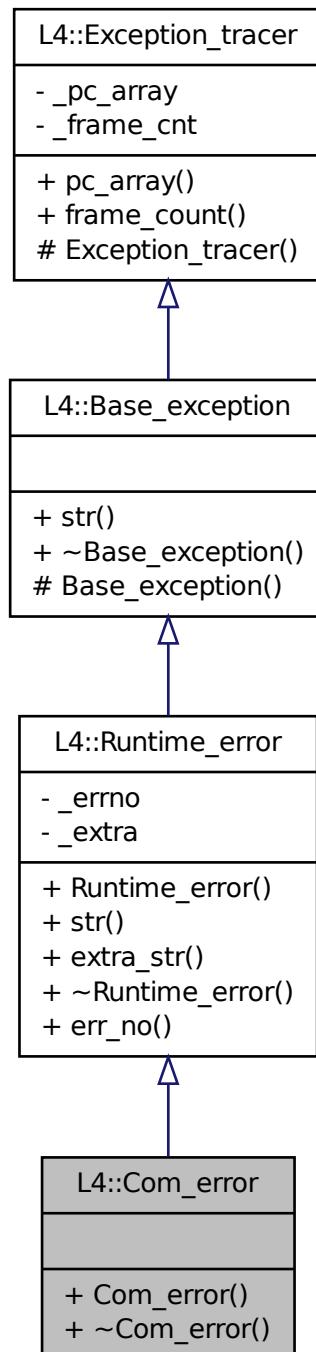
- l4/re/video/colors

## **11.30 L4::Com\_error Class Reference**

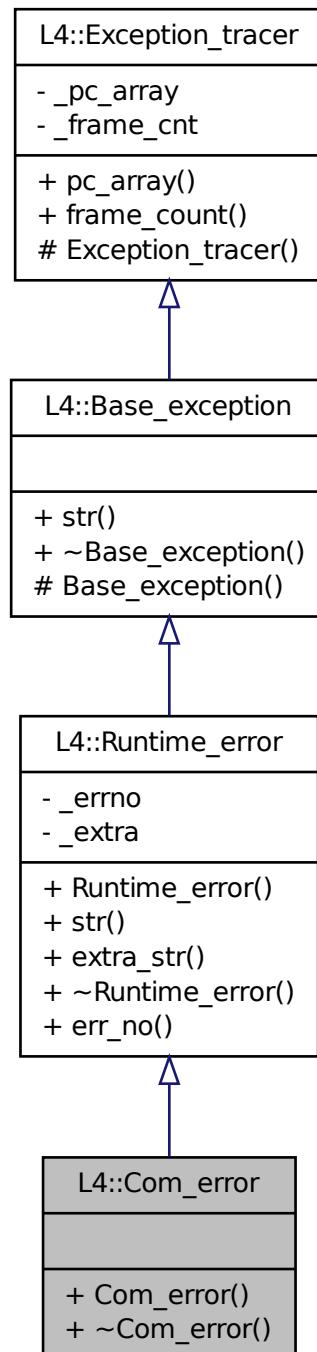
Error conditions during IPC.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Com\_error:



Collaboration diagram for L4::Com\_error:



## Public Member Functions

- [Com\\_error \(long err\) throw \(\)](#)

Create a [Com\\_error](#) for the givel [L4](#) IPC error code.

### 11.30.1 Detailed Description

Error conditions during IPC. This exception encapsulates all IPC error conditions of [L4](#) IPC.

Definition at line [249](#) of file [exceptions](#).

### 11.30.2 Constructor & Destructor Documentation

#### 11.30.2.1 L4::Com\_error::Com\_error ( long *err* ) throw () [inline, explicit]

Create a [Com\\_error](#) for the givel [L4](#) IPC error code.

##### Parameters

*err* The [L4](#) IPC error code (l4\_ipc... return value).

Definition at line [256](#) of file [exceptions](#).

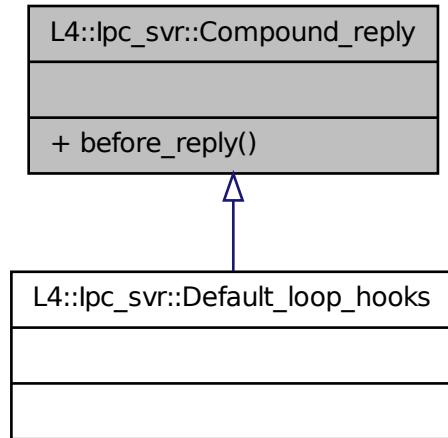
The documentation for this class was generated from the following file:

- l4/cxx/exceptions

## 11.31 L4::Ipc\_svr::Compound\_reply Struct Reference

Mix in for LOOP\_HOOKS to always use compound reply and wait.

Inheritance diagram for L4::Ipc\_svr::Compound\_reply:



### 11.31.1 Detailed Description

Mix in for LOOP\_HOOKS to always use compound reply and wait.

Definition at line 65 of file [ipc\\_server](#).

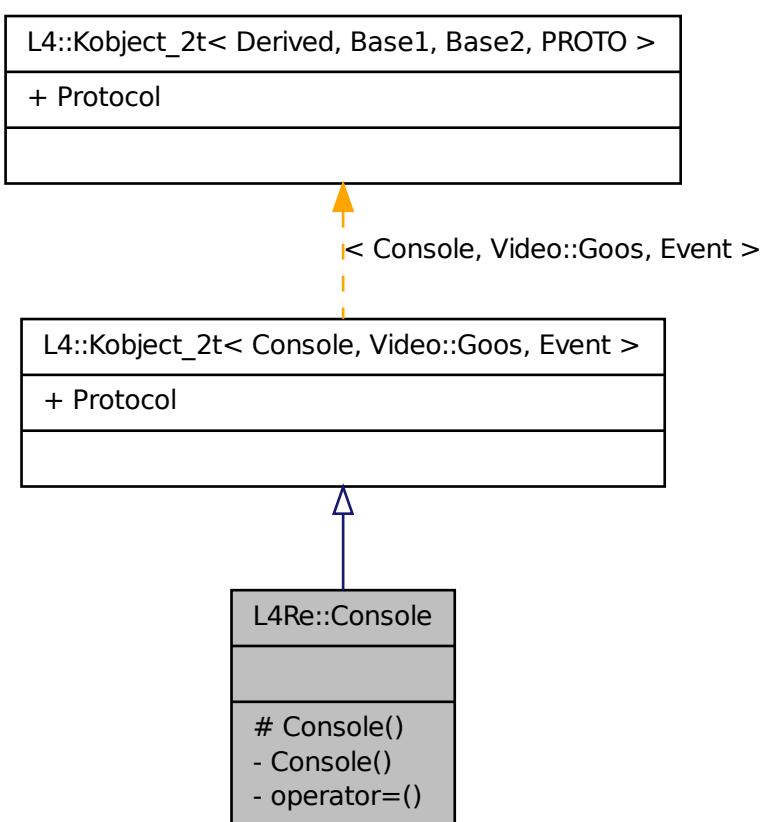
The documentation for this struct was generated from the following file:

- l4/cxx/ipc\_server

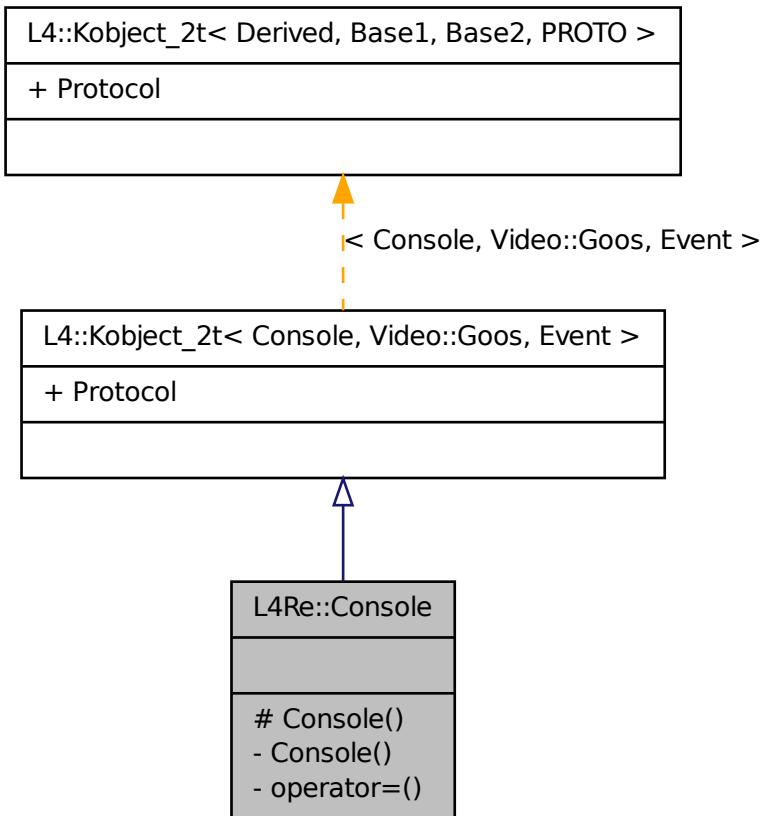
## 11.32 L4Re::Console Class Reference

[Console class](#).

Inheritance diagram for L4Re::Console:



Collaboration diagram for L4Re::Console:



### 11.32.1 Detailed Description

[Console](#) class.

Definition at line 38 of file [console](#).

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

- 14/re/console

## 11.33 L4Re::Util::Counting\_cap\_alloc< COUNTERTYPE > Class Template Reference

Reference-counting cap allocator.

### 11.33.1 Detailed Description

```
template<typename COUNTERTYPE = L4Re::Util::Counter<unsigned char>> class  
L4Re::Util::Counting_cap_alloc<COUNTERTYPE >
```

Reference-counting cap allocator.

Definition at line 52 of file [counting\\_cap\\_alloc](#).

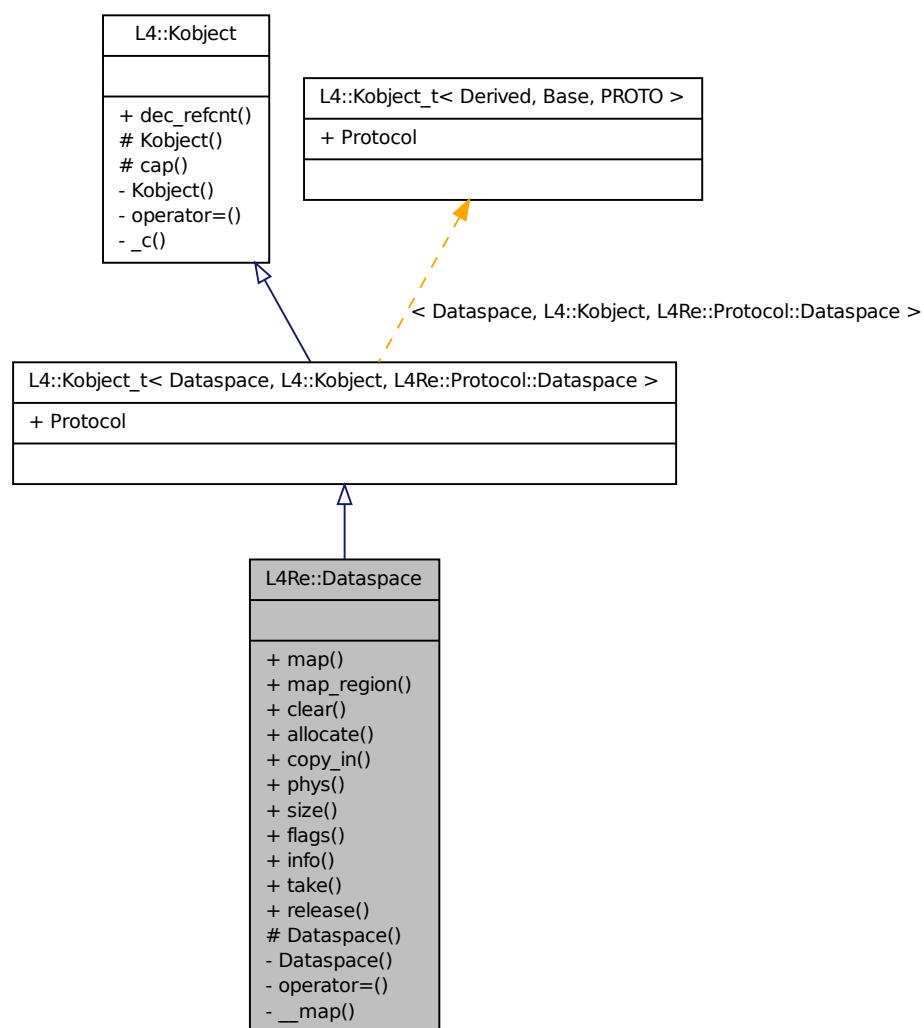
The documentation for this class was generated from the following file:

- l4/re/util/counting\_cap\_alloc

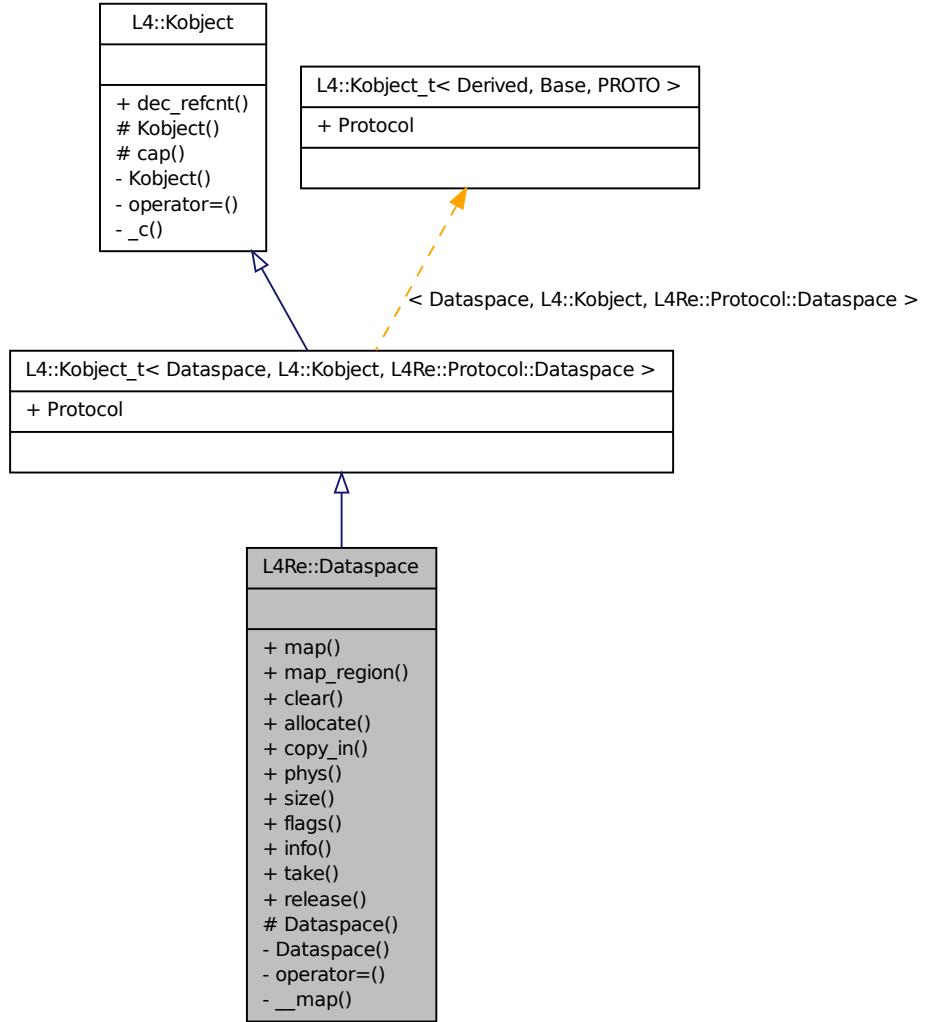
## 11.34 L4Re::Dataspace Class Reference

This class represents a data space.

Inheritance diagram for L4Re::Dataspace:



Collaboration diagram for L4Re::Dataspace:



## Data Structures

- struct [Stats](#)

*Information about the data space.*

## Public Types

- enum [Map\\_flags](#) { `Map_ro` = 0, `Map_rw` = 1 }

*Flags for map operations.*

## Public Member Functions

- long `map (l4_addr_t offset, unsigned long flags, l4_addr_t local_addr, l4_addr_t min_addr, l4_addr_t max_addr) const throw ()`  
*Request a flex-page mapping from the data space.*
- long `map_region (l4_addr_t offset, unsigned long flags, l4_addr_t min_addr, l4_addr_t max_addr) const throw ()`  
*Map a part of a data space completely.*
- long `clear (l4_addr_t offset, unsigned long size) const throw ()`  
*Clear parts of a data space.*
- long `allocate (l4_addr_t offset, l4_size_t size) throw ()`  
*Allocate a range in the dataspace.*
- long `copy_in (l4_addr_t dst_offs, L4::Cap< Dataspace > src, l4_addr_t src_offs, unsigned long size) const throw ()`  
*Copy data space contents.*
- long `phys (l4_addr_t offset, l4_addr_t &phys_addr, l4_size_t &phys_size) const throw ()`  
*Get the physical addresses of a data space.*
- long `size () const throw ()`  
*Get size of a data space.*
- long `flags () const throw ()`  
*Get flags of the data space.*
- int `info (Stats *stats) const throw ()`  
*Get information on the data space.*

### 11.34.1 Detailed Description

This class represents a data space. For more details, see [Data-Space API](#).

#### Examples:

`examples/libs/l4re/c++/mem_alloc/ma+rm.cc`, `examples/libs/l4re/c++/shared_ds/ds_clnt.cc`, and  
`examples/libs/l4re/c++/shared_ds/ds_srv.cc`.

Definition at line 67 of file `dataspace`.

### 11.34.2 Member Enumeration Documentation

#### 11.34.2.1 enum L4Re::Dataspace::Map\_flags

Flags for map operations.

**Enumerator:**

*Map\_ro* Request read-only mapping.

*Map\_rw* Request writable mapping.

Definition at line 77 of file [dataspace](#).

### 11.34.3 Member Function Documentation

#### 11.34.3.1 long L4Re::Dataspace::map ( *l4\_addr\_t offset*, *unsigned long flags*, *l4\_addr\_t local\_addr*, *l4\_addr\_t min\_addr*, *l4\_addr\_t max\_addr* ) const throw ()

Request a flex-page mapping from the data space.

**Parameters**

*offset* Offset to start within data space

*flags* map flags, see [Map\\_flags](#).

*local\_addr* Local address to map to.

*min\_addr* Defines start of receive window.

*max\_addr* Defines end of receive window.

**Returns**

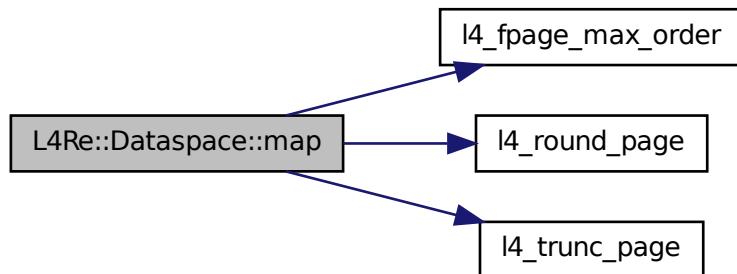
0 on success, <0 on error

- [-L4\\_ERANGE](#)
- [-L4\\_EPERM](#)
- IPC errors

Definition at line 94 of file [dataspace\\_impl.h](#).

References [l4\\_fpage\\_max\\_order\(\)](#), [L4\\_LOG2\\_PAGESIZE](#), [l4\\_round\\_page\(\)](#), and [l4\\_trunc\\_page\(\)](#).

Here is the call graph for this function:



### 11.34.3.2 long L4Re::Dataspace::map\_region ( l4\_addr\_t offset, unsigned long flags, l4\_addr\_t min\_addr, l4\_addr\_t max\_addr ) const throw ()

Map a part of a data space completely.

#### Parameters

- offset* Offset to start within data space
- flags* map flags, see [Map\\_flags](#).
- min\_addr* Defines start of receive window.
- max\_addr* Defines end of receive window.

#### Returns

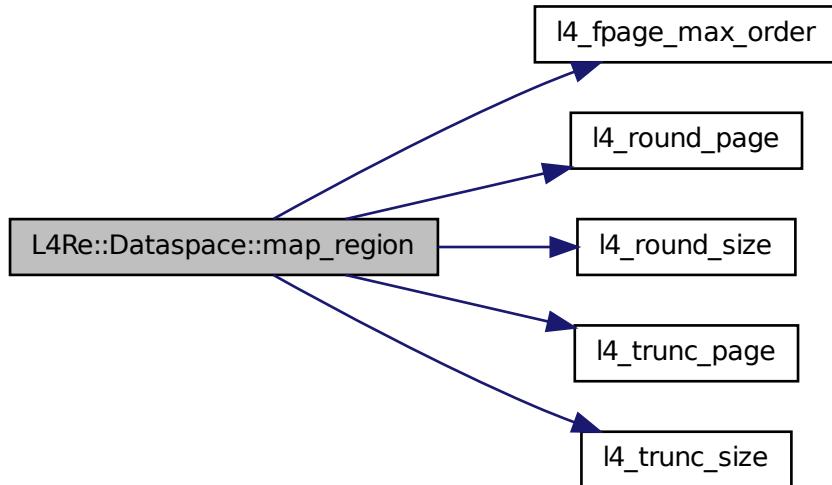
0 on success, <0 on error

- [-L4\\_ERANGE](#)
- [-L4\\_EPERM](#)
- IPC errors

Definition at line 57 of file [dataspace\\_impl.h](#).

References [EXPECT\\_FALSE](#), [l4\\_fpage\\_max\\_order\(\)](#), [l4\\_round\\_page\(\)](#), [l4\\_round\\_size\(\)](#), [l4\\_trunc\\_page\(\)](#), and [l4\\_trunc\\_size\(\)](#).

Here is the call graph for this function:



### 11.34.3.3 long L4Re::Dataspace::clear ( l4\_addr\_t offset, unsigned long size ) const throw ()

Clear parts of a data space.

**Parameters**

*offset* Offset within data space.

*size* Size to clear (in bytes).

**Returns**

>0 on success, <0 on error.

- -L4\_EACCESS
- IPC errors

Clears memory. Depending on the type of memory the memory could also be deallocated and replaced by shared zero-page.

**11.34.3.4 long L4Re::Dataspace::allocate ( l4\_addr\_t offset, l4\_size\_t size ) throw ()**

Allocate a range in the dataspace.

**Parameters**

*offset* Offset in the dataspace, in bytes.

*size* Size of the range, in bytes.

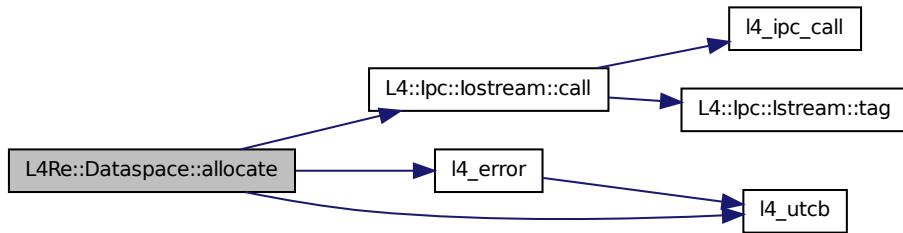
**Returns**

0 on success, <0 on error

Definition at line 177 of file [dataspace\\_impl.h](#).

References [L4::Ipc::Iostream::call\(\)](#), [L4Re::Protocol::Dataspace](#), [l4\\_error\(\)](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:

**11.34.3.5 long L4Re::Dataspace::copy\_in ( l4\_addr\_t dst\_offs, L4::Cap< DataSpace > src, l4\_addr\_t src\_offs, unsigned long size ) const throw ()**

Copy data space contents.

### Parameters

*dst\_offs* Offset in destination data space.

*src* Source data space.

*src\_offs* Offset in the source data space.

*size* Size to copy (in bytes).

### Returns

0 on success, <0 on error

- -L4\_EACCESS
- -L4\_EINVAL
- IPC errors

The copy operation may use copy-on-write mechanisms. The operation may also fail if both data spaces are not from the same data space manager or the data space managers do not cooperate.

### 11.34.3.6 long L4Re::Dataspace::phys ( l4\_addr\_t offset, l4\_addr\_t & phys\_addr, l4\_size\_t & phys\_size ) const throw ()

Get the physical addresses of a data space.

### Parameters

*offset* Offset in data space

### Return values

*phys\_addr* Physical address.

*phys\_size* Size of largest physically contiguous region in the data space (in bytes).

### Returns

0 on success, <0 on error

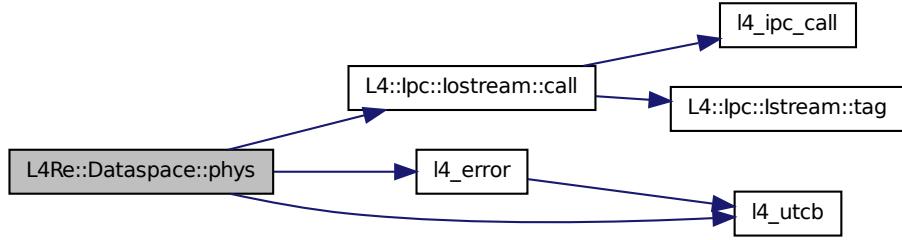
- -L4\_EINVAL
- IPC errors

Get the physical address(es) of a data space. This call will only succeed on pinned memory data spaces.

Definition at line 164 of file [dataspace\\_impl.h](#).

References [L4::Ipc::Iostream::call\(\)](#), [L4Re::Protocol::Dataspace](#), [EXPECT\\_FALSE](#), [l4\\_error\(\)](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



### 11.34.3.7 long L4Re::Dataspace::size ( ) const throw ()

Get size of a data space.

#### Returns

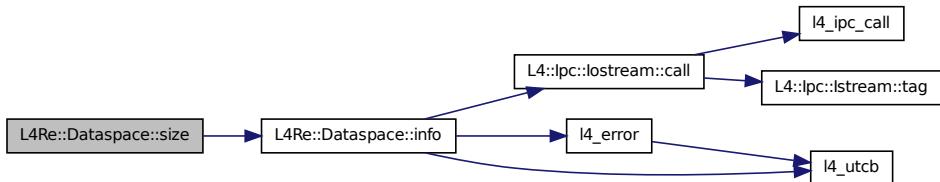
Size of the data space (in bytes), <0 on errors

- IPC errors

Definition at line 135 of file [dataspace\\_impl.h](#).

References [info\(\)](#), and [L4Re::Dataspace::Stats::size](#).

Here is the call graph for this function:



### 11.34.3.8 long L4Re::Dataspace::flags ( ) const throw ()

Get flags of the data space.

#### Returns

Flags of the data space, <0 on errors

- IPC errors

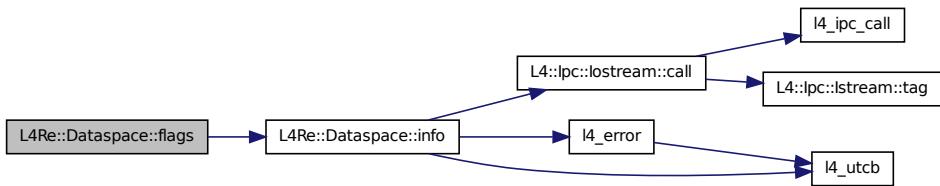
**See also**

[L4Re::Dataspace::Map\\_flags](#)

Definition at line 145 of file `dataspace_impl.h`.

References [L4Re::Dataspace::Stats::flags](#), and [info\(\)](#).

Here is the call graph for this function:

**11.34.3.9 int L4Re::Dataspace::info ( Stats \* stats ) const throw ()**

Get information on the data space.

**Return values**

*info* Data space information,

**See also**

[L4Re::Dataspace::Stats](#)

**Returns**

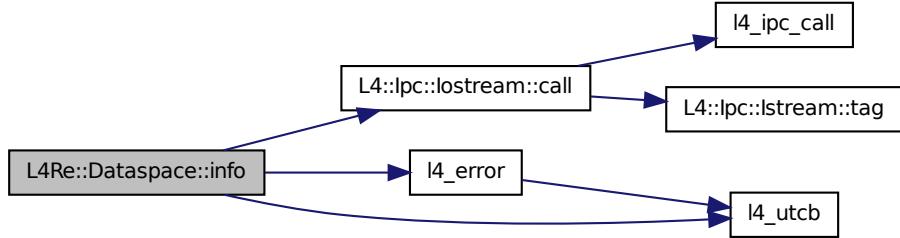
0 on success, < 0 on errors

Definition at line 122 of file `dataspace_impl.h`.

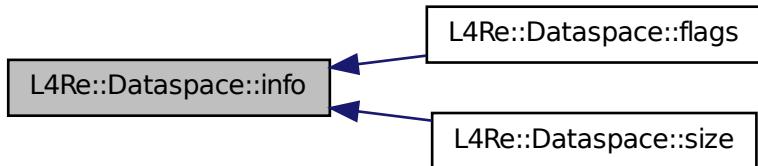
References [L4::Ipc::Iostream::call\(\)](#), [L4Re::Protocol::Dataspace](#), [EXPECT\\_FALSE](#), [l4\\_error\(\)](#), and [l4\\_utcb\(\)](#).

Referenced by [flags\(\)](#), and [size\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- `l4/re/dataspace`
- `l4/re/impl/dataspace_impl.h`

## 11.35 L4Re::Util::Dataspace\_svr Class Reference

[Dataspace](#) server class.

### Public Member Functions

- `int map (l4_addr_t offs, l4_addr_t spot, unsigned long flags, l4_addr_t min, l4_addr_t max, L4::Ipc::Snd_fpage &memory)`  
*Map a region of the dataspace.*
- `virtual int map_hook (l4_addr_t offs, unsigned long flags, l4_addr_t min, l4_addr_t max)`  
*A hook that is called as the first operation in each map request.*

- virtual int `phys` (`l4_addr_t` offset, `l4_addr_t` &`phys_addr`, `l4_size_t` &`phys_size`) throw ()
   
*Return physical address for a virtual address.*
- virtual void `take` () throw ()
   
*Take a reference to this dataspace.*
- virtual unsigned long `release` () throw ()
   
*Release a reference to this dataspace.*
- virtual unsigned long `copy` (unsigned long dst\_offs, `l4_umword_t` src\_id, unsigned long src\_offs, unsigned long size) throw ()
   
*Copy from src dataspace to this destination dataspace.*
- virtual long `clear` (unsigned long offs, unsigned long size) const throw ()
   
*Clear a region in the dataspace.*
- virtual unsigned long `page_shift` () const throw ()
   
*Define the size of the flexpage to map.*

### 11.35.1 Detailed Description

`Dataspace` server class. The default implementation of the interface provides a continuously mapped dataspace.

Definition at line 48 of file `dataspace_svr`.

### 11.35.2 Member Function Documentation

#### 11.35.2.1 int L4Re::Util::Dataspace\_svr::map ( `l4_addr_t` *offs*, `l4_addr_t` *spot*, unsigned long *flags*, `l4_addr_t` *min*, `l4_addr_t` *max*, `L4::Ipc::Snd_fpage` & *memory* )

Map a region of the dataspace.

#### Parameters

- offset*** Offset to start within data space
- flags*** map flags, see `Map_flags`.
- local\_addr*** Local address to map to.
- min\_addr*** Defines start of receive window.
- max\_addr*** Defines end of receive window.

#### Return values

- memory*** Send fpage to map

#### Returns

0 on success, <0 on error

**11.35.2.2 virtual int L4Re::Util::Dataspace\_svr::map\_hook( l4\_addr\_t *offs*, unsigned long *flags*, l4\_addr\_t *min*, l4\_addr\_t *max* ) [inline, virtual]**

A hook that is called as the first operation in each map request.

#### Parameters

*offs* Offs param to map

*flags* Flags param to map

*min* Min param to map

*max* Max param to map

#### Returns

< 0 on error and the map request will be aborted with that error >= 0: ok

#### See also

[map](#)

Definition at line 97 of file [dataspace\\_svr](#).

**11.35.2.3 virtual int L4Re::Util::Dataspace\_svr::phys( l4\_addr\_t *offset*, l4\_addr\_t & *phys\_addr*, l4\_size\_t & *phys\_size* ) throw() [virtual]**

Return physical address for a virtual address.

#### Parameters

*offset* Offset into the dataspace

#### Return values

*phys\_addr* Physical address

*phys\_size* Size of continious physical region

#### Returns

Zero on success, else failure

**11.35.2.4 virtual void L4Re::Util::Dataspace\_svr::take( ) throw() [inline, virtual]**

Take a reference to this dataspace.

Default does nothing.

Definition at line 120 of file [dataspace\\_svr](#).

**11.35.2.5 virtual unsigned long L4Re::Util::Dataspace\_svr::release( ) throw() [inline, virtual]**

Release a reference to this dataspace.

**Returns**

Number of references to the dataspace

Default does nothing and returns always zero.

Definition at line 130 of file [dataspace\\_svr](#).

**11.35.2.6 virtual unsigned long L4Re::Util::Dataspace\_svr::copy ( *unsigned long dst\_offs,*  
  *l4\_umword\_t src\_id, unsigned long src\_offs, unsigned long size* ) throw () [inline,  
  *virtual*]**

Copy from src dataspace to this destination dataspace.

**Parameters**

*dst\_offs* Offset into the destination dataspace

*src\_id* Local id of the source dataspace

*src\_offs* Offset into the source dataspace

*size* Number of bytes to copy

**Returns**

Number of bytes copied

Definition at line 143 of file [dataspace\\_svr](#).

**11.35.2.7 virtual long L4Re::Util::Dataspace\_svr::clear ( *unsigned long offs, unsigned long size* ) const throw () [virtual]**

Clear a region in the dataspace.

**Parameters**

*offs* Start of the region

*size* Size of the region

**11.35.2.8 virtual unsigned long L4Re::Util::Dataspace\_svr::page\_shift ( ) const throw () [inline, virtual]**

Define the size of the flexpage to map.

**Returns**

flexpage size

Definition at line 160 of file [dataspace\\_svr](#).

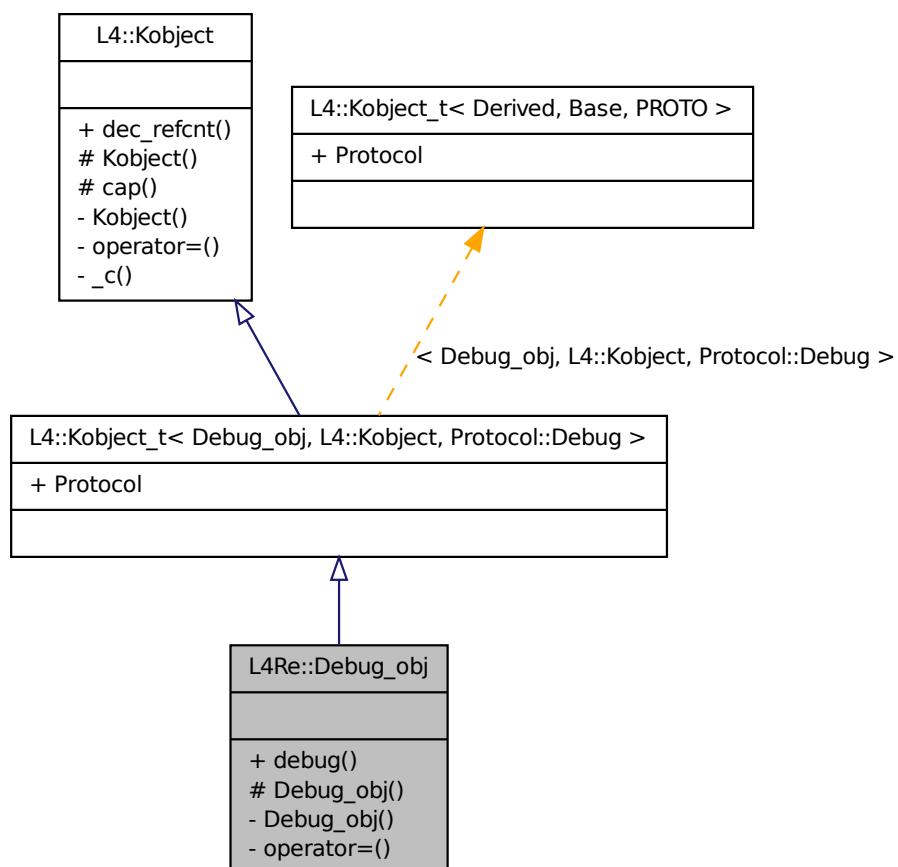
The documentation for this class was generated from the following file:

- l4/re/util/dataspace\_svr

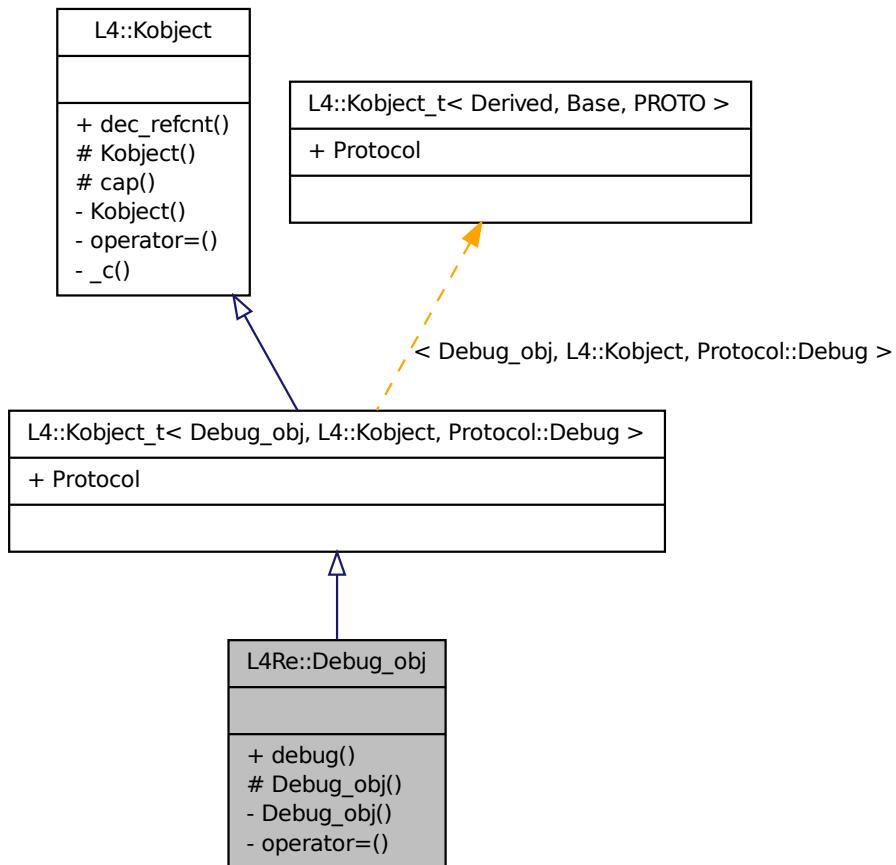
## 11.36 L4Re::Debug\_obj Class Reference

Debug interface.

Inheritance diagram for L4Re::Debug\_obj:



Collaboration diagram for L4Re::Debug\_obj:



## Public Member Functions

- int [debug](#) (unsigned long function) const throw ()  
*Debug call.*

### 11.36.1 Detailed Description

Debug interface.

#### See also

[Debugging API](#).

Definition at line 50 of file [debug](#).

## 11.36.2 Member Function Documentation

### 11.36.2.1 int L4Re::Debug\_obj::debug ( **unsigned long** *function* ) const throw ()

Debug call.

#### Parameters

*function* Function to call.

#### Returns

- L4\_EOK

- IPC errors

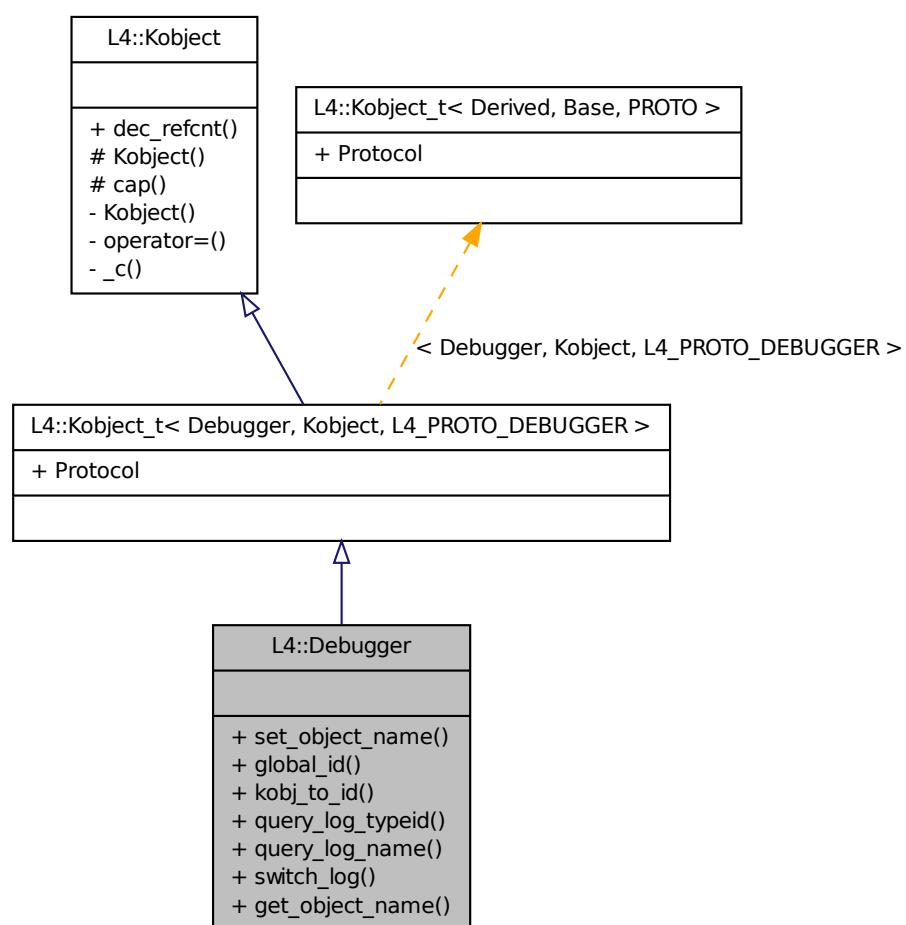
The documentation for this class was generated from the following file:

- l4/re/debug

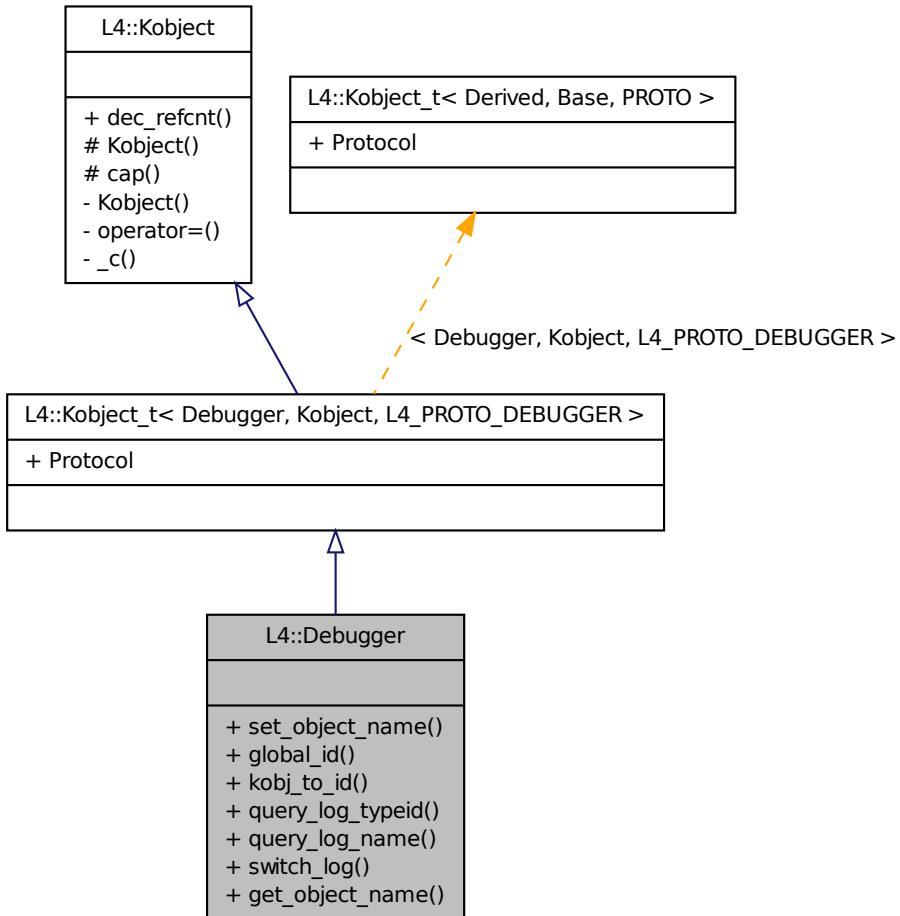
## 11.37 L4::Debugger Class Reference

[Debugger](#) interface.

Inheritance diagram for L4::Debugger:



Collaboration diagram for L4::Debugger:



## Public Member Functions

- `l4_mshtag_t set_object_name` (const char \*name, `l4_utcb_t` \*utcb=`l4_utcb()`) throw ()
- `unsigned long global_id` (`l4_utcb_t` \*utcb=`l4_utcb()`) throw ()
- `unsigned long kobj_to_id` (`l4_addr_t` kobjp, `l4_utcb_t` \*utcb=`l4_utcb()`) throw ()
- `int query_log_typeid` (const char \*name, unsigned idx, `l4_utcb_t` \*utcb=`l4_utcb()`) throw ()
- `int query_log_name` (unsigned idx, char \*name, unsigned namelen, char \*shortname, unsigned shortnamelen, `l4_utcb_t` \*utcb=`l4_utcb()`) throw ()
- `l4_mshtag_t switch_log` (const char \*name, unsigned on\_off, `l4_utcb_t` \*utcb=`l4_utcb()`) throw ()
- `l4_mshtag_t get_object_name` (unsigned id, char \*name, unsigned size, `l4_utcb_t` \*utcb=`l4_utcb()`) throw ()

### 11.37.1 Detailed Description

`Debugger` interface. #include <l4/sys/debugger>

Definition at line 39 of file `debugger`.

### 11.37.2 Member Function Documentation

#### 11.37.2.1 `l4_mshtag_t L4::Debugger::set_object_name ( const char * name, l4_utcb_t * utcb = 14_utcb() ) throw () [inline]`

The string name of kernel object.

##### Parameters

*cap* Capability

*name* Name

This is a debugging facility, the call might be invalid.

##### Note

the *cap* argument is the implicit *this* pointer.

Definition at line 45 of file `debugger`.

#### 11.37.2.2 `unsigned long L4::Debugger::global_id ( l4_utcb_t * utcb = 14_utcb() ) throw () [inline]`

Get the globally unique ID of the object behind a capability.

##### Parameters

*cap* Capability

##### Returns

~0UL on non-valid capability, ID otherwise

This is a debugging facility, the call might be invalid.

##### Note

the *cap* argument is the implicit *this* pointer.

Definition at line 53 of file `debugger`.

#### 11.37.2.3 `unsigned long L4::Debugger::kobj_to_id ( l4_addr_t kobjp, l4_utcb_t * utcb = 14_utcb() ) throw () [inline]`

Get the globally unique ID of the object behind the kobject pointer.

##### Parameters

*cap* Capability

*kobjp* Kobject pointer

### Returns

~0UL on non-valid capability or invalid kobject pointer, ID otherwise

This is a debugging facility, the call might be invalid.

### Note

the *cap* argument is the implicit *this* pointer.

Definition at line 60 of file [debugger](#).

**11.37.2.4 int L4::Debugger::query\_log\_typeid ( const char \* *name*, unsigned *idx*, l4\_utcb\_t \* *utcb* = *L4\_utcb()* ) throw () [inline]**

### Note

the *cap* argument is the implicit *this* pointer.

Definition at line 68 of file [debugger](#).

**11.37.2.5 int L4::Debugger::query\_log\_name ( unsigned *idx*, char \* *name*, unsigned *namelen*, char \* *shortname*, unsigned *shortnamelen*, l4\_utcb\_t \* *utcb* = *L4\_utcb()* ) throw () [inline]**

### Note

the *cap* argument is the implicit *this* pointer.

Definition at line 76 of file [debugger](#).

**11.37.2.6 l4\_mshtag\_t L4::Debugger::switch\_log ( const char \* *name*, unsigned *on\_off*, l4\_utcb\_t \* *utcb* = *L4\_utcb()* ) throw () [inline]**

### Note

the *cap* argument is the implicit *this* pointer.

Definition at line 89 of file [debugger](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



**11.37.2.7 l4\_mshtag\_t L4::Debugger::get\_object\_name ( `unsigned id, char * name, unsigned size, l4_utcb_t * utcb = l4_utcb ()` ) throw () [inline]**

#### Note

the `cap` argument is the implicit *this* pointer.

Definition at line 97 of file [debugger](#).

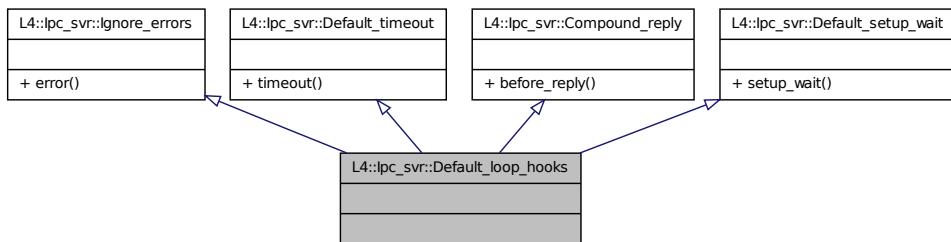
The documentation for this class was generated from the following file:

- [l4/sys/debugger](#)

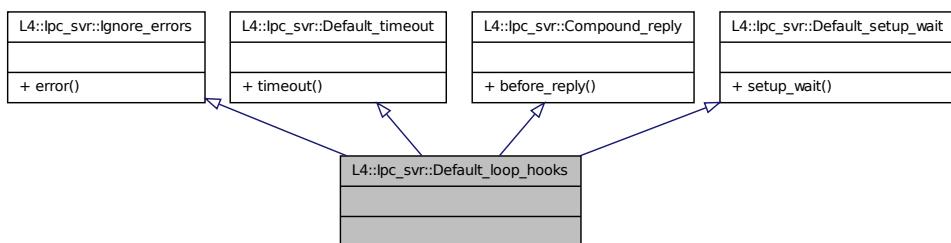
## 11.38 L4::Ipc\_svr::Default\_loop\_hooks Struct Reference

Default LOOP\_HOOKS.

Inheritance diagram for L4::Ipc\_svr::Default\_loop\_hooks:



Collaboration diagram for L4::Ipc\_svr::Default\_loop\_hooks:



### 11.38.1 Detailed Description

Default LOOP\_HOOKS. Combination of [Ignore\\_errors](#), [Default\\_timeout](#), [Compound\\_reply](#), and [Default\\_setup\\_wait](#).

Definition at line 83 of file [ipc\\_server](#).

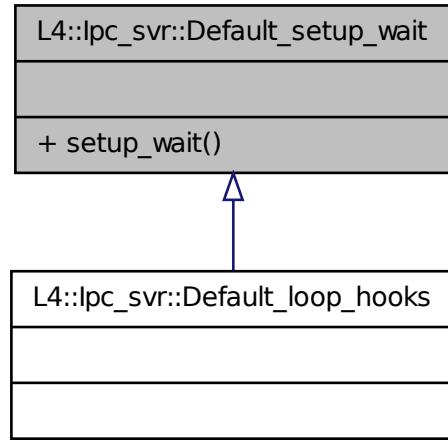
The documentation for this struct was generated from the following file:

- l4/cxx/ipc\_server

## 11.39 L4::Ipc\_svr::Default\_setup\_wait Struct Reference

Mix in for LOOP\_HOOKS for setup\_wait no op.

Inheritance diagram for L4::Ipc\_svr::Default\_setup\_wait:



### 11.39.1 Detailed Description

Mix in for LOOP\_HOOKS for setup\_wait no op.

Definition at line [74](#) of file [ipc\\_server](#).

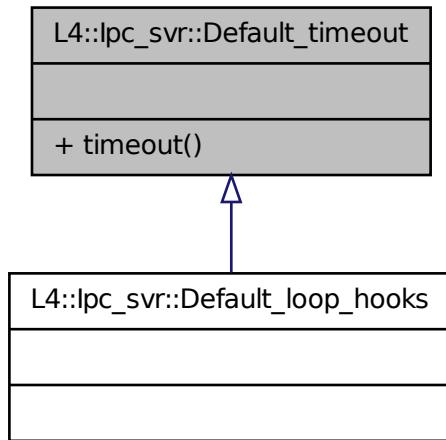
The documentation for this struct was generated from the following file:

- l4/cxx/ipc\_server

## 11.40 L4::Ipc\_svr::Default\_timeout Struct Reference

Mix in for LOOP\_HOOKS to use a 0 send and a infinite receive timeout.

Inheritance diagram for L4::Ipc\_svr::Default\_timeout:



### 11.40.1 Detailed Description

Mix in for LOOP\_HOOKS to use a 0 send and a infinite receive timeout.

Definition at line 59 of file [ipc\\_server](#).

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

- [l4/cxx/ipc\\_server](#)

## 11.41 cxx::Bits::Direction Struct Reference

The direction to go in a binary search tree.

```
#include <bst_base.h>
```

### Public Types

- enum `Direction_e` { `L` = 0, `R` = 1, `N` = 2 }

*The literal direction values.*

### Public Member Functions

- `Direction ()`

*Uninitialized direction.*

- **Direction** (`Direction_e d`)  
*Convert a literal direction (`L`, `R`, `N`) to an object.*
- **Direction** (`bool b`)  
*Convert a boolean to a direction (`false == L`, `true == R`).*
- **Direction operator!** () const  
*Negate the direction.*

### Comparison operators (equality and inequality)

- `bool operator==(Direction_e o) const`
- `bool operator!=(Direction_e o) const`
- `bool operator==(Direction o) const`
- `bool operator!=(Direction o) const`

#### 11.41.1 Detailed Description

The direction to go in a binary search tree.

Definition at line 39 of file `bst_base.h`.

#### 11.41.2 Member Enumeration Documentation

##### 11.41.2.1 enum cxx::Bits::Direction::Direction\_e

The literal direction values.

###### Enumerator:

- L** Go to the left child.
- R** Go to the right child.
- N** Stop.

Definition at line 42 of file `bst_base.h`.

#### 11.41.3 Member Function Documentation

##### 11.41.3.1 Direction cxx::Bits::Direction::operator! () const [inline]

Negate the direction.

###### Note

This is only defined for a current value of `L` or `R`

Definition at line 63 of file `bst_base.h`.

References `Direction()`.

Here is the call graph for this function:



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

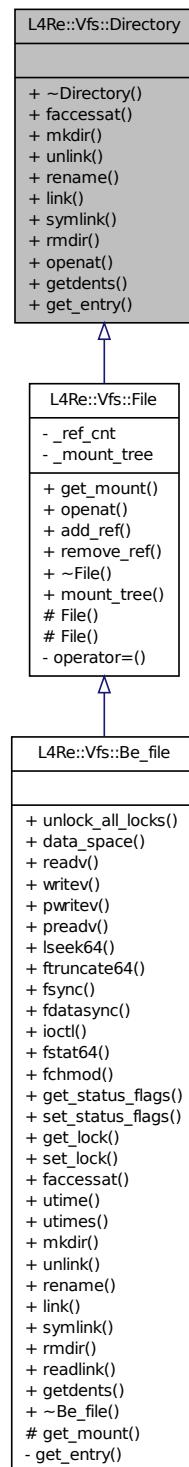
- 14/cxx/bits/bst\_base.h

## 11.42 L4Re::Vfs::Directory Class Reference

Interface for a POSIX file that is a directory.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Directory:



## Public Member Functions

- virtual int [faccessat](#) (const char \*path, int mode, int flags)=0 throw ()
 

*Check access permissions on the given file.*
- virtual int [mkdir](#) (const char \*path, mode\_t mode)=0 throw ()
 

*Create a new subdirectory.*
- virtual int [unlink](#) (const char \*path)=0 throw ()
 

*Unlink the given file from that directory.*
- virtual int [rename](#) (const char \*src\_path, const char \*dst\_path)=0 throw ()
 

*Rename the given file.*
- virtual int [link](#) (const char \*src\_path, const char \*dst\_path)=0 throw ()
 

*Create a hard link (second name) for the given file.*
- virtual int [symlink](#) (const char \*src\_path, const char \*dst\_path)=0 throw ()
 

*Create a symbolic link for the given file.*
- virtual int [rmdir](#) (const char \*)=0 throw ()
 

*Delete an empty directory.*

### 11.42.1 Detailed Description

Interface for a POSIX file that is a directory. This interface provides functionality for directory files in the [L4Re::Vfs](#). However, real objects use always the combined [L4Re::Vfs::File](#) interface.

Definition at line 140 of file [vfs.h](#).

### 11.42.2 Member Function Documentation

#### 11.42.2.1 virtual int L4Re::Vfs::Directory::faccessat ( const char \* path, int mode, int flags ) throw () [pure virtual]

Check access permissions on the given file.

Backend function for POSIX access and faccessat functions.

#### Parameters

**path** The path relative to this directory. Note: *path* is relative to this directory and may contain subdirectories.

**mode** The access mode to check.

**flags** The flags as in POSIX faccessat (AT\_EACCESS, AT\_SYMLINK\_NOFOLLOW).

#### Returns

0 on success, or <0 on error.

**11.42.2.2 virtual int L4Re::Vfs::Directory::mkdir ( const char \* *path*, mode\_t *mode* ) throw () [pure virtual]**

Create a new subdirectory.

Backend for POSIX mkdir and mkdirat function calls.

**Parameters**

*path* The name of the subdirectory to create. Note: *path* is relative to this directory and may contain subdirectories.

*mode* The file mode to use for the new directory.

**Returns**

0 on success, or <0 on error. -ENOTDIR if this or some component in path is not a directory.

**11.42.2.3 virtual int L4Re::Vfs::Directory::unlink ( const char \* *path* ) throw () [pure virtual]**

Unlink the given file from that directory.

Backend for the POSIX unlink and unlinkat functions.

**Parameters**

*path* The name to the file to unlink. Note: *path* is relative to this directory and may contain subdirectories.

**Returns**

0 on success, or <0 on error.

**11.42.2.4 virtual int L4Re::Vfs::Directory::rename ( const char \* *src\_path*, const char \* *dst\_path* ) throw () [pure virtual]**

Rename the given file.

Backend for the POSIX rename, renameat functions.

**Parameters**

*src\_path* The old name to the file to rename. Note: *src\_path* is relative to this directory and may contain subdirectories.

*dst\_path* The new name for the file. Note: *dst\_path* is relative to this directory and may contain subdirectories.

**Returns**

0 on success, or <0 on error.

**11.42.2.5 virtual int L4Re::Vfs::Directory::link ( const char \* *src\_path*, const char \* *dst\_path* ) throw () [pure virtual]**

Create a hard link (second name) for the given file.

Backend for the POSIX link and linkat functions.

**Parameters**

*src\_path* The old name to the file. Note: *src\_path* is relative to this directory and may contain subdirectories.

*dst\_path* The new (second) name for the file. Note: *dst\_path* is relative to this directory and may contain subdirectories.

**Returns**

0 on success, or <0 on error.

**11.42.2.6 virtual int L4Re::Vfs::Directory::symlink ( const char \* *src\_path*, const char \* *dst\_path* ) throw () [pure virtual]**

Create a symbolic link for the given file.

Backend for the POSIX symlink and symlinkat functions.

**Parameters**

*src\_path* The old name to the file. Note: *src\_path* shall be an absolute path.

*dst\_path* The name for symlink. Note: *dst\_path* is relative to this directory and may contain subdirectories.

**Returns**

0 on success, or <0 on error.

**11.42.2.7 virtual int L4Re::Vfs::Directory::rmdir ( const char \* ) throw () [pure virtual]**

Delete an empty directory.

Backend for POSIX rmdir, rmdirat functions.

**Parameters**

*path* The name of the directory to remove. Note: *path* is relative to this directory and may contain subdirectories.

**Returns**

0 on success, or <0 on error.

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

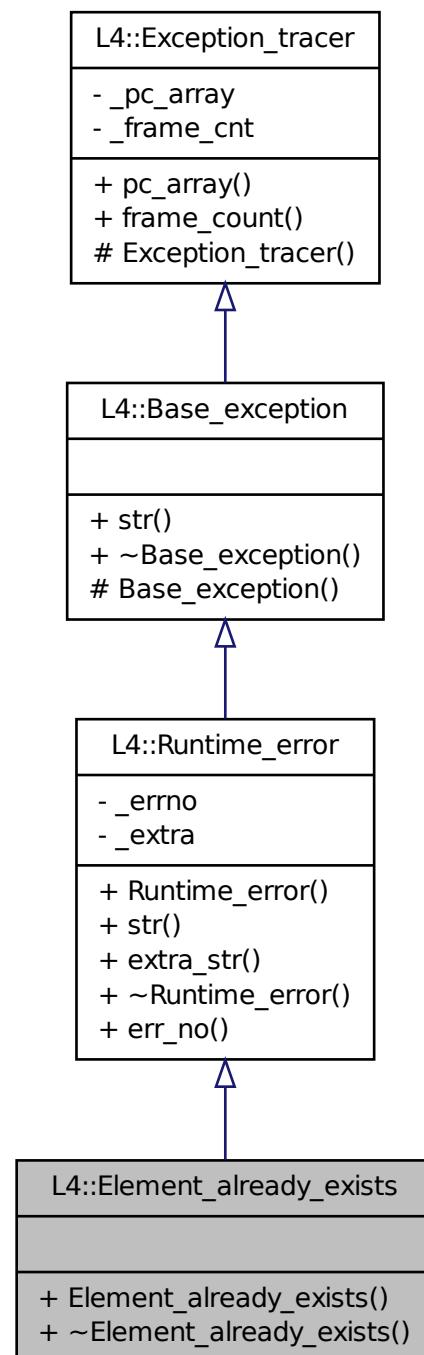
- l4/l4re\_vfs/vfs.h

## 11.43 L4::Element\_already\_exists Class Reference

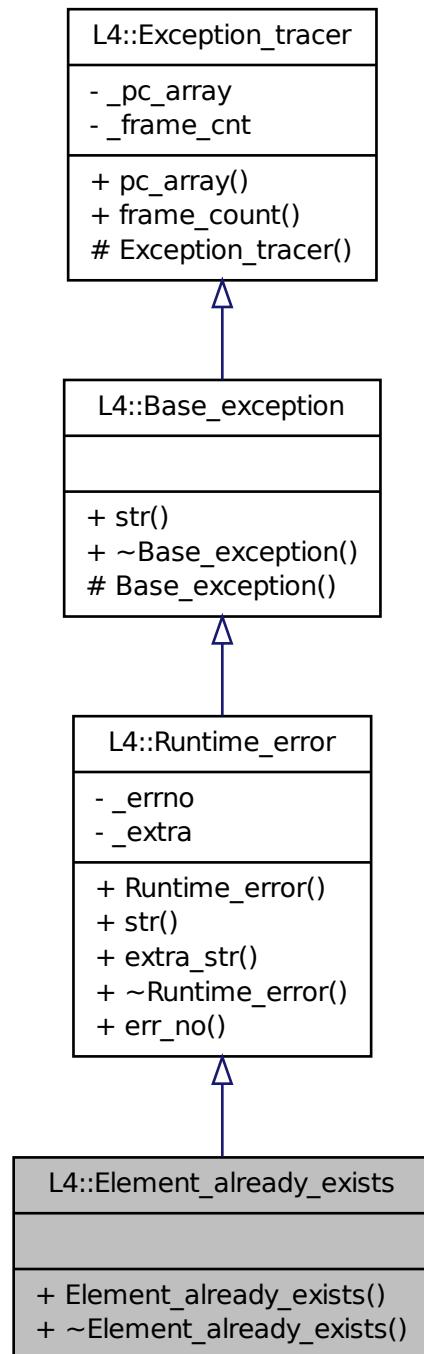
Exception for duplicate element insertions.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Element\_already\_exists:



Collaboration diagram for L4::Element\_already\_exists:



### **11.43.1 Detailed Description**

Exception for duplicate element insertions.

Definition at line [177](#) of file [exceptions](#).

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

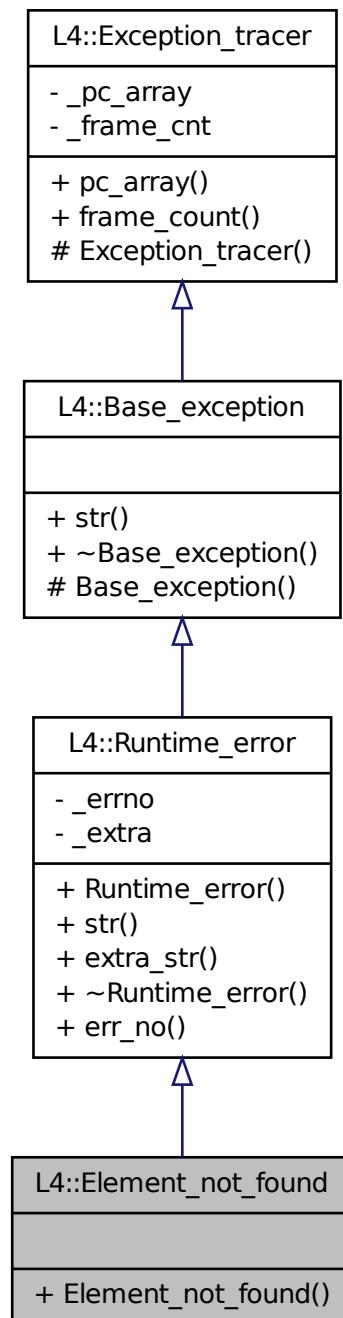
- [l4/cxx/exceptions](#)

## **11.44 L4::Element\_not\_found Class Reference**

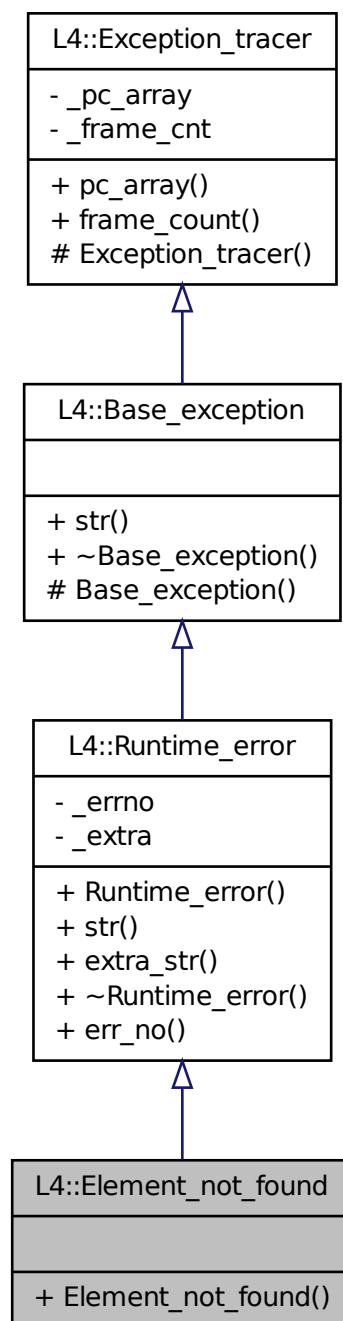
Exception for a failed lookup (element not found).

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Element\_not\_found:



Collaboration diagram for L4::Element\_not\_found:



### 11.44.1 Detailed Description

Exception for a failed lookup (element not found).

Definition at line 206 of file [exceptions](#).

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

- l4/cxx/exceptions

## 11.45 Elf32\_Dyn Struct Reference

ELF32 dynamic entry.

```
#include <elf.h>
```

### Data Fields

- [Elf32\\_Sword d\\_tag](#)  
*see DT\_values*
- [Elf32\\_Word d\\_val](#)  
*integer values with various interpret.*
- [Elf32\\_Addr d\\_ptr](#)  
*program virtual addresses*

### 11.45.1 Detailed Description

ELF32 dynamic entry.

Definition at line 459 of file [elf.h](#).

### 11.45.2 Field Documentation

#### 11.45.2.1 Elf32\_Word Elf32\_Dyn::d\_val

integer values with various interpret.

Definition at line 462 of file [elf.h](#).

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

- l4/util/elf.h

## 11.46 Elf32\_Ehdr Struct Reference

ELF32 header.

```
#include <elf.h>
```

## Data Fields

- `Elf32_Half e_type`  
*type of ELF file*
- `Elf32_Half e_machine`  
*required architecture*
- `Elf32_Word e_version`  
*file version*
- `Elf32_Addr e_entry`  
*initial eip*
- `Elf32_Off e_phoff`  
*offset of program header table*
- `Elf32_Off e_shoff`  
*offset of file header table*
- `Elf32_Word e_flags`  
*processor-specific flags*
- `Elf32_Half e_ehsize`  
*size of ELF header*
- `Elf32_Half e_phentsize`  
*size of program header entry*
- `Elf32_Half e_phnum`  
*# of entries in prog.*
- `Elf32_Half e_shentsize`  
*size of section header entry*
- `Elf32_Half e_shnum`  
*# of entries in sect.*
- `Elf32_Half e_shstrndx`  
*sect.head.tab.idx of strtab*

### 11.46.1 Detailed Description

ELF32 header.

Definition at line 118 of file `elf.h`.

## 11.46.2 Field Documentation

### 11.46.2.1 Elf32\_Half Elf32\_Ehdr::e\_phnum

# of entries in prog.

head. tab.

Definition at line 129 of file [elf.h](#).

### 11.46.2.2 Elf32\_Half Elf32\_Ehdr::e\_shnum

# of entries in sect.

head. tab.

Definition at line 131 of file [elf.h](#).

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

- 14/util/elf.h

## 11.47 Elf32\_Phdr Struct Reference

ELF32 program header.

```
#include <elf.h>
```

### Data Fields

- [Elf32\\_Word p\\_type](#)  
*type of program section*
- [Elf32\\_Off p\\_offset](#)  
*file offset of program section*
- [Elf32\\_Addr p\\_vaddr](#)  
*memory address of prog section*
- [Elf32\\_Addr p\\_paddr](#)  
*physical address (ignored)*
- [Elf32\\_Word p\\_filesz](#)  
*file size of program section*
- [Elf32\\_Word p\\_memsz](#)  
*memory size of program section*
- [Elf32\\_Word p\\_flags](#)  
*flags*
- [Elf32\\_Word p\\_align](#)

*alignment of section*

### 11.47.1 Detailed Description

ELF32 program header.

Definition at line 378 of file [elf.h](#).

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

- l4/util/elf.h

## 11.48 Elf32\_Shdr Struct Reference

ELF32 section header - figure 1-9, page 1-9.

```
#include <elf.h>
```

### Data Fields

- [Elf32\\_Word sh\\_name](#)  
*name of sect (idx into shstrtab)*
- [Elf32\\_Word sh\\_type](#)  
*section's type*
- [Elf32\\_Word sh\\_flags](#)  
*section's flags*
- [Elf32\\_Addr sh\\_addr](#)  
*memory address of section*
- [Elf32\\_Off sh\\_offset](#)  
*file offset of section*
- [Elf32\\_Word sh\\_size](#)  
*file size of section*
- [Elf32\\_Word sh\\_link](#)  
*idx to associated header section*
- [Elf32\\_Word sh\\_info](#)  
*extra info of header section*
- [Elf32\\_Word sh\\_addralign](#)  
*address alignment constraints*
- [Elf32\\_Word sh\\_entsize](#)  
*size of entry if sect is table*

### 11.48.1 Detailed Description

ELF32 section header - figure 1-9, page 1-9.

Definition at line 302 of file [elf.h](#).

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

- l4/util/elf.h

## 11.49 Elf32\_Sym Struct Reference

ELF32 symbol table entry.

```
#include <elf.h>
```

### Data Fields

- [Elf32\\_Word st\\_name](#)  
*name of symbol (idx symstrtab)*
- [Elf32\\_Addr st\\_value](#)  
*value of associated symbol*
- [Elf32\\_Word st\\_size](#)  
*size of associated symbol*
- unsigned char [st\\_info](#)  
*type and binding info*
- unsigned char [st\\_other](#)  
*undefined*
- [Elf32\\_Half st\\_shndx](#)  
*associated section header*

### 11.49.1 Detailed Description

ELF32 symbol table entry.

Definition at line 759 of file [elf.h](#).

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

- l4/util/elf.h

## 11.50 Elf64\_Dyn Struct Reference

ELF64 dynamic entry.

```
#include <elf.h>
```

## Data Fields

- [Elf64\\_Sxword d\\_tag](#)  
*see DT\_values*
- [Elf64\\_Xword d\\_val](#)  
*integer values with various interpret.*
- [Elf64\\_Addr d\\_ptr](#)  
*program virtual addresses*

### 11.50.1 Detailed Description

ELF64 dynamic entry.

Definition at line 468 of file [elf.h](#).

### 11.50.2 Field Documentation

#### 11.50.2.1 Elf64\_Xword Elf64\_Dyn::d\_val

integer values with various interpret.

Definition at line 471 of file [elf.h](#).

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

- l4/util/elf.h

## 11.51 Elf64\_Ehdr Struct Reference

ELF64 header.

```
#include <elf.h>
```

## Data Fields

- [Elf64\\_Half e\\_type](#)  
*type of ELF file*
- [Elf64\\_Half e\\_machine](#)  
*required architecture*
- [Elf64\\_Word e\\_version](#)  
*file version*
- [Elf64\\_Addr e\\_entry](#)  
*initial eip*

- [Elf64\\_Off e\\_phoff](#)  
*offset of program header table*
- [Elf64\\_Off e\\_shoff](#)  
*offset of file header table*
- [Elf64\\_Word e\\_flags](#)  
*processor-specific flags*
- [Elf64\\_Half e\\_ehsize](#)  
*size of ELF header*
- [Elf64\\_Half e\\_phentsize](#)  
*size of program header entry*
- [Elf64\\_Half e\\_phnum](#)  
*# of entries in prog.*
- [Elf64\\_Half e\\_shentsize](#)  
*size of section header entry*
- [Elf64\\_Half e\\_shnum](#)  
*# of entries in sect.*
- [Elf64\\_Half e\\_shstrndx](#)  
*sect.head.tab.idx of strtab*

### 11.51.1 Detailed Description

ELF64 header.

Definition at line [138](#) of file [elf.h](#).

### 11.51.2 Field Documentation

#### 11.51.2.1 Elf64\_Half Elf64\_Ehdr::e\_phnum

# of entries in prog.

head. tab.

Definition at line [149](#) of file [elf.h](#).

#### 11.51.2.2 Elf64\_Half Elf64\_Ehdr::e\_shnum

# of entries in sect.

head. tab.

Definition at line [151](#) of file [elf.h](#).

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

- l4/util/elf.h

## 11.52 Elf64\_Phdr Struct Reference

ELF64 program header.

```
#include <elf.h>
```

### Data Fields

- [Elf64\\_Word p\\_type](#)  
*type of program section*
- [Elf64\\_Word p\\_flags](#)  
*flags*
- [Elf64\\_Off p\\_offset](#)  
*file offset of program section*
- [Elf64\\_Addr p\\_vaddr](#)  
*memory address of prog section*
- [Elf64\\_Addr p\\_paddr](#)  
*physical address (ignored)*
- [Elf64\\_Xword p\\_filesz](#)  
*file size of program section*
- [Elf64\\_Xword p\\_memsz](#)  
*memory size of program section*
- [Elf64\\_Xword p\\_align](#)  
*alignment of section*

### 11.52.1 Detailed Description

ELF64 program header.

Definition at line 390 of file [elf.h](#).

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

- l4/util/elf.h

## 11.53 Elf64\_Shdr Struct Reference

ELF64 section header.

```
#include <elf.h>
```

## Data Fields

- `Elf64_Word sh_name`  
*name of sect (idx into strtab)*
- `Elf64_Word sh_type`  
*section's type*
- `Elf64_Xword sh_flags`  
*section's flags*
- `Elf64_Addr sh_addr`  
*memory address of section*
- `Elf64_Off sh_offset`  
*file offset of section*
- `Elf64_Xword sh_size`  
*file size of section*
- `Elf64_Word sh_link`  
*idx to associated header section*
- `Elf64_Word sh_info`  
*extra info of header section*
- `Elf64_Xword sh_addralign`  
*address alignment constraints*
- `Elf64_Xword sh_entsize`  
*size of entry if sect is table*

### 11.53.1 Detailed Description

ELF64 section header.

Definition at line 316 of file `elf.h`.

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

- 14/util/elf.h

## 11.54 Elf64\_Sym Struct Reference

ELF64 symbol table entry.

```
#include <elf.h>
```

## Data Fields

- [Elf64\\_Word st\\_name](#)

*name of symbol (idx symstrtab)*

- [unsigned char st\\_info](#)

*type and binding info*

- [unsigned char st\\_other](#)

*undefined*

- [Elf64\\_Half st\\_shndx](#)

*associated section header*

- [Elf64\\_Addr st\\_value](#)

*value of associated symbol*

- [Elf64\\_Xword st\\_size](#)

*size of associated symbol*

### 11.54.1 Detailed Description

ELF64 symbol table entry.

Definition at line [769](#) of file [elf.h](#).

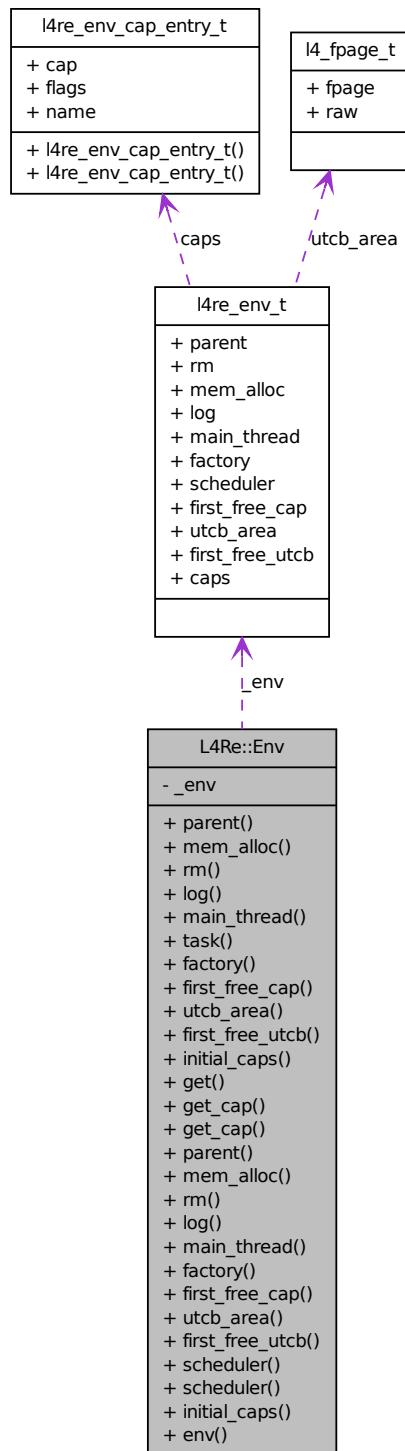
The documentation for this struct was generated from the following file:

- l4/util/elf.h

## 11.55 L4Re::Env Class Reference

Initial Environment (C++ version).

Collaboration diagram for L4Re::Env:



## Public Types

- `typedef l4re_env_cap_entry_t Cap_entry`  
*C++ type for an entry in the initial objects array.*

## Public Member Functions

- `L4::Cap< Parent > parent () const throw ()`  
*Object-capability to the parent.*
- `L4::Cap< Mem_alloc > mem_alloc () const throw ()`  
*Object-capability to the memory allocator.*
- `L4::Cap< Rm > rm () const throw ()`  
*Object-capability to the region map.*
- `L4::Cap< Log > log () const throw ()`  
*Object-capability to the logging service.*
- `L4::Cap< L4::Thread > main_thread () const throw ()`  
*Object-capability of the first user thread.*
- `L4::Cap< L4::Task > task () const throw ()`  
*Object-capability of the user task.*
- `L4::Cap< L4::Factory > factory () const throw ()`  
*Object-capability to the factory object available to the task.*
- `l4_cap_idx_t first_free_cap () const throw ()`  
*First available capability selector.*
- `l4_fpage_t utcb_area () const throw ()`  
*UTCB area of the task.*
- `l4_addr_t first_free_utcb () const throw ()`  
*First free UTCB.*
- `Cap_entry const * initial_caps () const throw ()`  
*Get a pointer to the first entry in the initial objects array.*
- `Cap_entry const * get (char const *name, unsigned l) const throw ()`  
*Get the Cap\_entry for the object named name.*
- `template<typename T > L4::Cap< T > get_cap (char const *name, unsigned l) const throw ()`  
*Get the capability selector for the object named name.*
- `template<typename T > L4::Cap< T > get_cap (char const *name) const throw ()`

*Get the capability selector for the object named name.*

- void `parent` (`L4::Cap< Parent > const &c`) throw ()
 

*Set parent object-capability.*
- void `mem_alloc` (`L4::Cap< Mem_alloc > const &c`) throw ()
 

*Set memory allocator object-capability.*
- void `rm` (`L4::Cap< Rm > const &c`) throw ()
 

*Set region map object-capability.*
- void `log` (`L4::Cap< Log > const &c`) throw ()
 

*Set log object-capability.*
- void `main_thread` (`L4::Cap< L4::Thread > const &c`) throw ()
 

*Set object-capability of first user thread.*
- void `factory` (`L4::Cap< L4::Factory > const &c`) throw ()
 

*Set factory object-capability.*
- void `first_free_cap` (`l4_cap_idx_t c`) throw ()
 

*Set first available capability selector.*
- void `utcbs_area` (`l4_fpage_t utcbs`) throw ()
 

*Set UTCB area of the task.*
- void `first_free_utcb` (`l4_addr_t u`) throw ()
 

*Set first free UTCB.*
- `L4::Cap< L4::Scheduler > scheduler` () const throw ()
 

*Get the scheduler capability for the task.*
- void `scheduler` (`L4::Cap< L4::Scheduler > const &c`) throw ()
 

*Set the scheduler capability.*
- void `initial_caps` (`Cap_entry *first`) throw ()
 

*Set the pointer to the first Cap\_entry in the initial objects array.*

## Static Public Member Functions

- static `Env const * env` () throw ()
 

*Returns the initial environment for the current task.*

### 11.55.1 Detailed Description

Initial Environment (C++ version). This class provides an initial set of capabilities as well as information the first free UTCB and used capability slots.

#### See also

[Initial environment](#)

Definition at line [85](#) of file [env](#).

### 11.55.2 Member Function Documentation

#### 11.55.2.1 static Env const\* L4Re::Env::env( ) throw() [inline, static]

Returns the initial environment for the current task.

#### Returns

Pointer to the initial environment class.

A typical use of this function is `L4Re::Env::env()-><member>()`

#### Examples:

`examples/clntsrv/client.cc`, `examples/libs/l4re/c++/mem_alloc/ma+rm.cc`,  
`examples/libs/l4re/c++/shared_ds/ds_clnt.cc`, `examples/libs/l4re/c++/shared_ds/ds_srv.cc`, `examples/libs/l4re/streammap/client.cc`, and `examples/sys/migrate/thread_migrate.cc`.

Definition at line [103](#) of file [env](#).

#### 11.55.2.2 L4::Cap<Parent> L4Re::Env::parent( ) const throw() [inline]

Object-capability to the parent.

#### Returns

`Parent` object-capability

Definition at line [110](#) of file [env](#).

#### 11.55.2.3 L4::Cap<Mem\_alloc> L4Re::Env::mem\_alloc( ) const throw() [inline]

Object-capability to the memory allocator.

#### Returns

Memory allocator object-capability

Definition at line [116](#) of file [env](#).

**11.55.2.4 L4::Cap<Rm> L4Re::Env::rm( ) const throw() [inline]**

Object-capability to the region map.

**Returns**

Region map object-capability

Definition at line 122 of file [env](#).

**11.55.2.5 L4::Cap<Log> L4Re::Env::log( ) const throw() [inline]**

Object-capability to the logging service.

**Returns**

[Log](#) object-capability

Definition at line 128 of file [env](#).

**11.55.2.6 L4::Cap<L4::Thread> L4Re::Env::main\_thread( ) const throw() [inline]**

Object-capability of the first user thread.

**Returns**

Object-capability of the first user thread.

Definition at line 134 of file [env](#).

**11.55.2.7 L4::Cap<L4::Task> L4Re::Env::task( ) const throw() [inline]**

Object-capability of the user task.

**Returns**

Object-capability of the user task.

Definition at line 140 of file [env](#).

**11.55.2.8 L4::Cap<L4::Factory> L4Re::Env::factory( ) const throw() [inline]**

Object-capability to the factory object available to the task.

**Returns**

Factory object-capability

Definition at line 146 of file [env](#).

**11.55.2.9 l4\_cap\_idx\_t L4Re::Env::first\_free\_cap( ) const throw() [inline]**

First available capability selector.

**Returns**

First capability selector.

First capability selector available for use for in the application.

Definition at line 154 of file [env](#).

**11.55.2.10 l4\_fpage\_t L4Re::Env::utcb\_area( ) const throw() [inline]**

UTCB area of the task.

**Returns**

UTCB area

Definition at line 160 of file [env](#).

**11.55.2.11 l4\_addr\_t L4Re::Env::first\_free\_utcb( ) const throw() [inline]**

First free UTCB.

**Returns**

object-capability

First free UTCB within the UTCB area available for the application to use.

Definition at line 169 of file [env](#).

**11.55.2.12 Cap\_entry const\* L4Re::Env::initial\_caps( ) const throw() [inline]**

Get a pointer to the first entry in the initial objects array.

**Returns**

A pointer to the first entry in the initial objects array.

Definition at line 176 of file [env](#).

**11.55.2.13 Cap\_entry const\* L4Re::Env::get( char const \* name, unsigned l ) const throw() [inline]**

Get the Cap\_entry for the object named *name*.

**Parameters**

*name* is the name of the object.

*l* is the length of the name, thus *name* might mot be zero terminated.

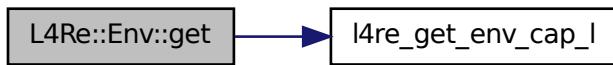
## Returns

A pointer to the Cap\_entry for the object named *name*, or NULL if no such object was found.

Definition at line 187 of file [env](#).

References [l4re\\_get\\_env\\_cap\\_l\(\)](#).

Here is the call graph for this function:



**11.55.2.14 template<typename T > L4::Cap<T> L4Re::Env::get\_cap ( char const \* name, unsigned l ) const throw () [inline]**

Get the capability selector for the object named *name*.

## Parameters

*name* is the name of the object.

*l* is the length of the name, thus *name* might not be zero terminated.

## Returns

A capability selector for the object named *name*, or an invalid capability selector if no such object was found.

Definition at line 199 of file [env](#).

**11.55.2.15 template<typename T > L4::Cap<T> L4Re::Env::get\_cap ( char const \* name ) const throw () [inline]**

Get the capability selector for the object named *name*.

## Parameters

*name* is the name of the object (zero terminated).

## Returns

A capability selector for the object named *name*, or an invalid capability selector if no such object was found.

Definition at line 214 of file [env](#).

**11.55.2.16 void L4Re::Env::parent ( L4::Cap< Parent > const & c ) throw () [inline]**

Set parent object-capability.

**Parameters**

*c* Parent object-capability

Definition at line 221 of file [env](#).

**11.55.2.17 void L4Re::Env::mem\_alloc ( L4::Cap< Mem\_alloc > const & c ) throw () [inline]**

Set memory allocator object-capability.

**Parameters**

*c* Memory allocator object-capability

Definition at line 227 of file [env](#).

**11.55.2.18 void L4Re::Env::rm ( L4::Cap< Rm > const & c ) throw () [inline]**

Set region map object-capability.

**Parameters**

*c* Region map object-capability

Definition at line 233 of file [env](#).

**11.55.2.19 void L4Re::Env::log ( L4::Cap< Log > const & c ) throw () [inline]**

Set log object-capability.

**Parameters**

*c* Log object-capability

Definition at line 239 of file [env](#).

**11.55.2.20 void L4Re::Env::main\_thread ( L4::Cap< L4::Thread > const & c ) throw () [inline]**

Set object-capability of first user thread.

**Parameters**

*c* First thread's object-capability

Definition at line 245 of file [env](#).

**11.55.2.21 void L4Re::Env::factory ( L4::Cap< L4::Factory > const & *c* ) throw () [inline]**

Set factory object-capability.

**Parameters**

*c* Factory object-capability

Definition at line 251 of file [env](#).

**11.55.2.22 void L4Re::Env::first\_free\_cap ( l4\_cap\_idx\_t *c* ) throw () [inline]**

Set first available capability selector.

**Parameters**

*c* First capability selector available to the application.

Definition at line 257 of file [env](#).

**11.55.2.23 void L4Re::Env::utcb\_area ( l4\_fpage\_t *utcbs* ) throw () [inline]**

Set UTCB area of the task.

**Parameters**

*utcbs* UTCB area

Definition at line 263 of file [env](#).

**11.55.2.24 void L4Re::Env::first\_free\_utcb ( l4\_addr\_t *u* ) throw () [inline]**

Set first free UTCB.

**Parameters**

*u* First UTCB available for the application to use.

Definition at line 269 of file [env](#).

**11.55.2.25 L4::Cap<L4::Scheduler> L4Re::Env::scheduler ( ) const throw () [inline]**

Get the scheduler capability for the task.

**Returns**

The capability selector for the default scheduler used for this task.

Definition at line 277 of file [env](#).

---

**11.55.2.26 void L4Re::Env::scheduler ( L4::Cap< L4::Scheduler > const & *c* ) throw () [inline]**

Set the scheduler capability.

#### Parameters

*c* is the capability to be set as scheduler.

Definition at line 284 of file [env](#).

**11.55.2.27 void L4Re::Env::initial\_caps ( Cap\_entry \* *first* ) throw () [inline]**

Set the pointer to the first Cap\_entry in the initial objects array.

#### Parameters

*first* is the first element in the array.

Definition at line 292 of file [env](#).

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

- l4/re/env

## 11.56 L4Re::Event\_buffer\_t< PAYLOAD >::Event Struct Reference

[Event](#) structure used in buffer.

### Public Member Functions

- void [free](#) () throw ()  
*Free the entry.*

### Data Fields

- long long [time](#)  
*Event time stamp.*

#### 11.56.1 Detailed Description

**template<typename PAYLOAD = Default\_event\_payload> struct L4Re::Event\_buffer\_t< PAYLOAD >::Event**

[Event](#) structure used in buffer.

Definition at line 85 of file [event](#).

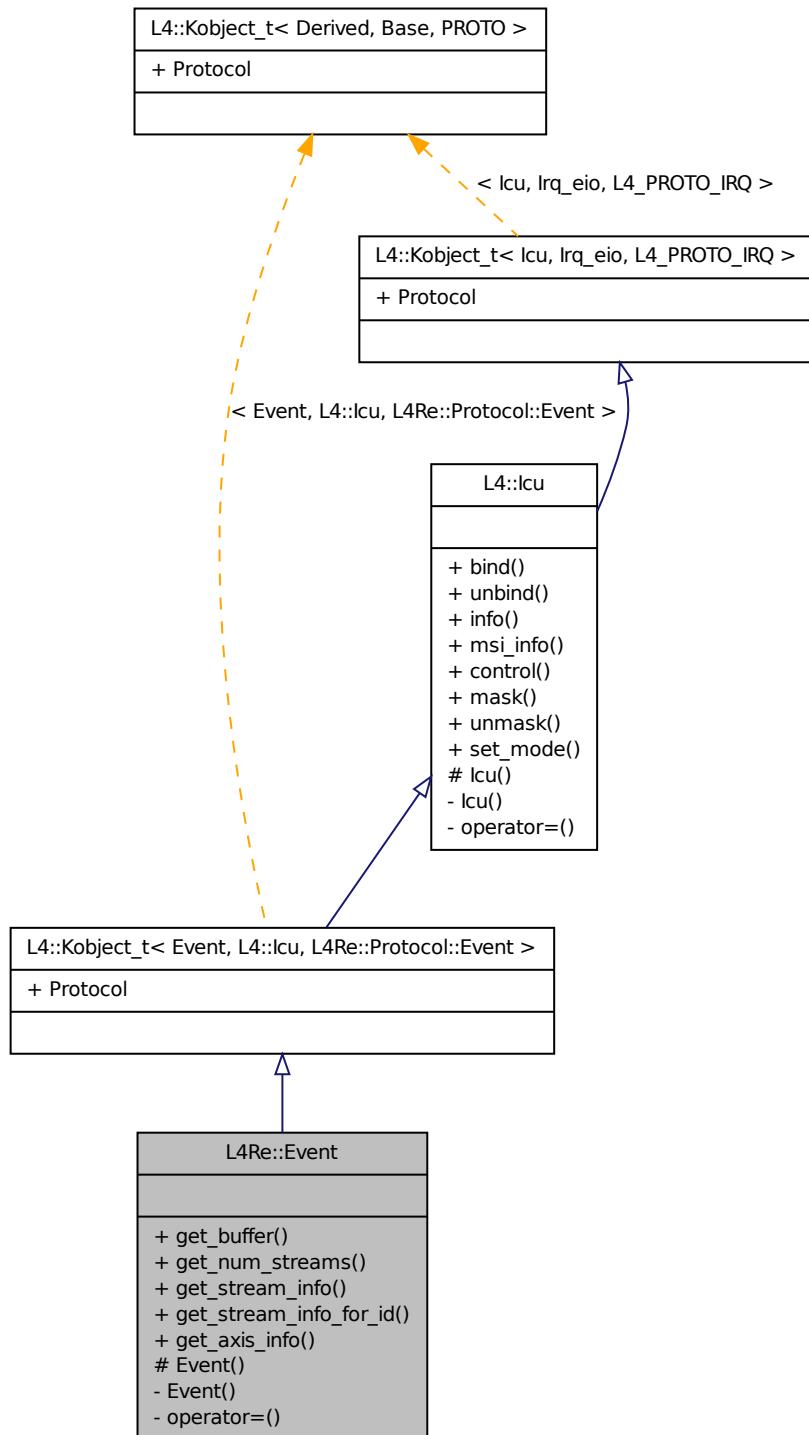
The documentation for this struct was generated from the following file:

- l4/re/event

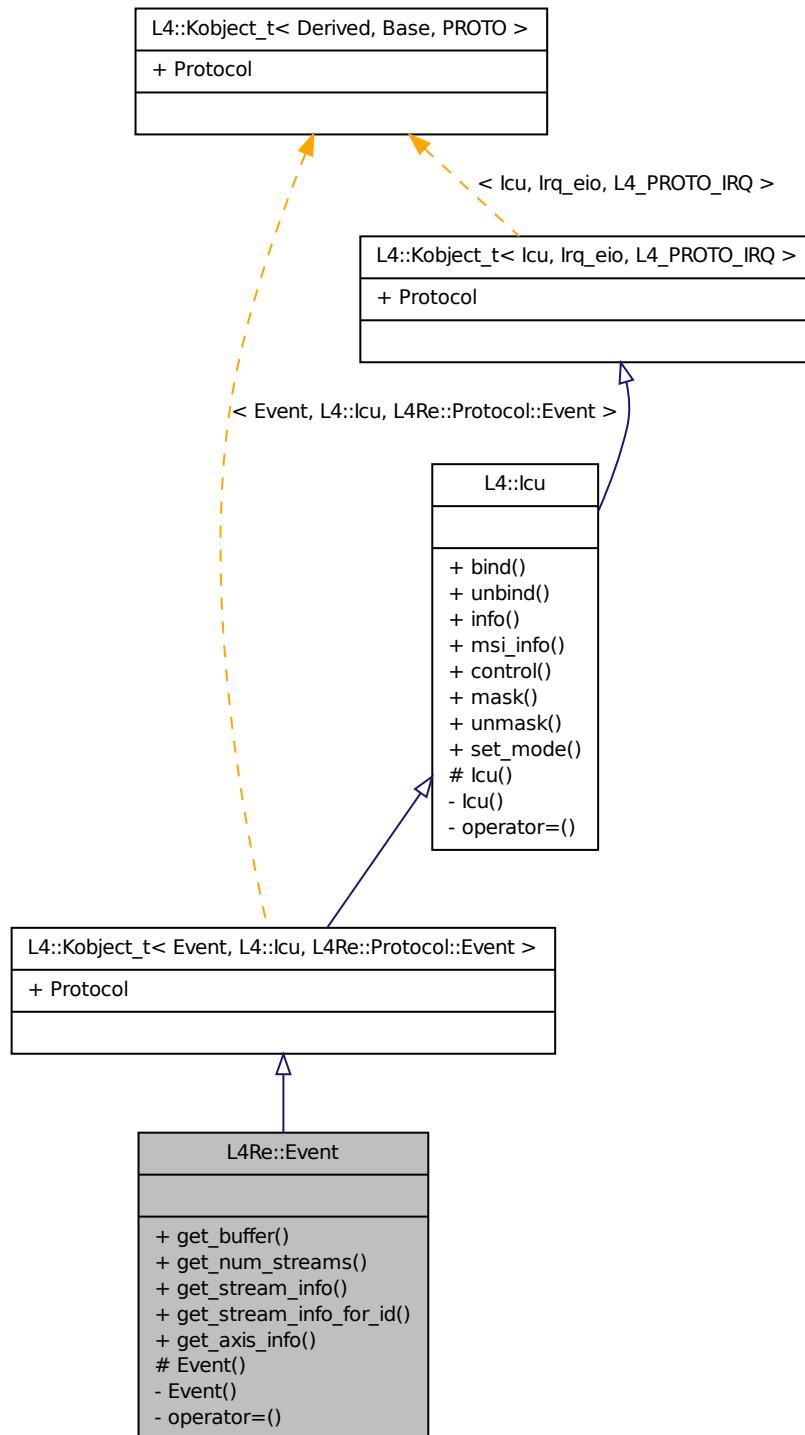
## 11.57 L4Re::Event Class Reference

[Event](#) class.

Inheritance diagram for L4Re::Event:



Collaboration diagram for L4Re::Event:



## Public Member Functions

- long [get\\_buffer \(L4::Cap< Dataspace > ds\) const throw \(\)](#)

*Get event signal and buffer.*

### 11.57.1 Detailed Description

[Event](#) class.

Definition at line 44 of file [event](#).

### 11.57.2 Member Function Documentation

#### 11.57.2.1 long L4Re::Event::get\_buffer ( L4::Cap< Dataspace > ds ) const throw ()

Get event signal and buffer.

##### Return values

*ds* [Event](#) buffer.

##### Returns

0 on success, negative error code otherwise.

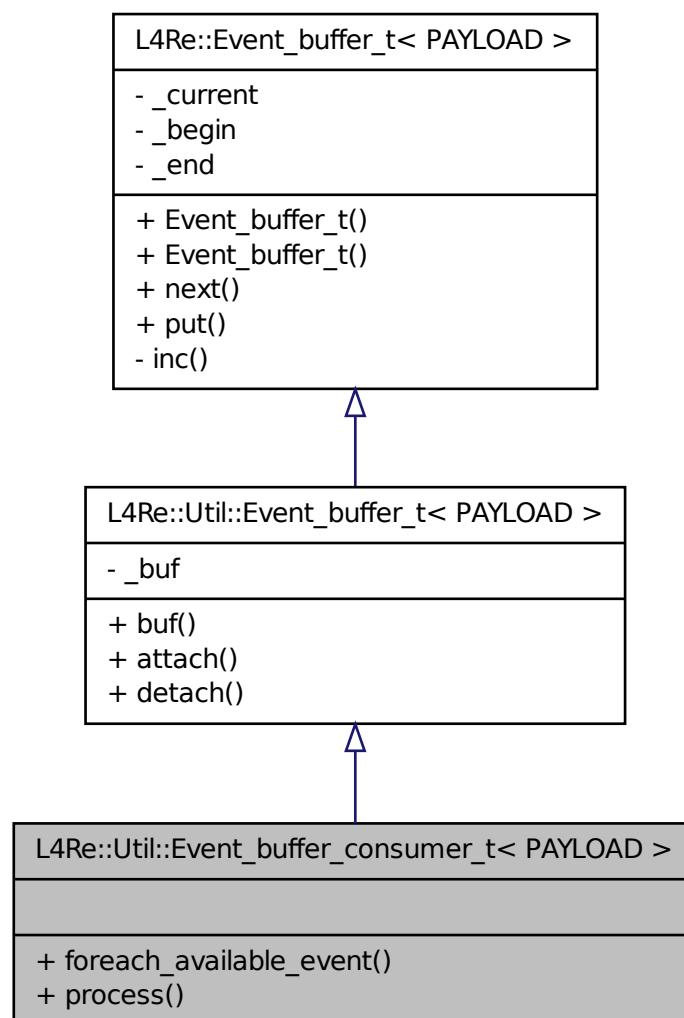
The documentation for this class was generated from the following file:

- [l4/re/event](#)

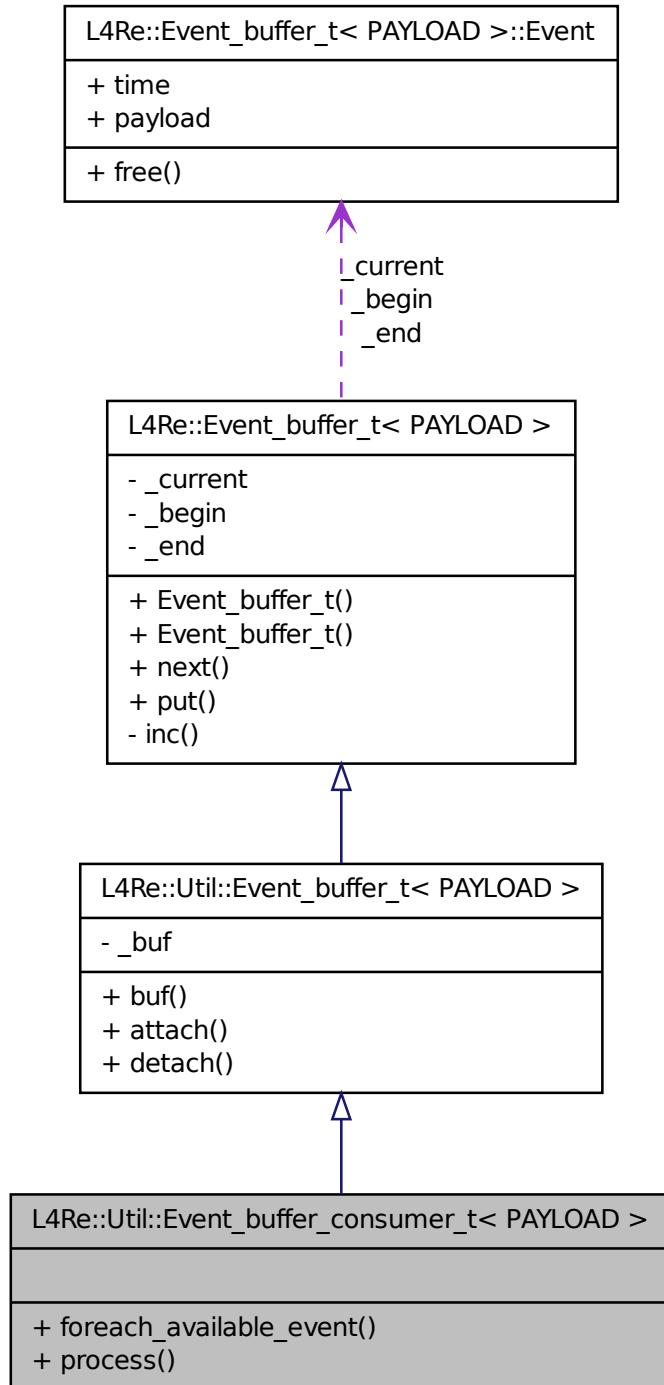
## 11.58 L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD > Class Template Reference

An event buffer consumer.

Inheritance diagram for L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD >:



Collaboration diagram for L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD >:



## Public Member Functions

- template<typename CB , typename D >  
void [foreach\\_available\\_event](#) (CB const &cb, D data=D())  
*Call function on every available event.*
- template<typename CB , typename D >  
void [process](#) ([L4::Cap< L4::Irq >](#) irq, [L4::Cap< L4::Thread >](#) thread, CB const &cb, D data=D())  
*Continuously wait for events and process them.*

### 11.58.1 Detailed Description

**template<typename PAYLOAD> class L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD >**

An event buffer consumer.

Definition at line 90 of file [event\\_buffer](#).

### 11.58.2 Member Function Documentation

#### 11.58.2.1 template<typename PAYLOAD > template<typename CB , typename D > void L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD >::foreach\_available\_event ( CB const & cb, D data = D() ) [inline]

Call function on every available event.

##### Parameters

*cb* Function callback.

Definition at line 100 of file [event\\_buffer](#).

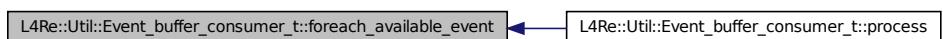
References [L4Re::Event\\_buffer\\_t< PAYLOAD >::Event::free\(\)](#).

Referenced by [L4Re::Util::Event\\_buffer\\_consumer\\_t< PAYLOAD >::process\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



**11.58.2.2 template<typename PAYLOAD> template<typename CB , typename D> void L4Re::Util::Event\_buffer\_consumer\_t< PAYLOAD >::process ( L4::Cap< L4::Irq > *irq*, L4::Cap< L4::Thread > *thread*, CB const & *cb*, D *data* = D() ) [inline]**

Continuously wait for events and process them.

### Parameters

*irq* [Event](#) signal to wait for.

*thread* Thread capability of the thread calling this function.

*cb* Callback function that is called for each received event.

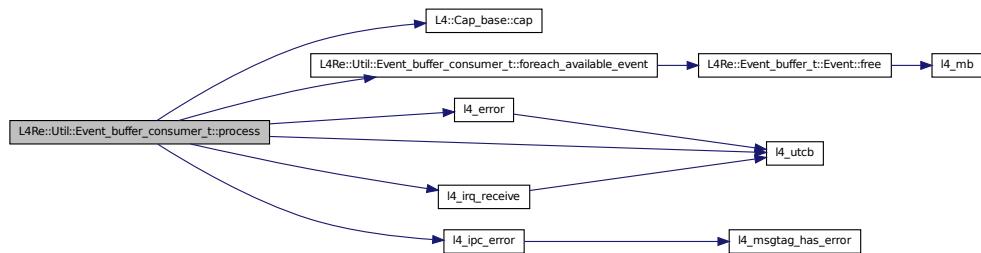
### Note

This function never returns.

Definition at line 120 of file [event\\_buffer](#).

References [L4::Cap\\_base::cap\(\)](#), [L4Re::Util::Event\\_buffer\\_consumer\\_t< PAYLOAD >::foreach\\_available\\_event\(\)](#), [l4\\_error\(\)](#), [l4\\_ipc\\_error\(\)](#), [l4\\_irq\\_receive\(\)](#), and [l4\\_utcb\(\)](#).

Here is the call graph for this function:



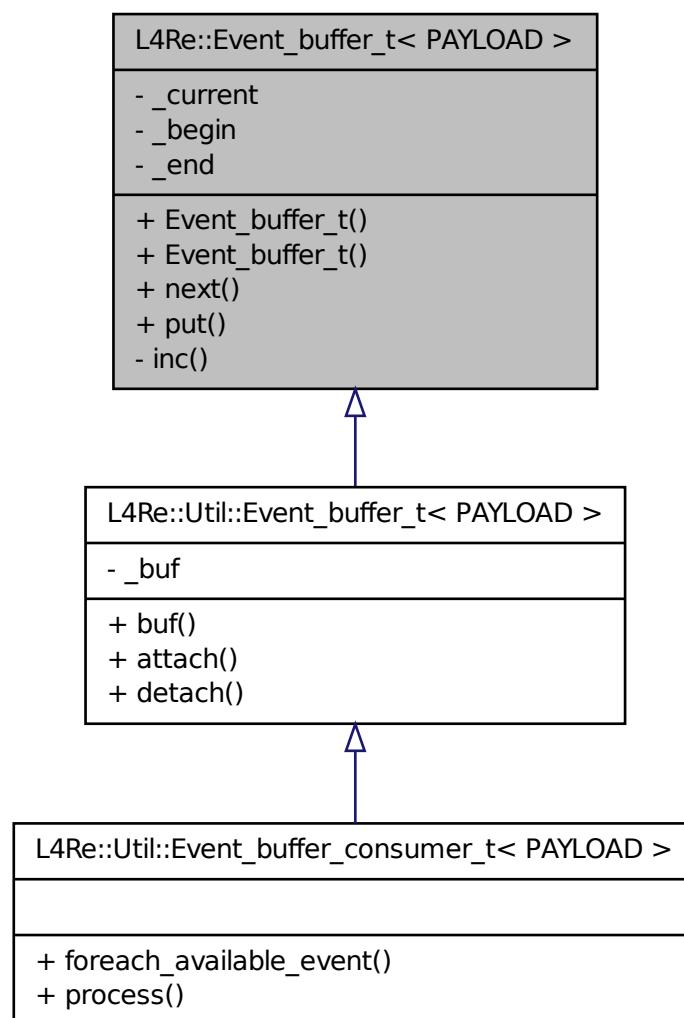
The documentation for this class was generated from the following file:

- [l4/re/util/event\\_buffer](#)

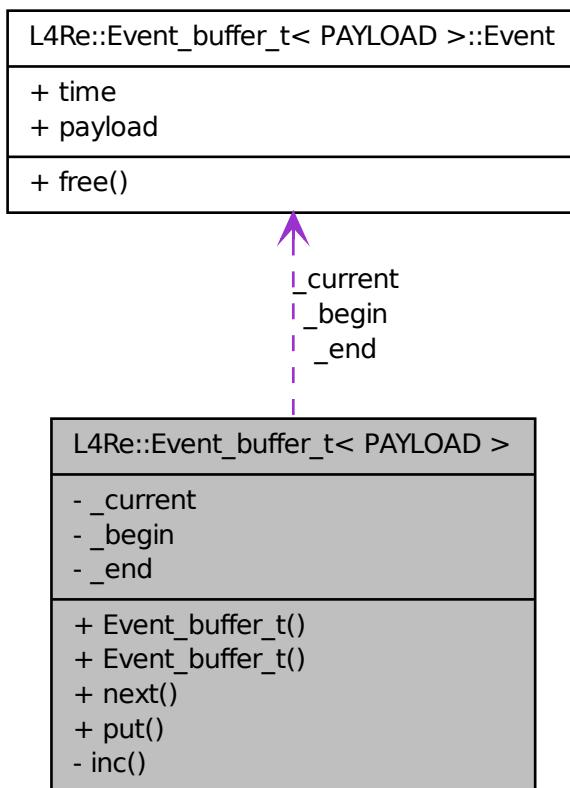
## 11.59 L4Re::Event\_buffer\_t< PAYLOAD > Class Template Reference

[Event](#) buffer class.

Inheritance diagram for L4Re::Event\_buffer\_t< PAYLOAD >:



Collaboration diagram for L4Re::Event\_buffer\_t< PAYLOAD >:



## Data Structures

- struct [Event](#)

*Event* structure used in buffer.

## Public Member Functions

- [Event\\_buffer\\_t](#) (void \*buffer, [l4\\_addr\\_t](#) size)  
*Initialize event buffer.*
- [Event \\* next \(\) throw \(\)](#)  
*Next event in buffer.*
- [bool put \(Event const &ev\) throw \(\)](#)  
*Put event into buffer at current position.*

### 11.59.1 Detailed Description

```
template<typename PAYLOAD = Default_event_payload> class L4Re::Event_buffer_t< PAYLOAD >
```

[Event](#) buffer class.

Definition at line 78 of file [event](#).

### 11.59.2 Constructor & Destructor Documentation

```
11.59.2.1 template<typename PAYLOAD = Default_event_payload> L4Re::Event_buffer_t< PAYLOAD >::Event_buffer_t( void * buffer, l4_addr_t size ) [inline]
```

Initialize event buffer.

#### Parameters

*buffer* Pointer to buffer.

*size* Size of buffer in bytes.

Definition at line 118 of file [event](#).

### 11.59.3 Member Function Documentation

```
11.59.3.1 template<typename PAYLOAD = Default_event_payload> Event* L4Re::Event_buffer_t< PAYLOAD >::next( ) throw() [inline]
```

Next event in buffer.

#### Returns

0 if no event available, event otherwise.

Definition at line 128 of file [event](#).

References [L4Re::Event\\_buffer\\_t< PAYLOAD >::Event::time](#).

```
11.59.3.2 template<typename PAYLOAD = Default_event_payload> bool L4Re::Event_buffer_t< PAYLOAD >::put( Event const & ev ) throw() [inline]
```

Put event into buffer at current position.

#### Parameters

*ev* [Event](#) to put into the buffer.

#### Returns

false if buffer is full and entry could not be added.

Definition at line 145 of file [event](#).

References [l4\\_wmb\(\)](#), and [L4Re::Event\\_buffer\\_t< PAYLOAD >::Event::time](#).

Here is the call graph for this function:



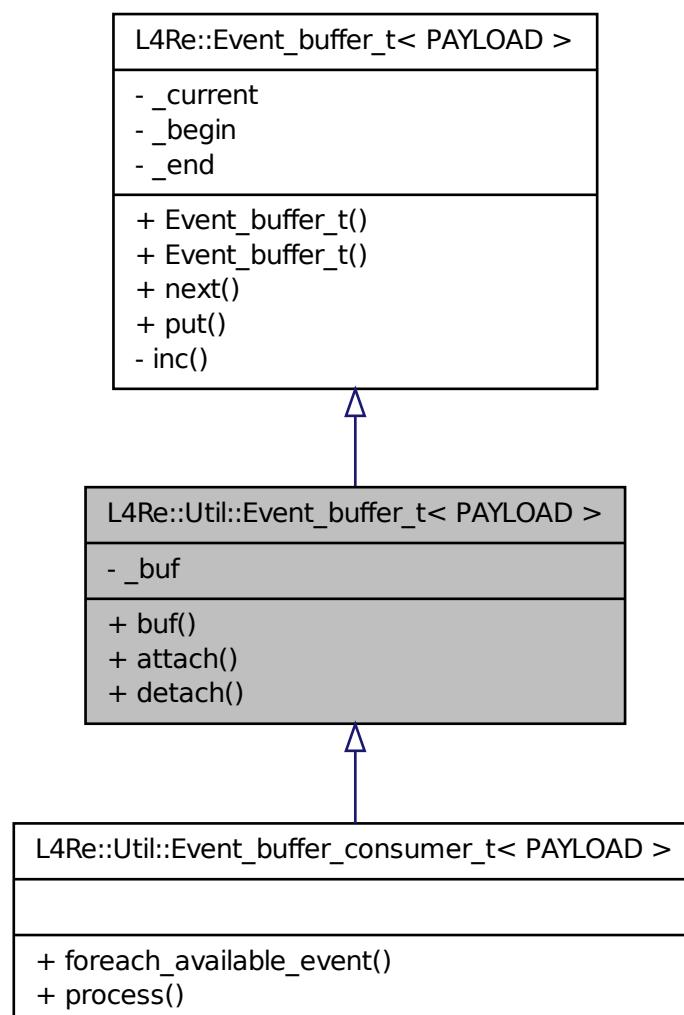
The documentation for this class was generated from the following file:

- l4/re/event

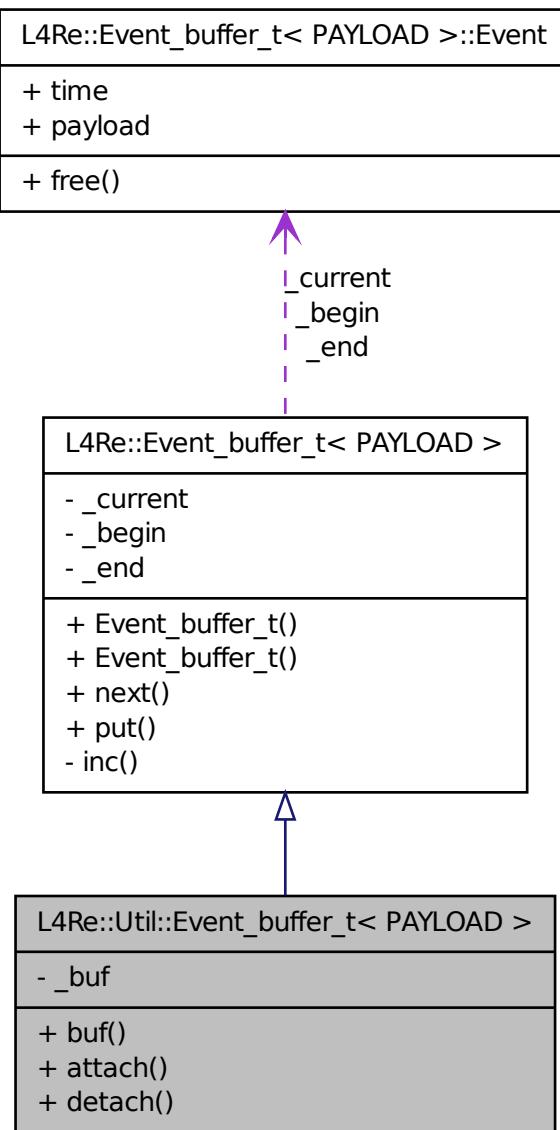
## **11.60 L4Re::Util::Event\_buffer\_t< PAYLOAD > Class Template Reference**

Event\_buffer utility class.

Inheritance diagram for L4Re::Util::Event\_buffer\_t< PAYLOAD >:



Collaboration diagram for L4Re::Util::Event\_buffer\_t< PAYLOAD >:



## Public Member Functions

- `void * buf() const throw ()`  
*Return the buffer.*
- `long attach (L4::Cap< L4Re::Dataspace > ds, L4::Cap< L4Re::Rm > rm) throw ()`  
*Attach event buffer from address space.*

- long `detach (L4::Cap< L4Re::Rm > rm, void *buf) throw ()`  
*Detach event buffer from address space.*

### 11.60.1 Detailed Description

`template<typename PAYLOAD> class L4Re::Util::Event_buffer_t< PAYLOAD >`

Event\_buffer utility class.

Definition at line 36 of file [event\\_buffer](#).

### 11.60.2 Member Function Documentation

#### 11.60.2.1 `template<typename PAYLOAD > void* L4Re::Util::Event_buffer_t< PAYLOAD >::buf ( ) const throw () [inline]`

Return the buffer.

##### Returns

Pointer to the event buffer.

Definition at line 46 of file [event\\_buffer](#).

Referenced by [L4Re::Util::Event\\_buffer\\_t< PAYLOAD >::detach\(\)](#).

Here is the caller graph for this function:



#### 11.60.2.2 `template<typename PAYLOAD > long L4Re::Util::Event_buffer_t< PAYLOAD >::attach ( L4::Cap< L4Re::Dataspace > ds, L4::Cap< L4Re::Rm > rm ) throw () [inline]`

Attach event buffer from address space.

##### Parameters

`ds` Dataspace of the event buffer.

`rm` Region manager to attach buffer to.

##### Returns

0 on success, negative error code otherwise.

Definition at line 56 of file [event\\_buffer](#).

References [L4Re::Rm::Search\\_addr](#).

**11.60.2.3 template<typename PAYLOAD > long L4Re::Util::Event\_buffer\_t< PAYLOAD >::detach ( L4::Cap< L4Re::Rm > rm, void \* buf ) throw () [inline]**

Detach event buffer from address space.

#### Parameters

*rm* Region manager to detach buffer from.

*buf* Buffer pointer.

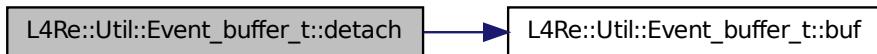
#### Returns

0 on success, negative error code otherwise.

Definition at line 77 of file [event\\_buffer](#).

References [L4Re::Util::Event\\_buffer\\_t< PAYLOAD >::buf\(\)](#).

Here is the call graph for this function:



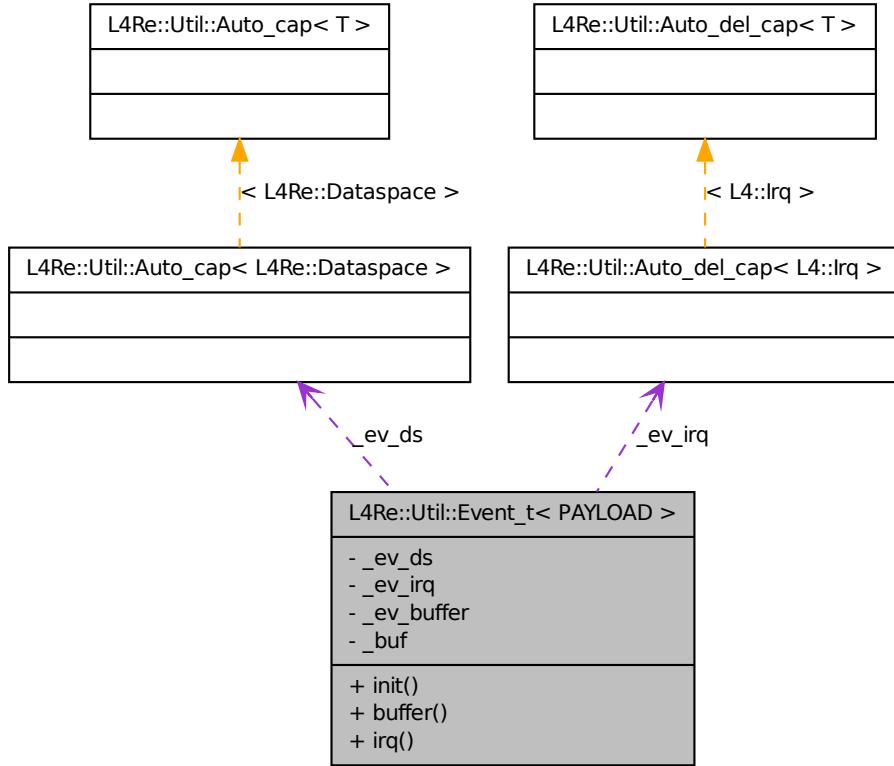
The documentation for this class was generated from the following file:

- [l4/re/util/event\\_buffer](#)

## 11.61 L4Re::Util::Event\_t< PAYLOAD > Class Template Reference

Convenience wrapper for getting access to an event object.

Collaboration diagram for L4Re::Util::Event\_t< PAYLOAD >:



## Public Types

- enum `Mode` { `Mode_irq`, `Mode_polling` }
- Modes of operation.*

## Public Member Functions

- int `init` (`L4::Cap< L4Re::Event >` event, `Mode` mode=`Mode_irq`, `L4Re::Env` const \*env=`L4Re::Env::env()`, `L4Re::Cap_alloc` \*ca=`L4Re::Cap_alloc::get_cap_alloc(L4Re::Util::cap_alloc())`)  
*Initialise an event object.*
- `L4Re::Event_buffer_t< PAYLOAD > & buffer()`  
*Get event buffer.*
- `L4::Cap< L4::Irq > irq()` const  
*Get event IRQ.*

### 11.61.1 Detailed Description

**template<typename PAYLOAD> class L4Re::Util::Event\_t< PAYLOAD >**

Convenience wrapper for getting access to an event object. After calling `init()` the class supplies the event-buffer and the associated IRQ object.

Definition at line 39 of file [event](#).

### 11.61.2 Member Enumeration Documentation

**11.61.2.1 template<typename PAYLOAD> enum L4Re::Util::Event\_t::Mode**

Modes of operation.

**Enumerator:**

*Mode\_irq* Create an IRQ and attach, to get notifications.

*Mode\_polling* Do not use an IRQ.

Definition at line 35 of file [event](#).

### 11.61.3 Member Function Documentation

**11.61.3.1 template<typename PAYLOAD> int L4Re::Util::Event\_t< PAYLOAD >::init ( L4::Cap< L4Re::Event > event, Mode mode = Mode\_irq, L4Re::Env const \* env = L4Re::Env::env(), L4Re::Cap\_alloc \* ca = L4Re::Cap\_alloc::get\_cap\_alloc(L4Re::Util::cap\_alloc) ) [inline]**

Initialise an event object.

#### Parameters

*event* Capability to event.

*env* Optional: Pointer to L4Re-Environment

*ca* Optional: Pointer to capability allocator.

#### Returns

0 on success, error code on error

Definition at line 49 of file [event](#).

**11.61.3.2 template<typename PAYLOAD> L4Re::Event\_buffer\_t<PAYLOAD>& L4Re::Util::Event\_t< PAYLOAD >::buffer ( ) [inline]**

Get event buffer.

#### Returns

[Event](#) buffer object.

Definition at line 98 of file [event](#).

### 11.61.3.3 template<typename PAYLOAD > L4::Cap<L4::Irq> L4Re::Util::Event\_t< PAYLOAD >::irq ( ) const [inline]

Get event IRQ.

#### Returns

Event IRQ.

Definition at line 104 of file [event](#).

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

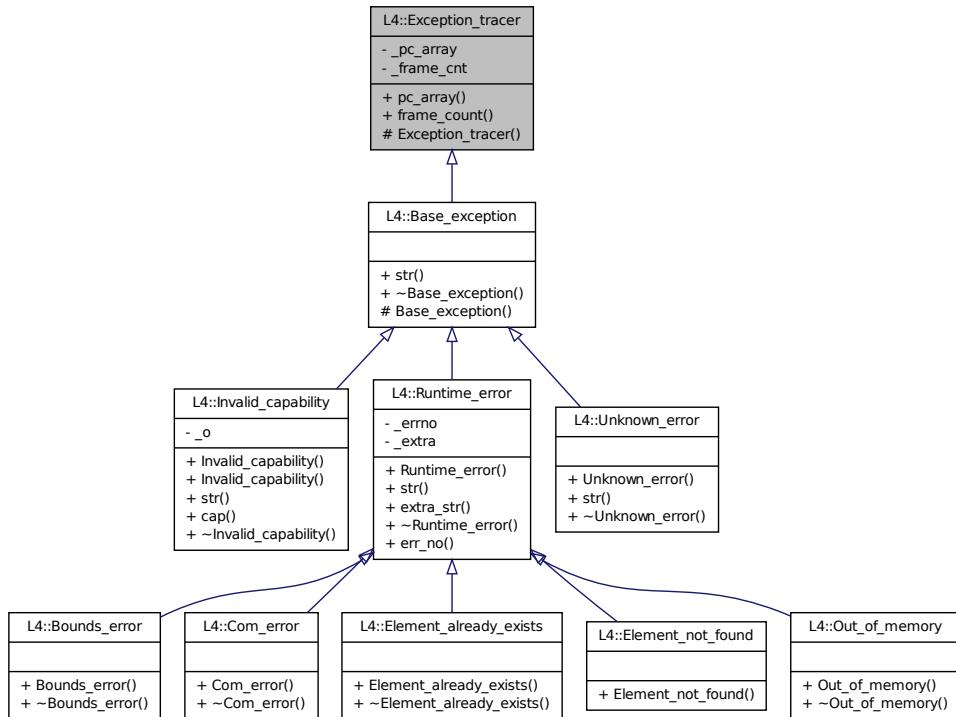
- [l4/re/util/event](#)

## 11.62 L4::Exception\_tracer Class Reference

Back-trace support for exceptions.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Exception\_tracer:



## Public Member Functions

- `void const *const * pc_array () const throw ()`

*Get the array containing the call trace.*

- int [frame\\_count \(\) const throw \(\)](#)

*Get the number of entries that are valid in the call trace.*

## Protected Member Functions

- [Exception\\_tracer \(\) throw \(\)](#)

*Create a back trace.*

### 11.62.1 Detailed Description

Back-trace support for exceptions. This class holds an array of at most L4\_CXX\_EXCEPTION\_BACKTRACE instruction pointers containing the call trace at the instant when an exception was thrown.

Definition at line [60](#) of file [exceptions](#).

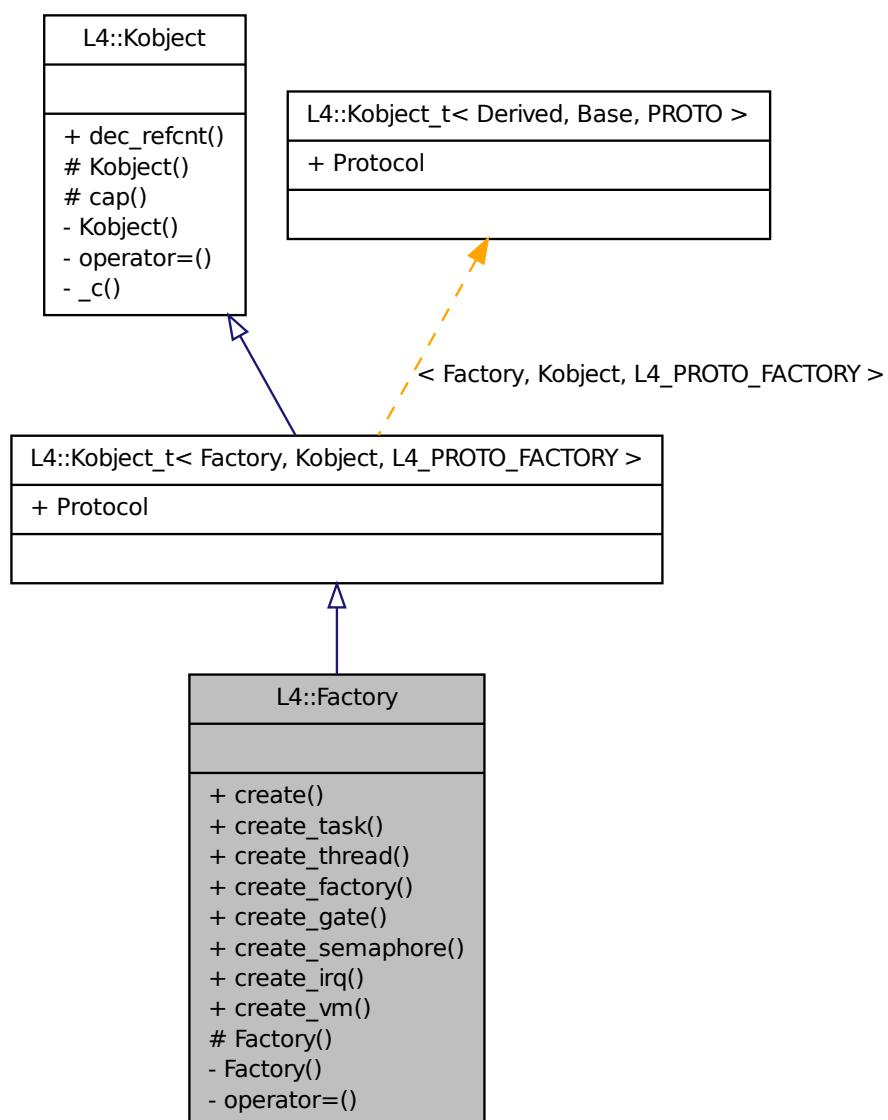
The documentation for this class was generated from the following file:

- [14/cxx/exceptions](#)

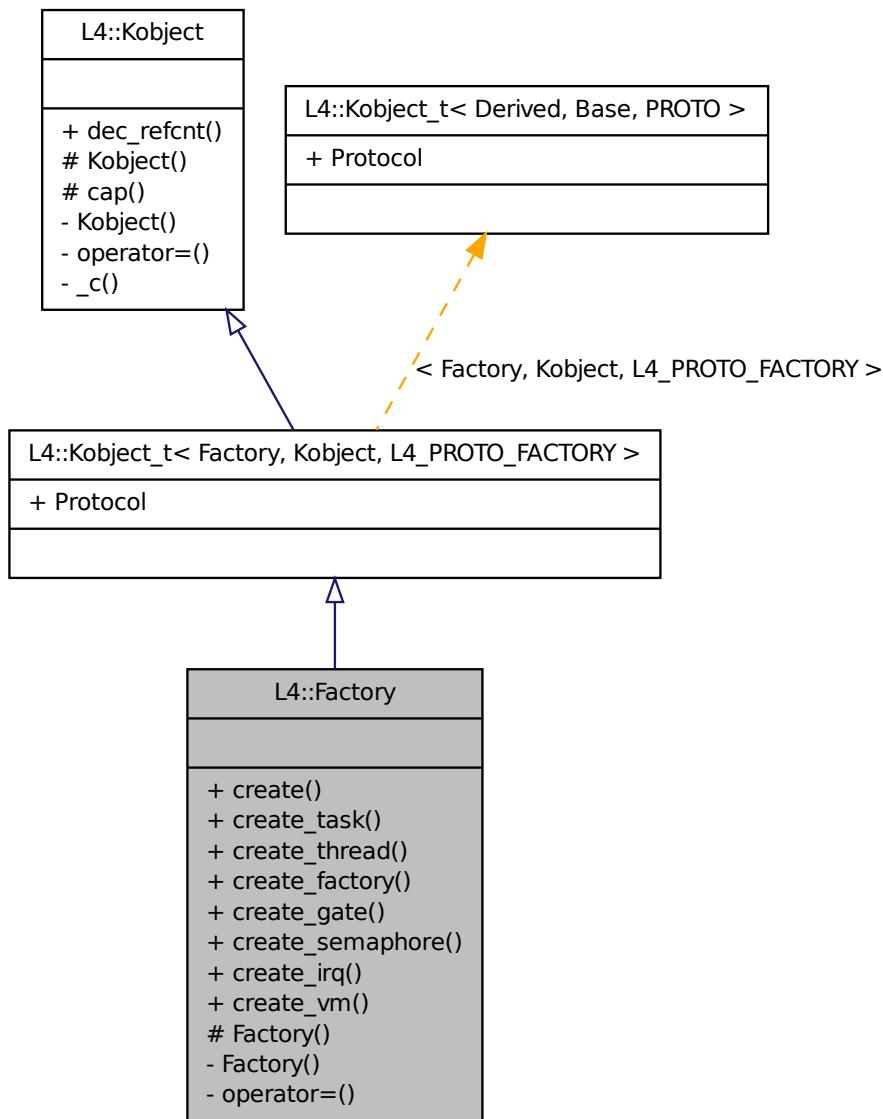
## 11.63 L4::Factory Class Reference

C++ [L4 Factory](#), to create all kinds of kernel objects.

Inheritance diagram for L4::Factory:



Collaboration diagram for L4::Factory:



## Data Structures

- struct [Lstr](#)

*Special type to add a pascal string into the factory create stream.*

- struct [Nil](#)

*Special type to add a void argument into the factory create stream.*

- class **S**

*Stream class for the [create\(\)](#) argument stream.*

## Public Member Functions

- **S create (Cap< Kobject > target, long obj, l4\_utcb\_t \*utcb=l4\_utcb()) throw ()**  
*Generic create call to the factory.*
- **l4\_mshtag\_t create\_task (Cap< Task > const &target\_cap, l4\_fpage\_t const &utcb\_area, l4\_utcb\_t \*utcb=l4\_utcb()) throw ()**
- **l4\_mshtag\_t create\_thread (Cap< Thread > const &target\_cap, l4\_utcb\_t \*utcb=l4\_utcb()) throw ()**
- **l4\_mshtag\_t create\_factory (Cap< Factory > const &target\_cap, unsigned long limit, l4\_utcb\_t \*utcb=l4\_utcb()) throw ()**
- **l4\_mshtag\_t create\_gate (Cap< Kobject > const &target\_cap, Cap< Thread > const &thread\_cap, l4\_umword\_t label, l4\_utcb\_t \*utcb=l4\_utcb()) throw ()**
- **l4\_mshtag\_t create\_semaphore (Cap< K\_semaphore > const &target\_cap, l4\_utcb\_t \*utcb=l4\_utcb()) throw ()**
- **l4\_mshtag\_t create\_irq (Cap< Irq >const &target\_cap, l4\_utcb\_t \*utcb=l4\_utcb()) throw ()**
- **l4\_mshtag\_t create\_vm (Cap< Vm >const &target\_cap, l4\_utcb\_t \*utcb=l4\_utcb()) throw ()**

### 11.63.1 Detailed Description

C++ [L4 Factory](#), to create all kinds of kernel objects. #include <l4/sys/factory>

#### See also

[Factory](#) for an overview and C bindings.

Definition at line 41 of file [factory](#).

### 11.63.2 Member Function Documentation

#### 11.63.2.1 S L4::Factory::create ( Cap< Kobject > target, long obj, l4\_utcb\_t \* utcb = **l4\_utcb ()** ) throw () [inline]

Generic create call to the factory.

#### Parameters

*target* is the target capability selector where the new object shall be received.

*obj* is the protocol ID that specifies which kind of object shall be created.

*utcb* is the UTCB to use for the operation.

#### Returns

a create stream that allows adding additional arguments to the [create\(\)](#) call.

This method does currently not directly invoke the factory. It returns a stream that shall invoke the factory after adding all additional arguments.

Usage:

```
L4::Cap<L4Re::Namespace> ns = L4Re::Util::cap_alloc.alloc<L4Re::Namespace>();
factory->create(ns, L4Re::Namespace::Protocol) << "Argument text";
```

Definition at line 213 of file [factory](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 11.63.2.2 l4\_mshtag\_t L4::Factory::create\_task ( Cap< Task > const & target\_cap, l4\_fpage\_t const & utcb\_area, l4\_utcb\_t \* utcb = l4\_utcb() ) throw () [inline]

Create a new task.

#### Parameters

*factory* Capability selector for factory to use for creation.

*target\_cap* Capability selector for the root capability of the new task.

*utcb\_area* Flexpage that describes the area for the UTCBs of the new task

#### Note

The size of the UTCB area specifies indirectly the maximum number of UTCBs available for this task and cannot be changed afterwards.

#### Returns

Syscall return tag

#### See also

[Task](#)

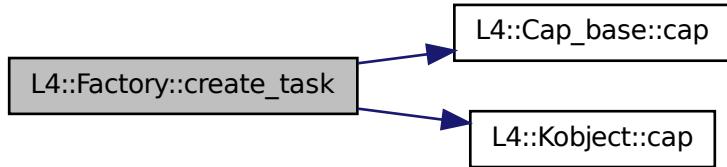
#### Note

*factory* is the implicit *this* pointer.

Definition at line 222 of file [factory](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 11.63.2.3 `l4_mshtag_t L4::Factory::create_thread ( Cap< Thread > const & target_cap, l4_utcb_t * utcb = l4_utcb () ) throw () [inline]`

Create a new thread.

#### Parameters

- `factory` Capability selector for factory to use for creation.
- `target_cap` Capability selector for the root capability of the new thread.

#### Returns

Syscall return tag

#### See also

[Thread](#)

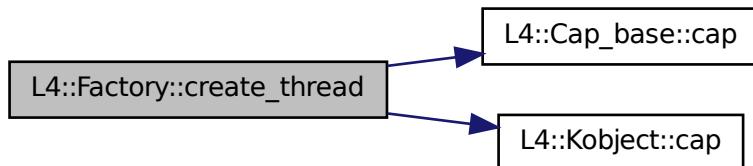
#### Note

`factory` is the implicit *this* pointer.

Definition at line 231 of file [factory](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



---

**11.63.2.4 l4\_mshtag\_t L4::Factory::create\_factory ( Cap< Factory > const & *target\_cap*,  
unsigned long *limit*, l4\_utcb\_t \* *utcb* = *L4\_utcb()* ) throw () [inline]**

Create a new factory.

#### Parameters

*factory* Capability selector for factory to use for creation.

*target\_cap* Capability selector for the root capability of the new factory.

*limit* Limit for the new factory in bytes

#### Note

The limit of the new factory is subtracted from the available amount of the factory used for creation.

#### Returns

Syscall return tag

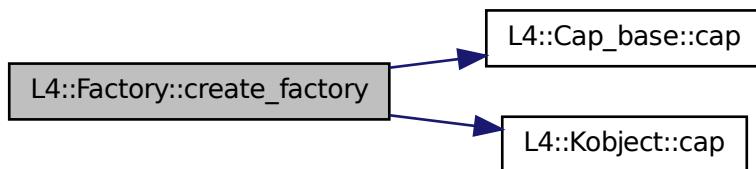
#### Note

*factory* is the implicit *this* pointer.

Definition at line 239 of file [factory](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:




---

**11.63.2.5 l4\_mshtag\_t L4::Factory::create\_gate ( Cap< Kobject > const & *target\_cap*, Cap<  
Thread > const & *thread\_cap*, l4\_umword\_t *label*, l4\_utcb\_t \* *utcb* = *L4\_utcb()* )  
throw () [inline]**

Create a new IPC gate.

#### Parameters

*factory* Capability selector for factory to use for creation.

*target\_cap* Capability selector for the root capability of the new IPC gate.

*thread\_cap* Thread to bind the gate to

*label* Label of the gate

### Returns

Syscall return tag

### See also

[IPC-Gate API](#)

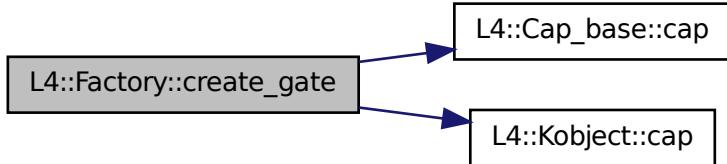
### Note

*factory* is the implicit *this* pointer.

Definition at line [248](#) of file [factory](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



**11.63.2.6 l4\_mshtag\_t L4::Factory::create\_semaphore ( Cap< K\_semaphore > const & target\_cap, l4\_utcb\_t \* utcb = *l4\_utcb()* ) throw () [inline]**

Create a new semaphore.

### Parameters

*factory* Capability selector for factory to use for creation.

*target\_cap* Capability selector for the root capability of the new semaphore.

### Returns

Syscall return tag

### See also

[l4\\_sem\\_api](#)

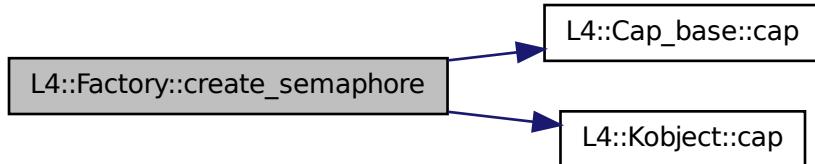
### Note

*factory* is the implicit *this* pointer.

Definition at line 257 of file [factory](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



**11.63.2.7 l4\_msgtag\_t L4::Factory::create\_irq ( Cap< Irq >const & target\_cap, l4\_utcb\_t \* utcb = [l4\\_utcb\(\)](#) ) throw () [inline]**

Create a new IRQ.

#### Parameters

*factory* Capability selector for factory to use for creation.

*target\_cap* Capability selector for the root capability of the new IRQ.

#### Returns

Syscall return tag

#### See also

[IRQs](#)

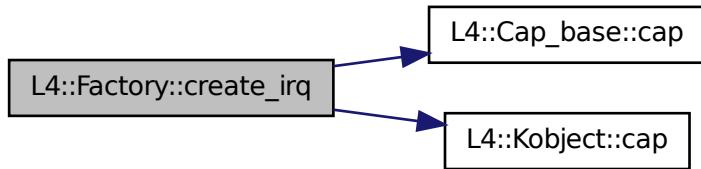
#### Note

*factory* is the implicit *this* pointer.

Definition at line 265 of file [factory](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 11.63.2.8 l4\_mshtag\_t L4::Factory::create\_vm ( Cap< Vm >const & target\_cap, l4\_utcb\_t \* utcb = *l4\_utcb()* ) throw () [inline]

Create a new virtual machine.

#### Parameters

*factory* Capability selector for factory to use for creation.

*target\_cap* Capability selector for the root capability of the new VM.

#### Returns

Syscall return tag

#### See also

[Virtual Machines](#)

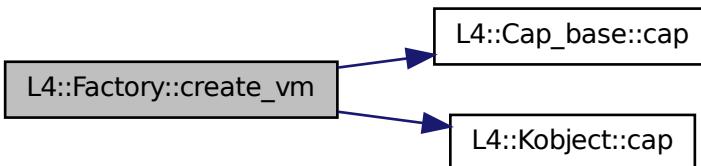
#### Note

*factory* is the implicit *this* pointer.

Definition at line 273 of file [factory](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



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

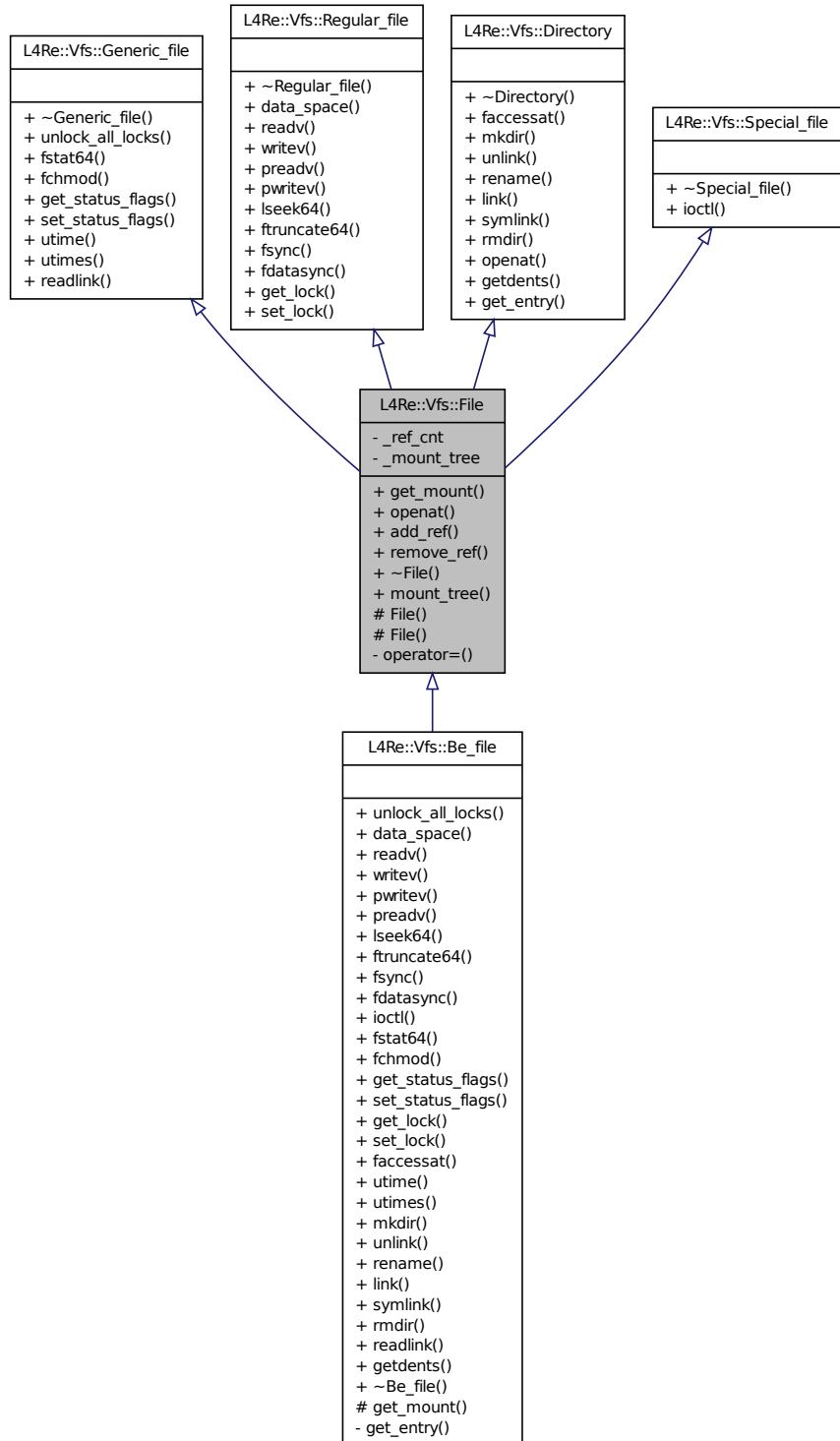
- l4/sys/factory

## **11.64 L4Re::Vfs::File Class Reference**

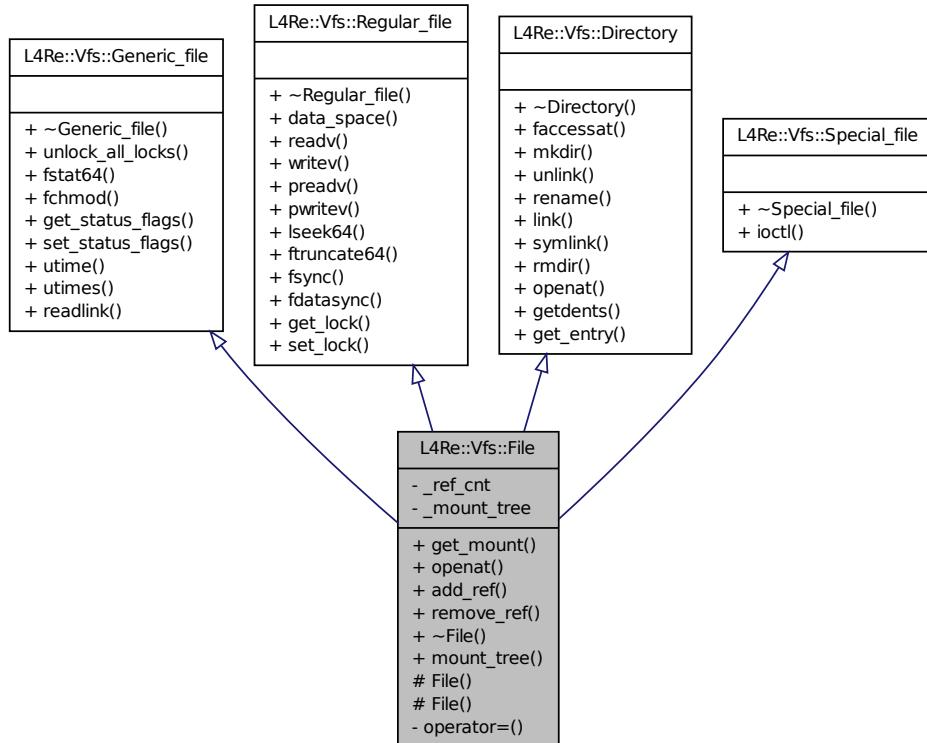
The basic interface for an open POSIX file.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::File:



Collaboration diagram for L4Re::Vfs::File:



### 11.64.1 Detailed Description

The basic interface for an open POSIX file. An open POSIX file can be anything that hides behind a POSIX file descriptor. This means that even a directories are files. An open file can be anything from a directory to a special device file so see [Generic\\_file](#), [Regular\\_file](#), [Directory](#), and [Special\\_file](#) for more information.

#### Note

For implementing a backend for the [L4Re::Vfs](#) you may use [L4Re::Vfs::Be\\_file](#) as a base class.

Definition at line 403 of file [vfs.h](#).

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

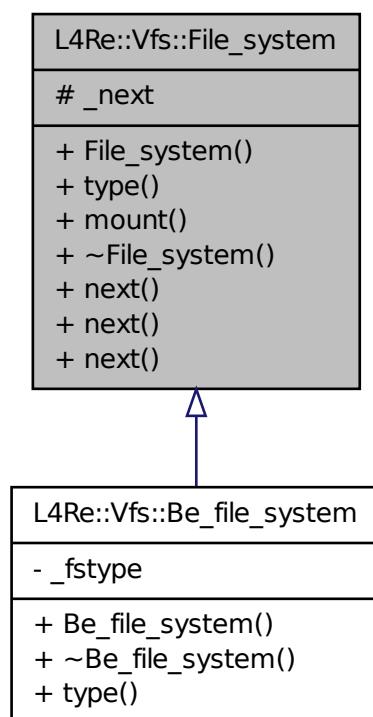
- [l4/l4re\\_vfs/vfs.h](#)

## 11.65 L4Re::Vfs::File\_system Class Reference

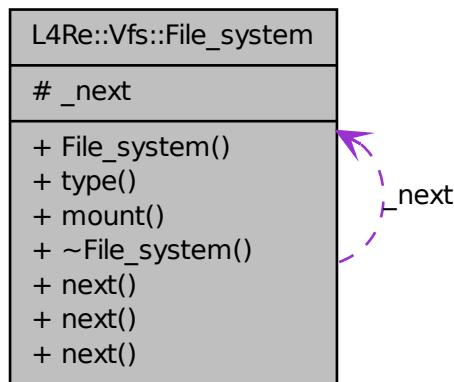
Basic interface for an [L4Re::Vfs](#) file system.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::File\_system:



Collaboration diagram for L4Re::Vfs::File\_system:



## Public Member Functions

- virtual char const \* [type \(\) const =0 throw \(\)](#)

*Returns the type of the file system, used in mount as fstype argument.*

- virtual int [mount \(char const \\*source, unsigned long mountflags, void const \\*data, cxx::Ref\\_ptr< File > \\*dir\)=0 throw \(\)](#)

*Create a directory object dir representing source mounted with this file system.*

### 11.65.1 Detailed Description

Basic interface for an [L4Re::Vfs](#) file system.

#### Note

For implementing a special file system you may use [L4Re::Vfs::Be\\_file\\_system](#) as a base class.

The main purpose of this interface is that there is a single object for each supported file-system type (e.g., ext2, vfat) exists in your application and is registered at the [L4Re::Vfs::Fs](#) singleton available in via [L4Re::Vfs::vfs\\_ops](#). At the end the POSIX mount function call the [File\\_system::mount](#) method for the given file-system type given in mount.

Definition at line [799](#) of file [vfs.h](#).

## 11.65.2 Member Function Documentation

### 11.65.2.1 virtual char const\* L4Re::Vfs::File\_system::type ( ) const throw () [pure virtual]

Returns the type of the file system, used in mount as fstype argument.

#### Note

This method is already provided by [Be\\_file\\_system](#).

Implemented in [L4Re::Vfs::Be\\_file\\_system](#).

### 11.65.2.2 virtual int L4Re::Vfs::File\_system::mount ( char const \* source, unsigned long mountflags, void const \* data, cxx::Ref\_ptr< File > \* dir ) throw () [pure virtual]

Create a directory object *dir* representing *source* mounted with this file system.

#### Parameters

*source* The path to the source device to mount. This may also be some URL or anything file-system specific.

*mountflags* The mount flags as specified in the POSIX mount call.

*data* The data as specified in the POSIX mount call. The contents are file-system specific.

#### Return values

*dir* A new directory object representing the file-system root directory.

#### Returns

0 on success, and <0 on error (e.g. -EINVAL).

#### Examples:

[tmpfs/lib/src/fs.cc](#).

Referenced by [L4Re::Vfs::Fs::mount\(\)](#).

Here is the caller graph for this function:



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

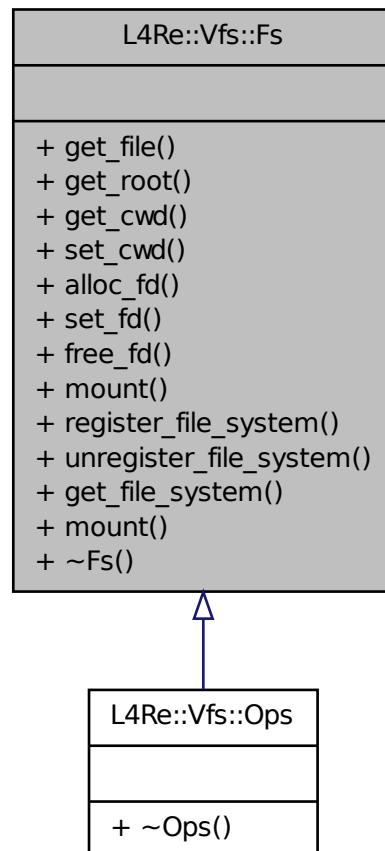
- [l4/re\\_vfs/vfs.h](#)

## 11.66 L4Re::Vfs::Fs Class Reference

POSIX File-system related functionality.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Fs:



### Public Member Functions

- virtual cxx::Ref\_ptr< [File](#) > [get\\_file](#) (int fd)=0 throw ()
   
*Get the [L4Re::Vfs::File](#) for the file descriptor fd.*
- virtual cxx::Ref\_ptr< [File](#) > [get\\_root](#) ()=0 throw ()
   
*Get the directory object for the applications root directory.*
- virtual cxx::Ref\_ptr< [File](#) > [get\\_cwd](#) () throw ()
   
*Get the directory object for the applications current working directory.*

- virtual void [set\\_cwd](#) (cxx::Ref\_ptr<File> const &) throw ()
 

*Set the current working directory for the application.*
- virtual int [alloc\\_fd](#) (cxx::Ref\_ptr<File> const &f=cxx::Ref\_ptr<>::Nil)=0 throw ()
 

*Allocate the next free file descriptor.*
- virtual cxx::Ref\_ptr<File> [set\\_fd](#) (int fd, cxx::Ref\_ptr<File> const &f=cxx::Ref\_ptr<>::Nil)=0 throw ()
 

*Set the file object referenced by the file descriptor fd.*
- virtual cxx::Ref\_ptr<File> [free\\_fd](#) (int fd)=0 throw ()
 

*Free the file descriptor fd.*
- int [mount](#) (char const \*path, cxx::Ref\_ptr<File> const &dir) throw ()
 

*Mount a given file object at the given global path in the VFS.*
- int [mount](#) (char const \*source, char const \*target, char const \*fstype, unsigned long mountflags, void const \*data) throw ()
 

*Backend for the POSIX mount call.*

### 11.66.1 Detailed Description

POSIX File-system related functionality.

#### Note

This class usually exists as a singleton as a superclass of [L4Re::Vfs::Ops](#) (

#### See also

[L4Re::Vfs::vfs\\_ops](#)).

Definition at line 851 of file [vfs.h](#).

### 11.66.2 Member Function Documentation

#### 11.66.2.1 virtual cxx::Ref\_ptr<File> [L4Re::Vfs::Fs::get\\_file](#) ( int *fd* ) throw () [pure virtual]

Get the [L4Re::Vfs::File](#) for the file descriptor *fd*.

#### Parameters

*fd* The POSIX file descriptor number.

#### Returns

A pointer to the [File](#) object, or 0 if *fd* is not open.

**11.66.2.2 virtual int L4Re::Vfs::Fs::alloc\_fd ( cxx::Ref\_ptr< File > const & *f* =  
  cxx::Ref\_ptr<>::Nil ) throw () [pure virtual]**

Allocate the next free file descriptor.

#### Parameters

*f* The file to assign to that file descriptor.

#### Returns

the allocated file descriptor, or -EMFILE on error.

**11.66.2.3 virtual cxx::Ref\_ptr<File> L4Re::Vfs::Fs::set\_fd ( int *fd*, cxx::Ref\_ptr< File > const  
& *f* = cxx::Ref\_ptr<>::Nil ) throw () [pure virtual]**

Set the file object referenced by the file descriptor *fd*.

#### Parameters

*fd* The file descriptor to set to *f*;

*f* The file object to assign.

#### Returns

A pointer to the file object that was previously assigned to fd.

**11.66.2.4 virtual cxx::Ref\_ptr<File> L4Re::Vfs::Fs::free\_fd ( int *fd* ) throw () [pure  
virtual]**

Free the file descriptor *fd*.

#### Parameters

*fd* The file descriptor to free.

#### Returns

A pointer to the file object that was assigned to the fd.

**11.66.2.5 int L4Re::Vfs::Fs::mount ( char const \* *path*, cxx::Ref\_ptr< File > const & *dir* )  
throw () [inline]**

Mount a given file object at the given global path in the VFS.

#### Parameters

*path* The global path to mount *dir* at.

*dir* A pointer to the file/directory object that shall be mounted at *path*.

#### Returns

0 on success, or <0 on error.

Definition at line 940 of file [vfs.h](#).

References [get\\_root\(\)](#).

Referenced by [mount\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

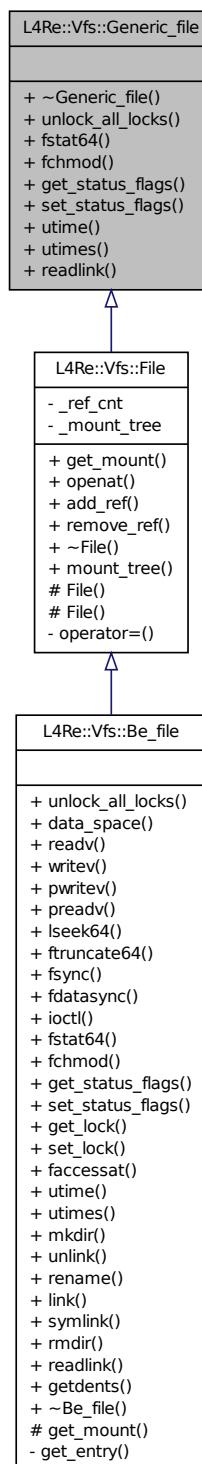
- [14/l4re\\_vfs/vfs.h](#)

## 11.67 L4Re::Vfs::Generic\_file Class Reference

The common interface for an open POSIX file.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Generic\_file:



## Public Member Functions

- virtual int [unlock\\_all\\_locks](#) ()=0 throw()  
*Unlock all locks on the file.*
- virtual int [fstat64](#) (struct stat64 \*buf) const =0 throw()  
*Get status information for the file.*
- virtual int [fchmod](#) (mode\_t)=0 throw()  
*Change POSIX access rights on that file.*
- virtual int [get\\_status\\_flags](#) () const =0 throw()  
*Get file status flags (fcntl F\_GETFL).*
- virtual int [set\\_status\\_flags](#) (long flags)=0 throw()  
*Set file status flags (fcntl F\_SETFL).*

### 11.67.1 Detailed Description

The common interface for an open POSIX file. This interface is common to all kinds of open files, independent of the the file type (e.g., directory, regular file etc.). However, in the [L4Re::Vfs](#) the interface [File](#) is used for every real object.

#### See also

[L4Re::Vfs::File](#) for mor information.

Definition at line [62](#) of file [vfs.h](#).

### 11.67.2 Member Function Documentation

#### 11.67.2.1 virtual int L4Re::Vfs::Generic\_file::unlock\_all\_locks( ) throw() [pure virtual]

Unlock all locks on the file.

#### Note

All locks means all locks independent by which file the locks were taken.

This method is called by the POSIX close implementation to get the POSIX semantics of releasing all locks taken by this application on a close for any fd referencing the real file.

#### Returns

0 on success, or <0 on error.

**11.67.2.2 virtual int L4Re::Vfs::Generic\_file::fstat64 ( struct stat64 \* *buf* ) const throw () [pure virtual]**

Get status information for the file.

This is the backend for POSIX fstat, stat, fstat64 and friends.

**Returns**

*buf* This buffer is filled with the status information.

**Returns**

0 on success, or <0 on error.

**11.67.2.3 virtual int L4Re::Vfs::Generic\_file::fchmod ( mode\_t ) throw () [pure virtual]**

Change POSIX access rights on that file.

Backend for POSIX chmod and fchmod.

**11.67.2.4 virtual int L4Re::Vfs::Generic\_file::get\_status\_flags ( ) const throw () [pure virtual]**

Get file status flags (fcntl F\_GETFL).

This function is used by the fcntl implementation for the F\_GETFL command).

**Returns**

flags such as O\_RDONLY, O\_WRONLY, O\_RDWR, O\_DIRECT, O\_ASYNC, O\_NOATIME, O\_NONBLOCK, or <0 on error.

**11.67.2.5 virtual int L4Re::Vfs::Generic\_file::set\_status\_flags ( long *flags* ) throw () [pure virtual]**

Set file status flags (fcntl F\_SETFL).

This function is used by the fcntl implementation for the F\_SETFL command).

**Parameters**

*flags* The file status flags to set. This must be a combination of O\_RDONLY, O\_WRONLY, O\_RDWR, O\_APPEND, O\_ASYNC, O\_DIRECT, O\_NOATIME, O\_NONBLOCK.

**Note**

Creation flags such as O\_CREAT, O\_EXCL, O\_NOCTTY, O\_TRUNC are ignored.

**Returns**

0 on success, or <0 on error.

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

- l4/re vfs/vfs.h

## 11.68 gfxbitmap\_offset Struct Reference

offsets in pmap[] and bmap[]

```
#include <bitmap.h>
```

### Data Fields

- `l4_uint32_t preskip_x`

*skip pixels at beginning of line*

- `l4_uint32_t preskip_y`

*skip lines*

- `l4_uint32_t endskip_x`

*skip pixels at end of line*

### 11.68.1 Detailed Description

offsets in pmap[] and bmap[]

Definition at line [67](#) of file [bitmap.h](#).

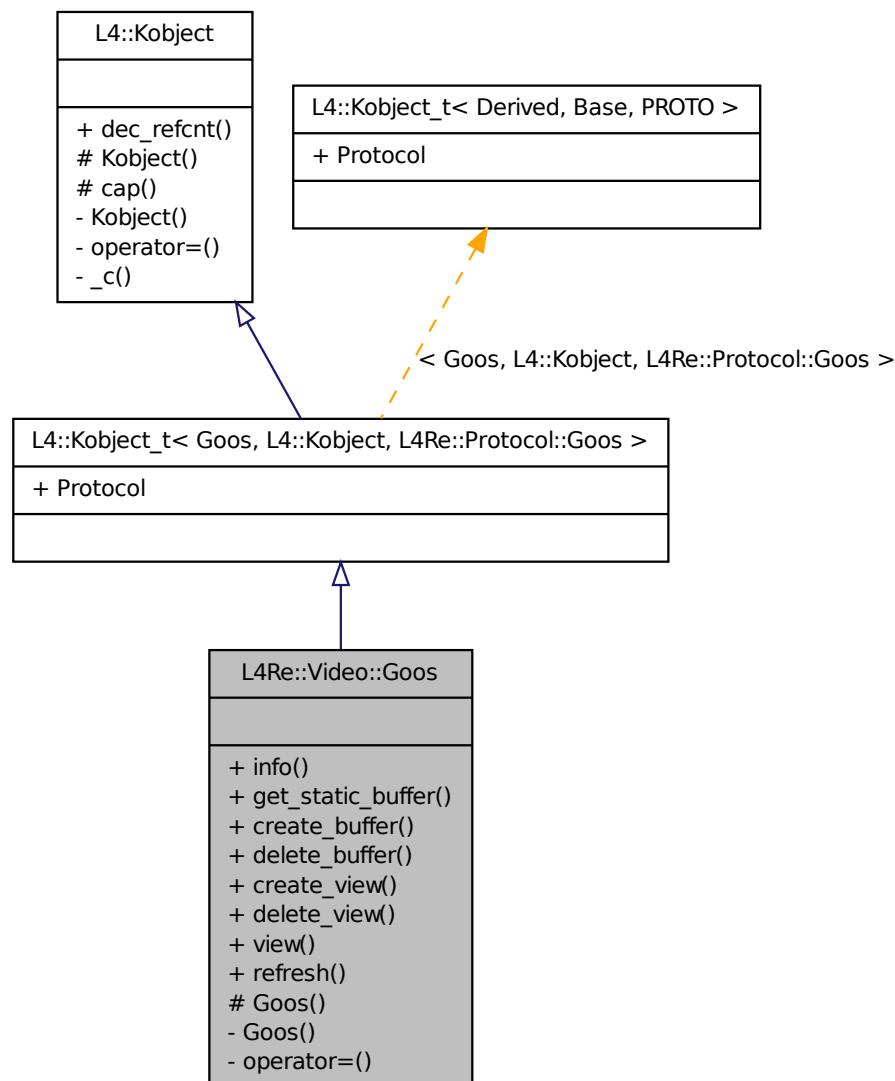
The documentation for this struct was generated from the following file:

- `l4/libgfxbitmap(bitmap.h)`

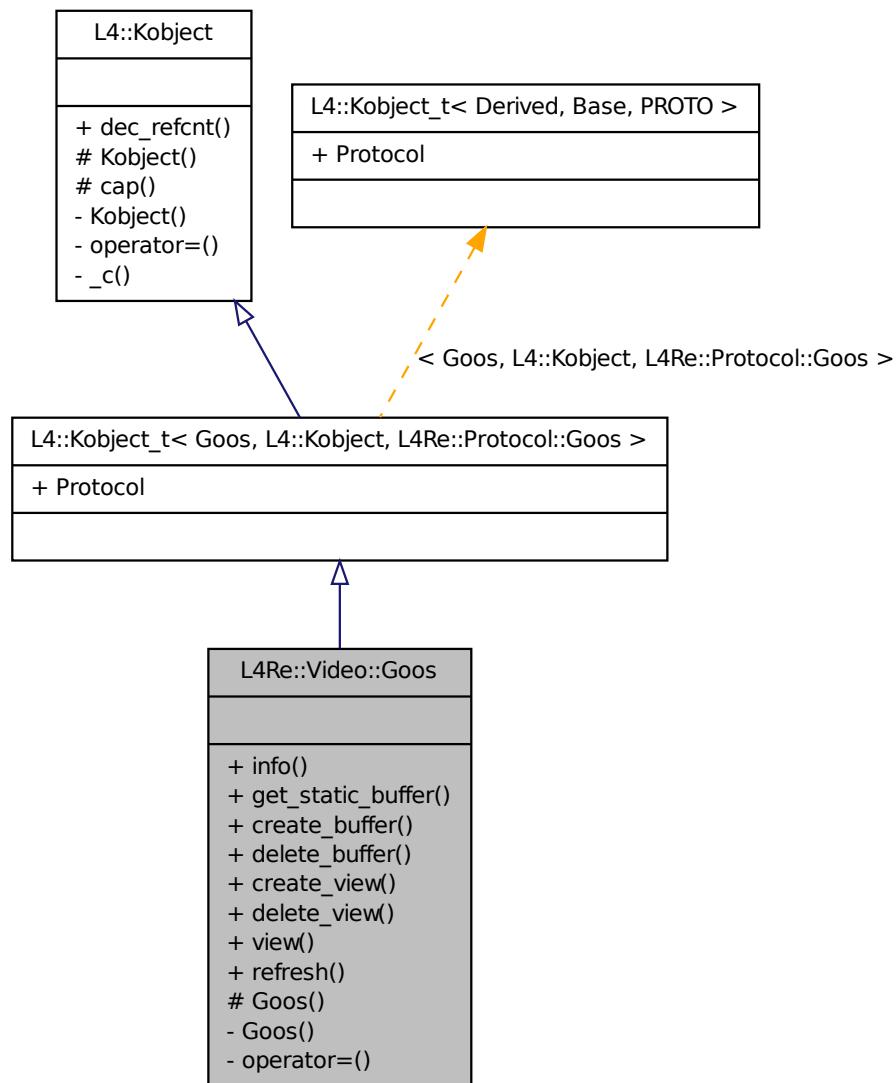
## 11.69 L4Re::Video::Goos Class Reference

A goos.

Inheritance diagram for L4Re::Video::Goos:



Collaboration diagram for L4Re::Video::Goos:



## Data Structures

- struct [Info](#)

*Information structure of a goos.*

## Public Types

- enum **Flags** { **F\_auto\_refresh** = 0x01, **F\_pointer** = 0x02, **F\_dynamic\_views** = 0x04, **F\_dynamic\_buffers** = 0x08 }

*Flags for a goos.*

## Public Member Functions

- int **info** (**Info** \*) const throw ()  
*Return the goos information of the goos.*
- int **get\_static\_buffer** (unsigned idx, **L4::Cap< L4Re::Dataspace >** rbuf) const throw ()  
*Return a static buffer of a goos.*
- int **create\_buffer** (unsigned long size, **L4::Cap< L4Re::Dataspace >** rbuf) const throw ()  
*Create a buffer.*
- int **delete\_buffer** (unsigned idx) const throw ()  
*Delete a buffer.*
- int **create\_view** (**View** \*view) const throw ()  
*Create a view.*
- int **delete\_view** (**View** const &v) const throw ()  
*Delete a view.*
- **View** **view** (unsigned index) const throw ()  
*Return a view.*
- int **refresh** (int x, int y, int w, int h) throw ()  
*Trigger refreshing of the given area on the virtual screen.*

### 11.69.1 Detailed Description

A goos.

Definition at line 39 of file [goos](#).

### 11.69.2 Member Enumeration Documentation

#### 11.69.2.1 enum L4Re::Video::Goos::Flags

Flags for a goos.

##### Enumerator:

**F\_auto\_refresh** The graphics display is automatically refreshed.

**F\_pointer** We have a mouse pointer.

*F\_dynamic\_views* Supports dynamically allocated views.

*F\_dynamic\_buffers* Supports dynamically allocated buffers.

Definition at line 46 of file [goos](#).

### 11.69.3 Member Function Documentation

#### 11.69.3.1 int L4Re::Video::Goos::info ( Info \* ) const throw ()

Return the goos information of the goos.

##### Return values

*info* Goos information structure pointer.

##### Returns

0 on success, error otherwise

#### 11.69.3.2 int L4Re::Video::Goos::get\_static\_buffer ( unsigned idx, L4::Cap< L4Re::Dataspace > rbuf ) const throw ()

Return a static buffer of a goos.

##### Parameters

*idx* Index of the static buffer.

*rbuf* Capability slot to point the buffer dataspace to.

##### Returns

0 on success, error otherwise

#### 11.69.3.3 int L4Re::Video::Goos::create\_buffer ( unsigned long size, L4::Cap< L4Re::Dataspace > rbuf ) const throw ()

Create a buffer.

##### Parameters

*size* Size of buffer in bytes.

*rbuf* Capability slot to point the buffer dataspace to.

##### Returns

Positive: buffer index, negative: Error code

**11.69.3.4 int L4Re::Video::Goos::delete\_buffer ( unsigned *idx* ) const throw ()**

Delete a buffer.

**Parameters**

*idx* Buffer to delete.

**Returns**

0 on success, error otherwise

**11.69.3.5 int L4Re::Video::Goos::create\_view ( View \* *view* ) const throw ()**

Create a view.

**Return values**

*view* A view object.

**Returns**

Positive: view index, negative: Error code

**11.69.3.6 int L4Re::Video::Goos::delete\_view ( View const & *v* ) const throw ()**

Delete a view.

**Parameters**

*v* The view object to delete.

**Returns**

0 on success, error otherwise

**11.69.3.7 View L4Re::Video::Goos::view ( unsigned *index* ) const throw () [inline]**

Return a view.

**Parameters**

*index* Index of the view to return.

**Returns**

The view.

Definition at line 133 of file [goos](#).

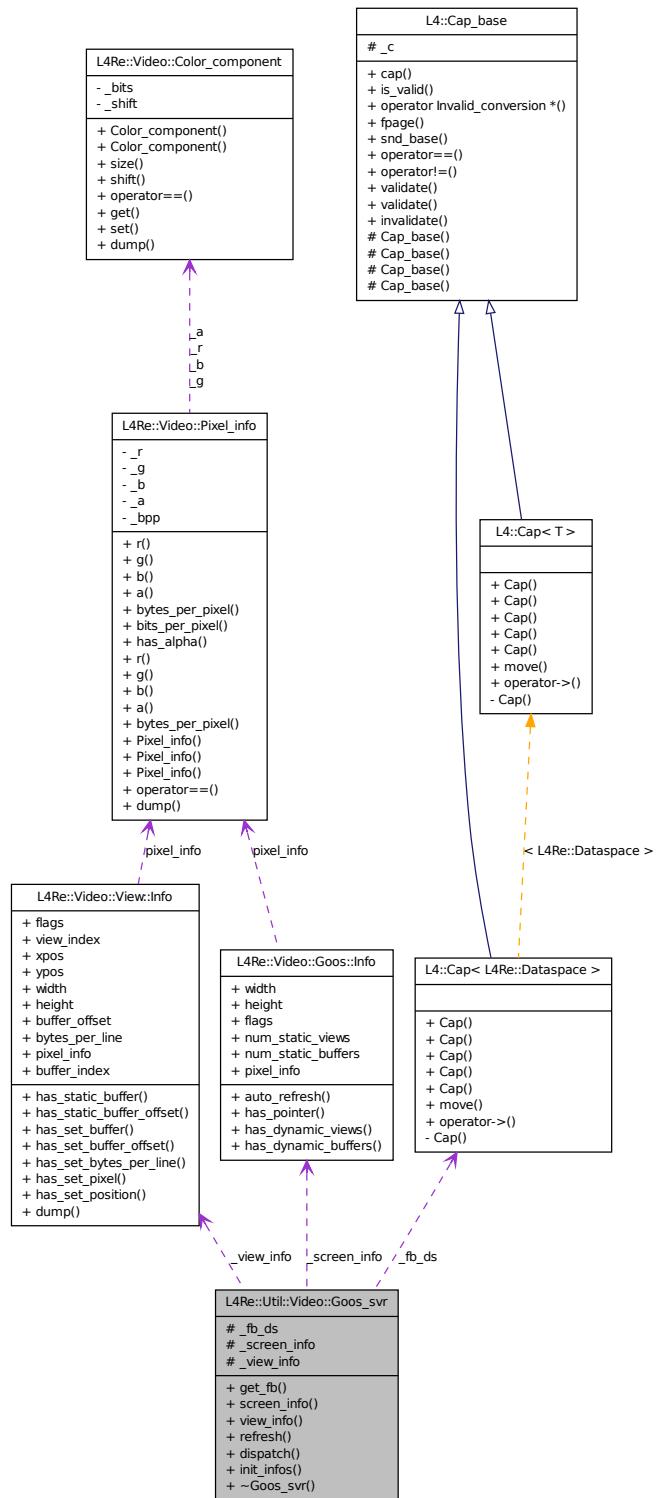
The documentation for this class was generated from the following file:

- [l4/re/video/goos](#)

## 11.70 L4Re::Util::Video::Goos\_svr Class Reference

Goos server class.

Collaboration diagram for L4Re::Util::Video::Goos\_svr:



## Public Member Functions

- `L4::Cap< L4Re::Dataspace > get_fb () const`  
*Return framebuffer memory dataspace.*
- `L4Re::Video::Goos::Info const * screen_info () const`  
*Goos information structure.*
- `L4Re::Video::View::Info const * view_info () const`  
*View information structure.*
- virtual int `refresh (int x, int y, int w, int h)`  
*Refresh area of the framebuffer.*
- int `dispatch (l4_umword_t obj, L4::Ipc::Iostream &ios)`  
*Server dispatch function.*
- void `init_infos ()`  
*Initialize the view information structure of this object.*
- virtual `~Goos_svr ()`  
*Destructor.*

## Protected Attributes

- `L4::Cap< L4Re::Dataspace > _fb_ds`  
*Goos memory dataspace.*
- `L4Re::Video::Goos::Info _screen_info`  
*Goos information.*
- `L4Re::Video::View::Info _view_info`  
*View information.*

### 11.70.1 Detailed Description

Goos server class.

Definition at line 33 of file `goos_svr`.

### 11.70.2 Member Function Documentation

#### 11.70.2.1 `L4::Cap<L4Re::Dataspace> L4Re::Util::Video::Goos_svr::get_fb ( ) const [inline]`

Return framebuffer memory dataspace.

**Returns**

Goos memory dataspace

Definition at line 41 of file [goos\\_svr](#).

**11.70.2.2 L4Re::Video::Goos::Info const\* L4Re::Util::Video::Goos\_svr::screen\_info( ) const [inline]**

Goos information structure.

**Returns**

Return goos information structure.

Definition at line 47 of file [goos\\_svr](#).

**11.70.2.3 L4Re::Video::View::Info const\* L4Re::Util::Video::Goos\_svr::view\_info( ) const [inline]**

View information structure.

**Returns**

Return view information structure.

Definition at line 53 of file [goos\\_svr](#).

References [\\_fb\\_ds](#).

**11.70.2.4 virtual int L4Re::Util::Video::Goos\_svr::refresh( int x, int y, int w, int h ) [inline, virtual]**

Refresh area of the framebuffer.

**Parameters**

*x* X coordinate (pixels)

*y* Y coordinate (pixels)

*w* Width of area in pixels

*h* Height of area in pixels

**Returns**

0 on success, negative error code otherwise

Definition at line 65 of file [goos\\_svr](#).

References [\\_view\\_info](#).

**11.70.2.5 int L4Re::Util::Video::Goos\_svr::dispatch ( l4\_umword\_t obj, L4::Ipc::Iostream & ios ) [inline]**

Server dispatch function.

**Parameters**

*obj* Server object ID to work on

*ios* Input/Output stream.

**Returns**

error code.

Definition at line 116 of file [goos\\_svr](#).

**11.70.2.6 void L4Re::Util::Video::Goos\_svr::init\_infos ( ) [inline]**

Initialize the view information structure of this object.

This function initializes the view info structure of this goos object based on the information in the goos information, i.e. the width, height and pixel\_info of the goos information has to contain valid values before calling init\_info().

Definition at line 86 of file [goos\\_svr](#).

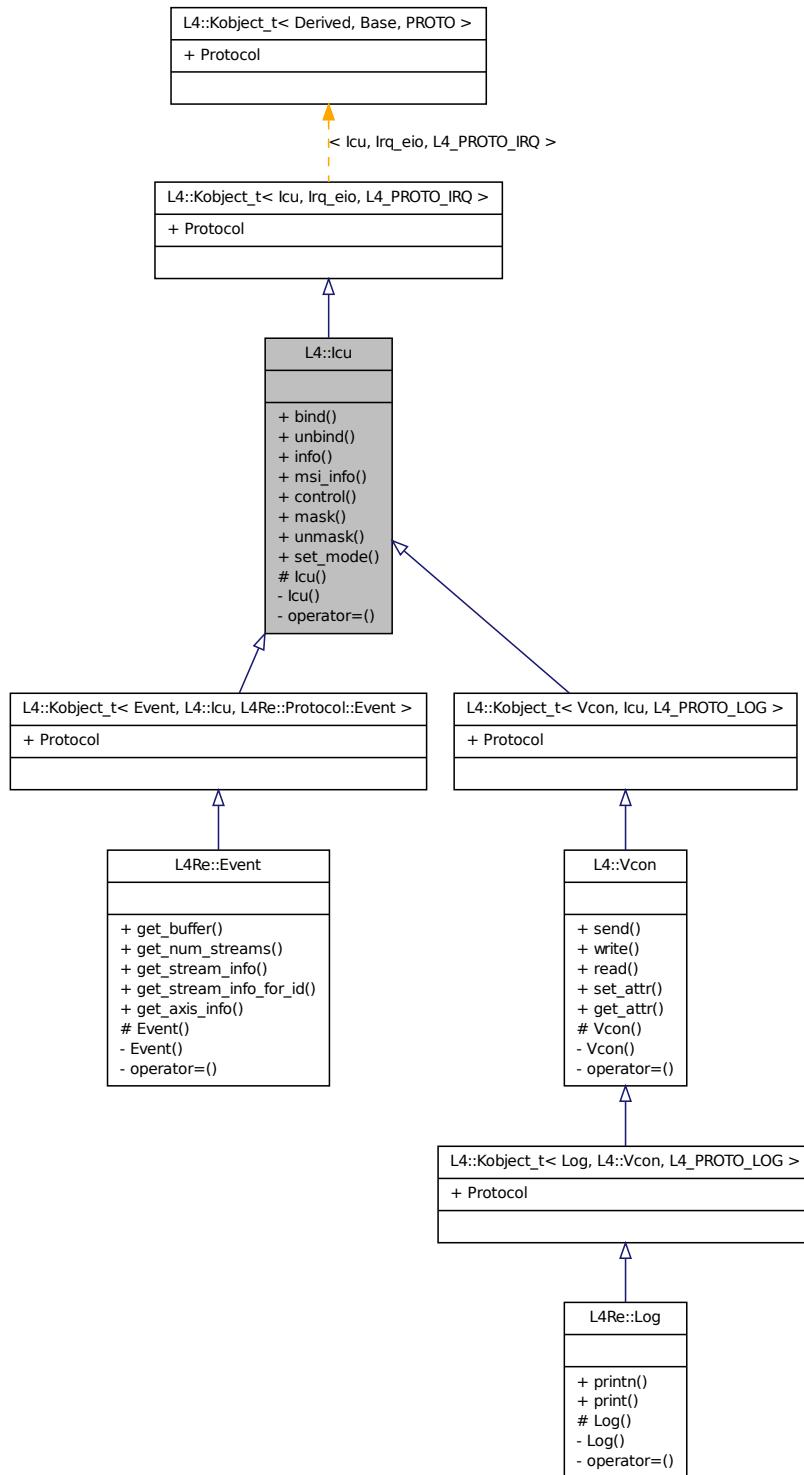
The documentation for this class was generated from the following file:

- l4/re/util/video/goos\_svr

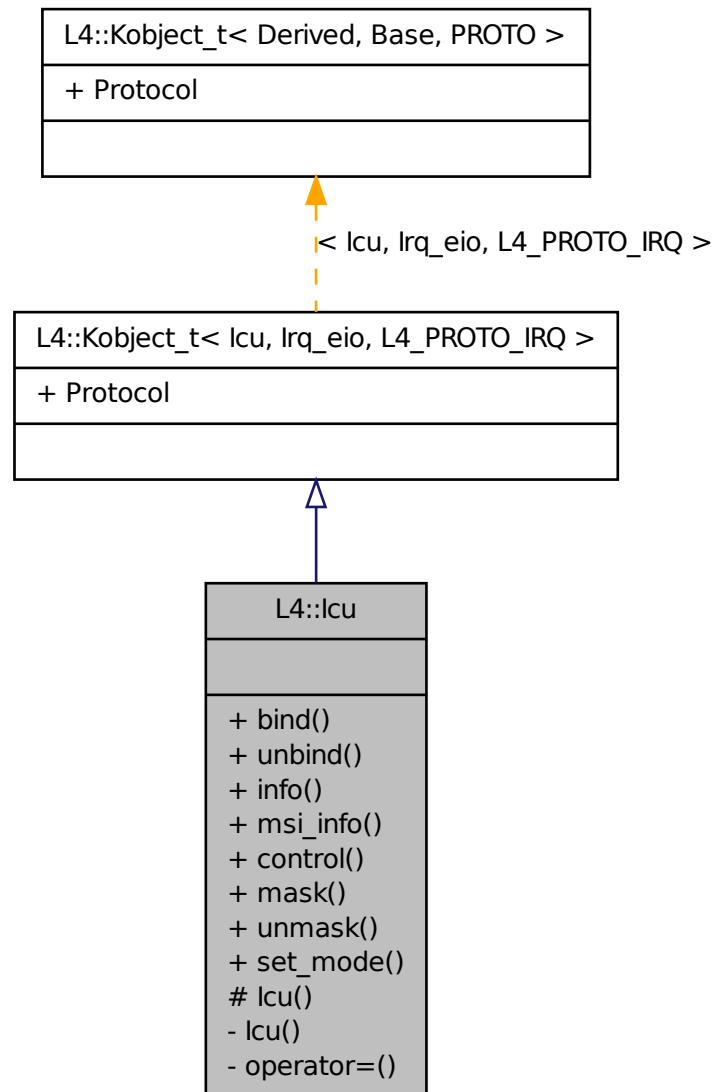
## 11.71 L4::Icu Class Reference

C++ version of an interrupt controller.

Inheritance diagram for L4::Icu:



## Collaboration diagram for L4::Icu:



## Data Structures

- class Info

*Info for an ICU.*

## Public Member Functions

- [l4\\_mshtag\\_t bind](#) (unsigned irqnum, [L4::Cap<Irq>](#) irq, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) throw ()
- [l4\\_mshtag\\_t unbind](#) (unsigned irqnum, [L4::Cap<Irq>](#) irq, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) throw ()
- [l4\\_mshtag\\_t info](#) ([l4\\_icu\\_info\\_t](#) \*info, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) throw ()
- [l4\\_mshtag\\_t msi\\_info](#) (unsigned irqnum, [l4\\_umword\\_t](#) \*msg, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) throw ()
- [l4\\_mshtag\\_t mask](#) (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=[L4\\_IPC\\_NEVER](#), [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) throw ()
- [l4\\_mshtag\\_t unmask](#) (unsigned irqnum, [l4\\_umword\\_t](#) \*label=0, [l4\\_timeout\\_t](#) to=[L4\\_IPC\\_NEVER](#), [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) throw ()
- [l4\\_mshtag\\_t set\\_mode](#) (unsigned irqnum, [l4\\_umword\\_t](#) mode, [l4\\_utcb\\_t](#) \*utcb=[l4\\_utcb\(\)](#)) throw ()

### 11.71.1 Detailed Description

C++ version of an interrupt controller. #include <l4/sys/icu>

#### See also

[Interrupt controller](#) for an overview and C bindings.

Definition at line [125](#) of file [irq](#).

### 11.71.2 Member Function Documentation

#### 11.71.2.1 [l4\\_mshtag\\_t L4::Icu::bind\( unsigned irqnum, L4::Cap<Irq> irq, l4\\_utcb\\_t \\* utcb = l4\\_utcb\(\) \) throw\(\) \[inline\]](#)

Bind an interrupt vector of an interrupt controller to an interrupt object.

#### Parameters

*icu* ICU to use.

*irqnum* IRQ vector at the ICU.

*irq* IRQ capability to bind the IRQ to.

#### Returns

Syscall return tag

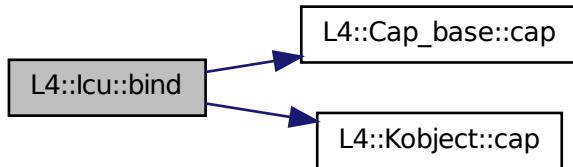
#### Note

*the icu argument is the implicit this pointer.*

Definition at line [162](#) of file [irq](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



**11.71.2.2 l4\_mshtag\_t L4::Icu::unbind ( unsigned irqnum, L4::Cap< Irq > irq, l4\_utcb\_t \* utcb = *l4\_utcb()* ) throw () [inline]**

Remove binding of an interrupt vector from the interrupt controller object.

#### Parameters

- icu* ICU to use.
- irqnum* IRQ vector at the ICU.
- irq* IRQ object to remove from the ICU.

#### Returns

Syscall return tag

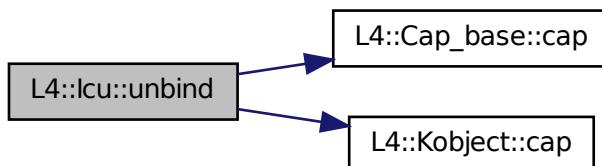
#### Note

*the icu argument is the implicit *this* pointer.*

Definition at line 170 of file [irq.h](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



**11.71.2.3 l4\_mshtag\_t L4::Icu::info ( l4\_icu\_info\_t \* info, l4\_utcb\_t \* utcb = `l4_utcb()` )  
throw () [inline]**

Get info about capabilites of ICU.

#### Parameters

*icu* ICU to use.

*info* Pointer to an info structure to be filled with information.

#### Returns

Syscall return tag

#### Note

*the icu argument is the implicit this pointer.*

Definition at line 178 of file [irq](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



**11.71.2.4 l4\_mshtag\_t L4::Icu::msi\_info ( unsigned irqnum, l4\_umword\_t \* msg, l4\_utcb\_t \* utcb = `l4_utcb()` ) throw () [inline]**

Get MSI info about IRQ.

#### Parameters

*icu* ICU to use.

*irqnum* IRQ vector at the ICU.

*msg* Pointer to a word to receive the message that must be used for the PCI devices MSI message.

#### Returns

Syscall return tag

#### Note

*the icu argument is the implicit this pointer.*

Definition at line 185 of file [irq](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### **11.71.2.5 l4\_mshtag\_t L4::Icu::mask ( unsigned irqnum, l4\_umword\_t \* label = 0, l4\_timeout\_t to = L4\_IPC\_NEVER, l4\_utcb\_t \* utcb = l4\_utcb() ) throw () [inline]**

Mask an IRQ vector.

#### Parameters

*icu* ICU to use.

*irqnum* IRQ vector at the ICU.

*label* If non-NULL the function also waits for the next message.

*to* Timeout for message to ICU, if unsure use L4\_IPC\_NEVER.

#### Returns

Syscall return tag

#### Note

*the icu argument is the implicit this pointer.*

Definition at line 200 of file [irq](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



---

**11.71.2.6 l4\_msntag\_t L4::Icu::unmask ( *unsigned irqnum*, *l4\_umword\_t \* label* = 0, *l4\_timeout\_t to* = *L4\_IPC\_NEVER*, *l4\_utcb\_t \* utcb* = *L4\_utcb()* ) throw () [inline]**

Unmask an IRQ vector.

#### Parameters

*icu* ICU to use.

*irqnum* IRQ vector at the ICU.

*label* If non-NULL the function also waits for the next message.

*to* Timeout for message to ICU, if unsure use L4\_IPC\_NEVER.

#### Returns

Syscall return tag

#### Note

*the icu argument is the implicit this pointer.*

Definition at line 210 of file [irq](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



**11.71.2.7 l4\_msntag\_t L4::Icu::set\_mode ( *unsigned irqnum*, *l4\_umword\_t mode*, *l4\_utcb\_t \* utcb* = *L4\_utcb()* ) throw () [inline]**

Set mode of interrupt.

#### Parameters

*icu* ICU to use.

*irqnum* IRQ vector at the ICU.

*mode* Mode, see L4\_irq\_flow\_type.

#### Returns

Syscall return tag

**Note**

*the icu argument is the implicit *this* pointer.*

Definition at line 220 of file [irq](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



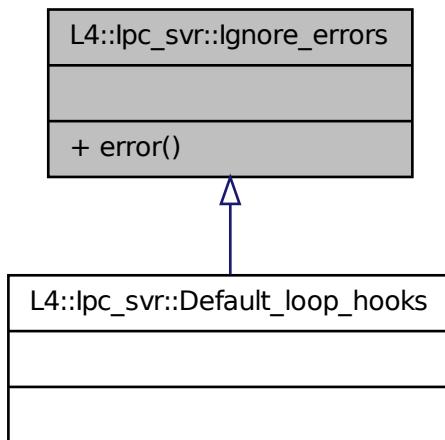
The documentation for this class was generated from the following file:

- [l4/sys/irq](#)

## 11.72 L4::Ipc\_svr::Ignore\_errors Struct Reference

Mix in for LOOP\_HOOKS to ignore IPC errors.

Inheritance diagram for L4::Ipc\_svr::Ignore\_errors:



### 11.72.1 Detailed Description

Mix in for LOOP\_HOOKS to ignore IPC errors.

Definition at line 53 of file [ipc\\_server](#).

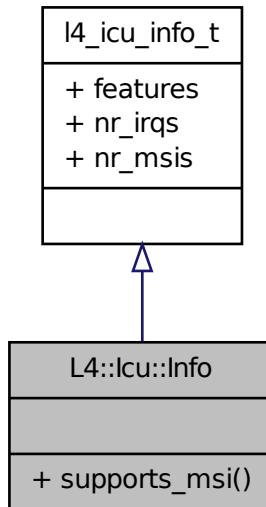
The documentation for this struct was generated from the following file:

- l4/cxx/ipc\_server

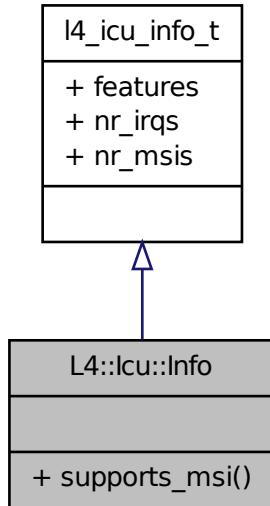
## 11.73 L4::Icu::Info Class Reference

[Info](#) for an ICU.

Inheritance diagram for L4::Icu::Info:



Collaboration diagram for L4::Icu::Info:



### 11.73.1 Detailed Description

`Info` for an ICU. This class adds access functions.

#### See also

[l4\\_icu\\_info\(\)](#).

Definition at line [152](#) of file [irq](#).

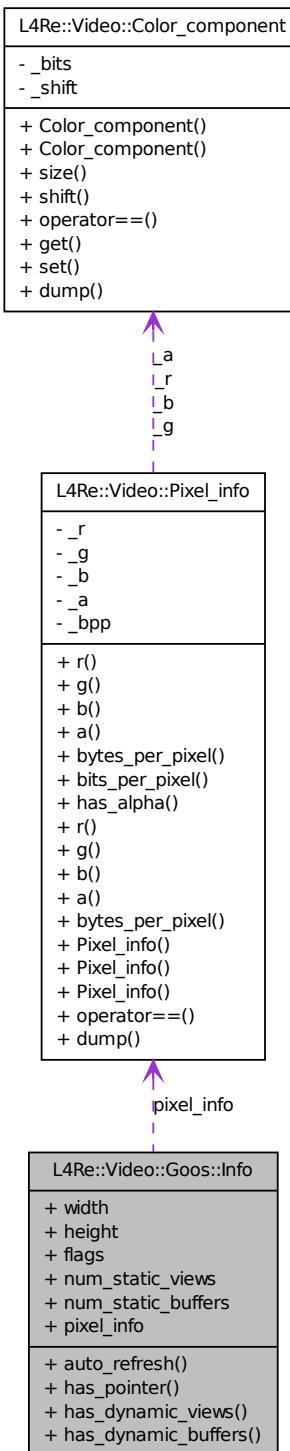
The documentation for this class was generated from the following file:

- [l4/sys/irq](#)

## 11.74 L4Re::Video::Goos::Info Struct Reference

Information structure of a goos.

Collaboration diagram for L4Re::Video::Goos::Info:



## Public Member Functions

- `bool auto_refresh () const`  
*Return whether this goos does auto refreshing or the view refresh functions must be used to make changes visible.*
- `bool has_pointer () const`  
*Return whether a pointer is used by the provider of the goos.*
- `bool has_dynamic_views () const`  
*Return whether dynamic view are supported.*
- `bool has_dynamic_buffers () const`  
*Return whether dynamic buffers are supported.*

## Data Fields

- `unsigned long width`  
*Width.*
- `unsigned long height`  
*Height.*
- `unsigned flags`  
*Flags, see Flags.*
- `unsigned num_static_views`  
*Number of static view.*
- `unsigned num_static_buffers`  
*Number of static buffers.*
- `Pixel_info pixel_info`  
*Pixel information.*

### 11.74.1 Detailed Description

Information structure of a goos.

Definition at line [55](#) of file [goos](#).

### 11.74.2 Member Function Documentation

#### 11.74.2.1 `bool L4Re::Video::Goos::Info::auto_refresh ( ) const [inline]`

Return whether this goos does auto refreshing or the view refresh functions must be used to make changes visible.

Definition at line [66](#) of file [goos](#).

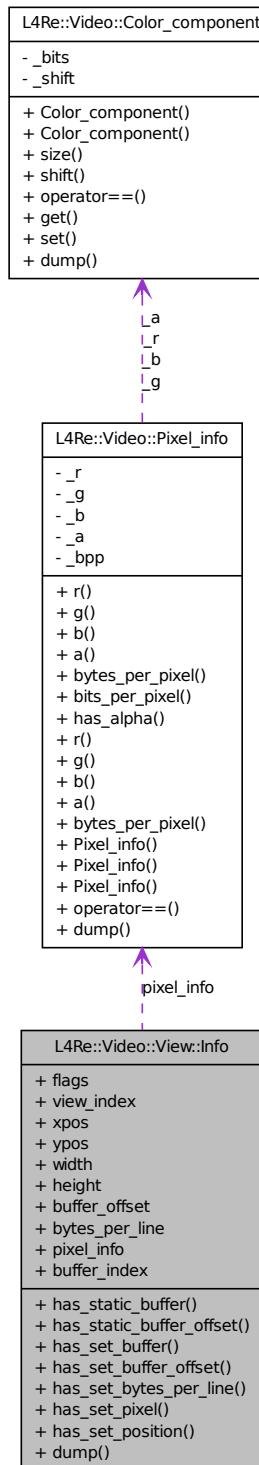
The documentation for this struct was generated from the following file:

- [l4/re/video/goos](#)

## **11.75 L4Re::Video::View::Info Struct Reference**

Information structure of a view.

Collaboration diagram for L4Re::Video::View::Info:



## Public Member Functions

- `bool has_static_buffer () const`  
*Return whether the view has a static buffer.*
- `bool has_static_buffer_offset () const`  
*Return whether the static buffer offset is available.*
- `bool has_set_buffer () const`  
*Return whether a buffer is set.*
- `bool has_set_buffer_offset () const`  
*Return whether the given buffer offset is valid.*
- `bool has_set_bytes_per_line () const`  
*Return whether the given bytes-per-line value is valid.*
- `bool has_set_pixel () const`  
*Return whether the given pixel information is valid.*
- `bool has_set_position () const`  
*Return whether the position information given is valid.*
- template<typename STREAM >  
`STREAM & dump (STREAM &s) const`  
*Dump information on the view information to a stream.*

## Data Fields

- `unsigned flags`  
*Flags.,*
- `unsigned view_index`  
*Index of the view.*
- `unsigned long xpos`  
*X position in pixels of the view in the goos.*
- `unsigned long ypos`  
*Y position in pixels of the view in the goos.*
- `unsigned long width`  
*Width of the view in pixels.*
- `unsigned long height`  
*Height of the view in pixels.*
- `unsigned long buffer_offset`  
*Offset in the memory buffer in bytes.*

- `unsigned long bytes_per_line`

*Bytes per line.*

- `Pixel_info pixel_info`

*Pixel information.*

- `unsigned buffer_index`

*Number of the buffer used for this view.*

### 11.75.1 Detailed Description

Information structure of a view.

Definition at line 85 of file `view`.

### 11.75.2 Field Documentation

#### 11.75.2.1 `unsigned L4Re::Video::View::Info::flags`

Flags,.

##### See also

`Flags` and `V_flags`

Definition at line 87 of file `view`.

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

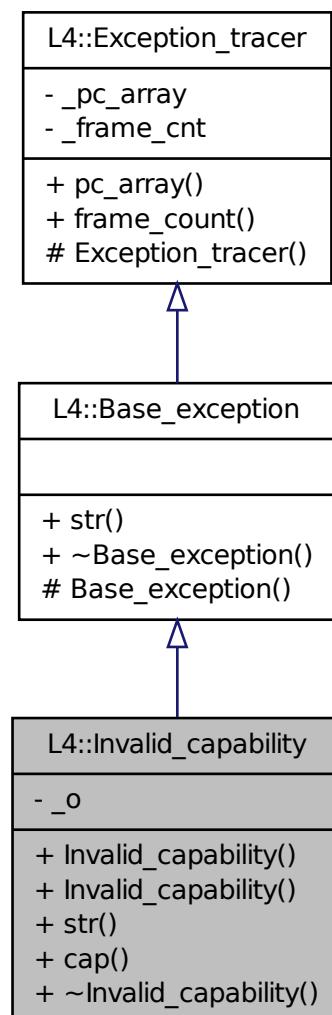
- `l4/re/video/view`

## 11.76 `L4::Invalid_capability` Class Reference

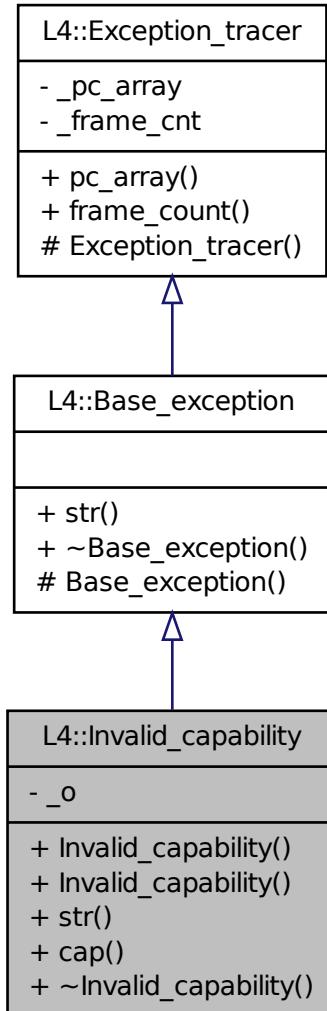
Indicates that an invalid object was invoked.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Invalid\_capability:



Collaboration diagram for L4::Invalid\_capability:



## Public Member Functions

- **Invalid\_capability (Cap< void > const &o) throw ()**  
*Create an Invalid\_object exception for the Object o.*
- **char const \* str () const throw ()**  
*Should return a human readable string for the exception.*
- **Cap< void > const & cap () const throw ()**  
*Get the object that caused the error.*

## 11.76.1 Detailed Description

Indicates that an invalid object was invoked. An Object is invalid if it has L4\_INVALID\_ID as server L4 UID, or if the server does not know the object ID.

Definition at line 220 of file [exceptions](#).

## 11.76.2 Constructor & Destructor Documentation

### 11.76.2.1 L4::Invalid\_capability::Invalid\_capability ( Cap< void > const & o ) throw () [inline, explicit]

Create an Invalid\_obejct exception for the Object o.

#### Parameters

- o* The object that caused the server side error.

Definition at line 230 of file [exceptions](#).

## 11.76.3 Member Function Documentation

### 11.76.3.1 Cap<void> const& L4::Invalid\_capability::cap ( ) const throw () [inline]

Get the object that caused the error.

#### Returns

- The object that caused the error on invocation.

Definition at line 239 of file [exceptions](#).

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

- l4/cxx/exceptions

## 11.77 L4::IOModifier Class Reference

Modifier class for the IO stream.

## 11.77.1 Detailed Description

Modifier class for the IO stream. An IO Modifier can be used to change properties of an IO stream for example the number format.

Definition at line 33 of file [basic\\_ostream](#).

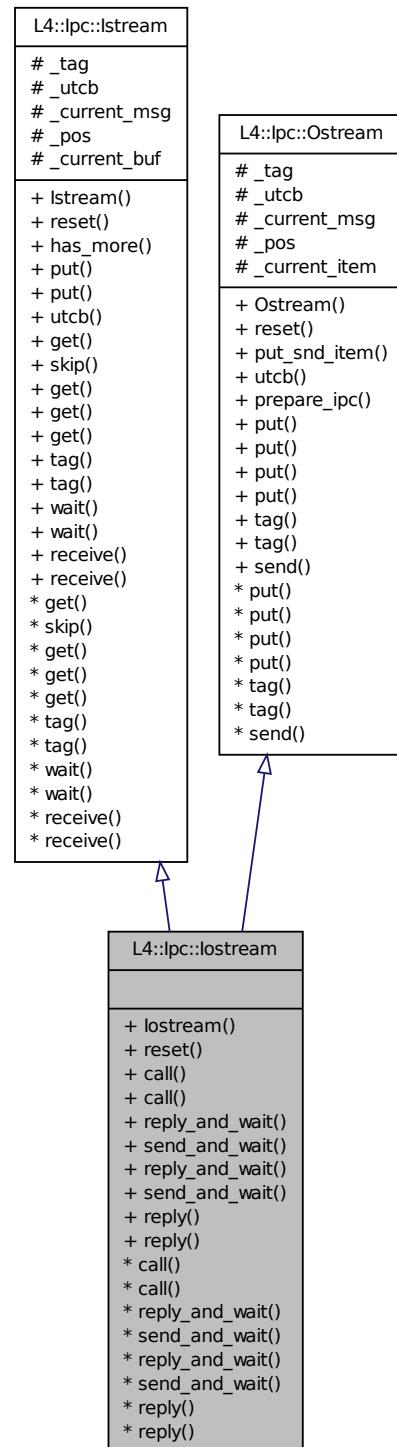
The documentation for this class was generated from the following file:

- l4/cxx/basic\_ostream

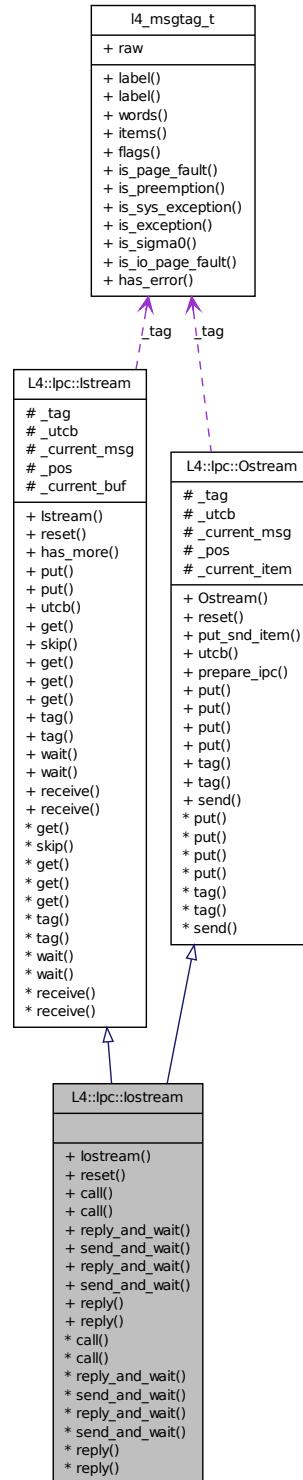
## 11.78 L4::Ipc::Iostream Class Reference

Input/Output stream for IPC [un]marshalling.

Inheritance diagram for L4::Ipc::Iostream:



Collaboration diagram for L4::Ipc::Iostream:



## Public Member Functions

- [Iostream \(l4\\_utcb\\_t \\*utcb\)](#)  
*Create an IPC IO stream with a single message buffer.*
- [void reset \(\)](#)  
*Reset the stream to its initial state.*

### IPC operations.

- [l4\\_mshtag\\_t call \(l4\\_cap\\_idx\\_t dst\)](#)  
*Do an IPC call using the message in the output stream and receiving to the input stream.*
- [l4\\_mshtag\\_t call \(l4\\_cap\\_idx\\_t dst, long label\)](#)
- [l4\\_mshtag\\_t reply\\_and\\_wait \(l4\\_umword\\_t \\*src\\_dst, long proto=0\)](#)  
*Do an IPC reply and wait.*
- [l4\\_mshtag\\_t send\\_and\\_wait \(l4\\_cap\\_idx\\_t dest, l4\\_umword\\_t \\*src, long proto=0\)](#)
- [l4\\_mshtag\\_t reply\\_and\\_wait \(l4\\_umword\\_t \\*src\\_dst, l4\\_timeout\\_t timeout, long proto=0\)](#)  
*Do an IPC reply and wait.*
- [l4\\_mshtag\\_t send\\_and\\_wait \(l4\\_cap\\_idx\\_t dest, l4\\_umword\\_t \\*src, l4\\_timeout\\_t timeout, long proto=0\)](#)
- [l4\\_mshtag\\_t reply \(l4\\_timeout\\_t timeout, long proto=0\)](#)
- [l4\\_mshtag\\_t reply \(long proto=0\)](#)

### 11.78.1 Detailed Description

Input/Output stream for IPC [un]marshalling. The [Ipc::Iostream](#) is part of the AW Env IPC framework as well as [Ipc::Istream](#) and [Ipc::Ostream](#). In particular an [Ipc::Iostream](#) is a combination of an [Ipc::Istream](#) and an [Ipc::Ostream](#). It can use either a single message buffer for receiving and sending messages or a pair of a receive and a send buffer. The stream also supports combined IPC operations such as [call\(\)](#) and [reply\\_and\\_wait\(\)](#), which can be used to implement RPC functionality.

#### Examples:

[examples/clntsrv/client.cc](#), [examples/clntsrv/server.cc](#), [examples/libs/l4re/c++/shared\\_ds/ds\\_srv.cc](#), [examples/libs/l4re/streammap/client.cc](#), and [examples/libs/l4re/streammap/server.cc](#).

Definition at line 995 of file [ipc\\_stream](#).

### 11.78.2 Constructor & Destructor Documentation

#### 11.78.2.1 L4::Ipc::Iostream::Iostream ( l4\_utcb\_t \* utcb ) [inline, explicit]

Create an IPC IO stream with a single message buffer.

##### Parameters

*msg* The message buffer used as backing store.

The created IO stream uses the same message buffer for sending and receiving IPC messages.

Definition at line 1006 of file [ipc\\_stream](#).

### 11.78.3 Member Function Documentation

#### 11.78.3.1 void L4::Ipc::Iostream::reset( ) [inline]

Reset the stream to its initial state.

Input as well as the output stream are reset.

Reimplemented from [L4::Ipc::Ostream](#).

Definition at line [1016](#) of file [ipc\\_stream](#).

#### 11.78.3.2 l4\_mshtag\_t L4::Ipc::Iostream::call( l4\_cap\_idx\_t dst ) [inline]

Do an IPC call using the message in the output stream and receiving to the input stream.

##### Parameters

*dst* The destination [L4](#) UID (thread) to call.

##### Returns

the result dope of the IPC operation.

This is a combined IPC operation consisting of a send and a receive to/from the given destination *dst*.

A call is usually used by clients for RPCs to a server.

##### Examples:

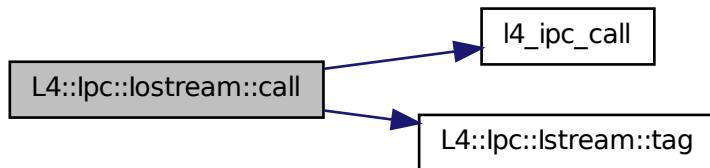
[examples/clntsrv/client.cc](#), and [examples/libs/l4re/streammap/client.cc](#).

Definition at line [1154](#) of file [ipc\\_stream](#).

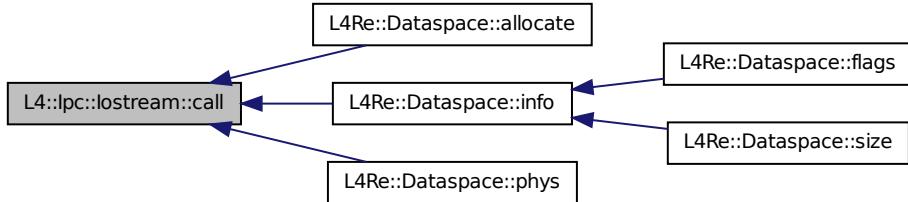
References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), and [L4::Ipc::Istream::tag\(\)](#).

Referenced by [L4Re::Dataspace::allocate\(\)](#), [L4Re::Dataspace::info\(\)](#), and [L4Re::Dataspace::phys\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 11.78.3.3 `l4_mshtag_t L4::Ipc::Iostream::reply_and_wait( l4_umword_t * src_dst, long proto = 0 ) [inline]`

Do an IPC reply and wait.

#### Parameters

`src_dst` Input: the destination for the send operation.

Output: the source of the received message.

#### Returns

the result dope of the IPC operation.

This is a combined IPC operation consisting of a send operation and an open wait for any message.

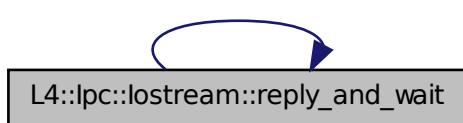
A reply and wait is usually used by servers that reply to a client and wait for the next request by any other client.

Definition at line 1071 of file `ipc_stream`.

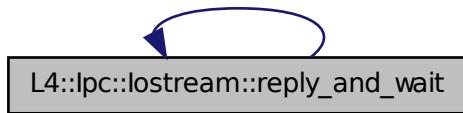
References `L4_IPC_SEND_TIMEOUT_0`, and `reply_and_wait()`.

Referenced by `reply_and_wait()`.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 11.78.3.4 ***l4\_mshtag\_t L4::Ipc::Iostream::reply\_and\_wait( l4\_umword\_t \* src\_dst, l4\_timeout\_t timeout, long proto = 0 ) [inline]***

Do an IPC reply and wait.

##### Parameters

*src\_dst* Input: the destination for the send operation.

Output: the source of the received message.

*timeout* Timeout used for IPC.

##### Returns

the result dope of the IPC operation.

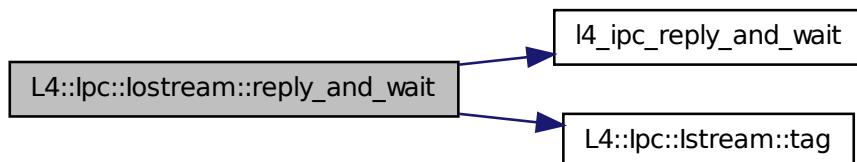
This is a combined IPC operation consisting of a send operation and an open wait for any message.

A reply and wait is usually used by servers that reply to a client and wait for the next request by any other client.

Definition at line 1175 of file [ipc\\_stream](#).

References [l4\\_ipc\\_reply\\_and\\_wait\(\)](#), and [L4::Ipc::Istream::tag\(\)](#).

Here is the call graph for this function:



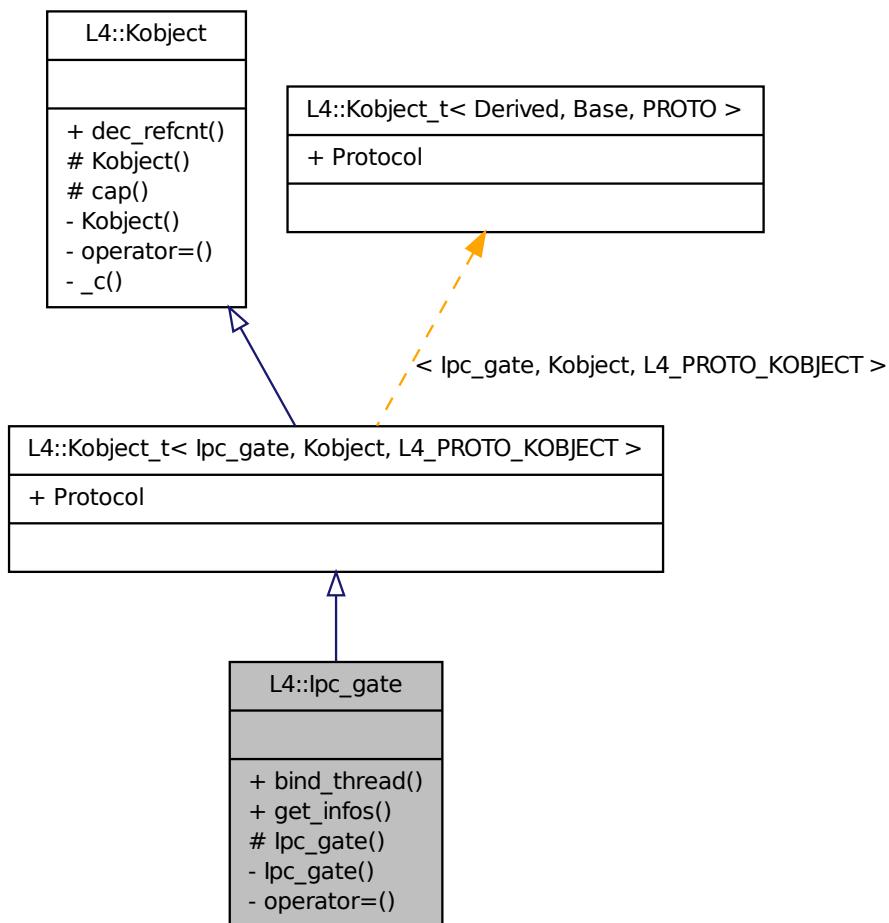
The documentation for this class was generated from the following file:

- l4/cxx/ipc\_stream

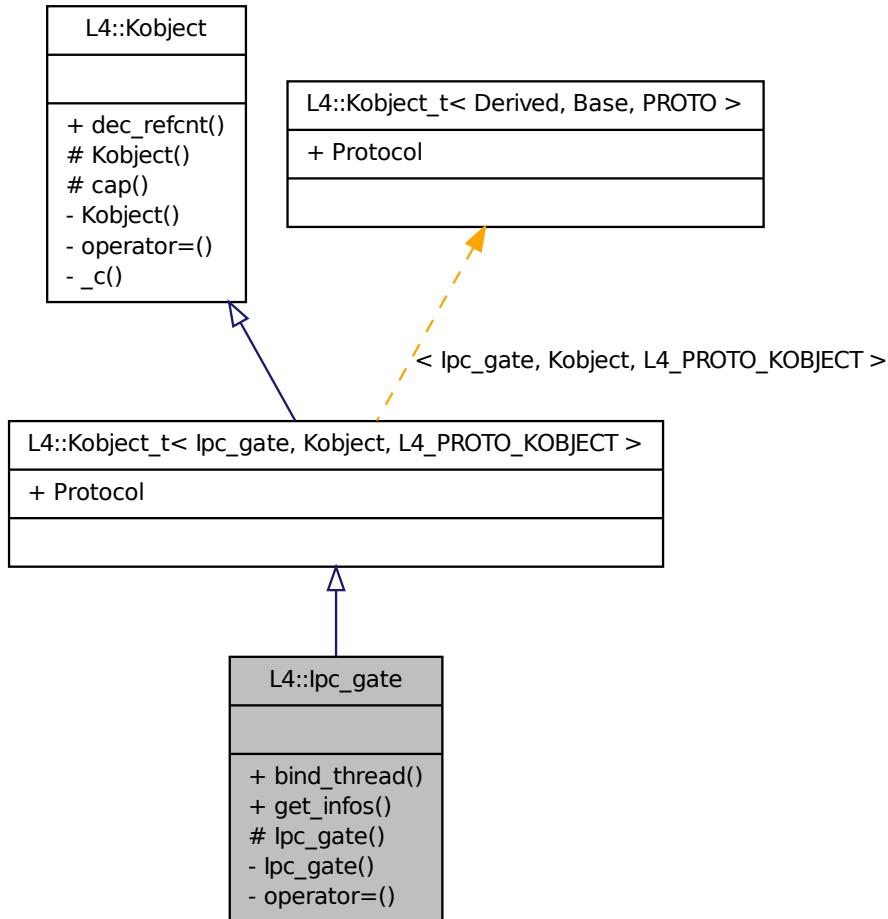
## 11.79 L4::Ipc\_gate Class Reference

[L4](#) IPC gate.

Inheritance diagram for L4::Ipc\_gate:



Collaboration diagram for L4::Ipc\_gate:



## Public Member Functions

- `l4_msgtag_t bind_thread (Cap< Thread > t, l4_umword_t label, l4_utcb_t *utcb=l4_utcb()) throw ()`  
*Bind the IPC-gate to the thread.*
- `l4_msgtag_t get_infos (l4_umword_t *label, l4_utcb_t *utcb=l4_utcb()) throw ()`  
*Get information on the IPC-gate.*

### 11.79.1 Detailed Description

[L4](#) IPC gate. #include <l4/sys/ipc\_gate>

Definition at line 39 of file [ipc\\_gate](#).

### 11.79.2 Member Function Documentation

**11.79.2.1 `l4_mshtag_t L4::Ipc_gate::bind_thread ( Cap< Thread > t, l4_umword_t label, l4_utcb_t * utcb = l4_utcb() ) throw () [inline]`**

Bind the IPC-gate to the thread.

#### See also

[l4\\_ipc\\_gate\\_bind\\_thread](#)

Definition at line 50 of file [ipc\\_gate](#).

References [L4::Cap\\_base::cap\(\)](#).

Here is the call graph for this function:



**11.79.2.2 `l4_mshtag_t L4::Ipc_gate::get_infos ( l4_umword_t * label, l4_utcb_t * utcb = l4_utcb() ) throw () [inline]`**

Get information on the IPC-gate.

#### See also

[l4\\_ipc\\_gate\\_get\\_infos](#)

Definition at line 59 of file [ipc\\_gate](#).

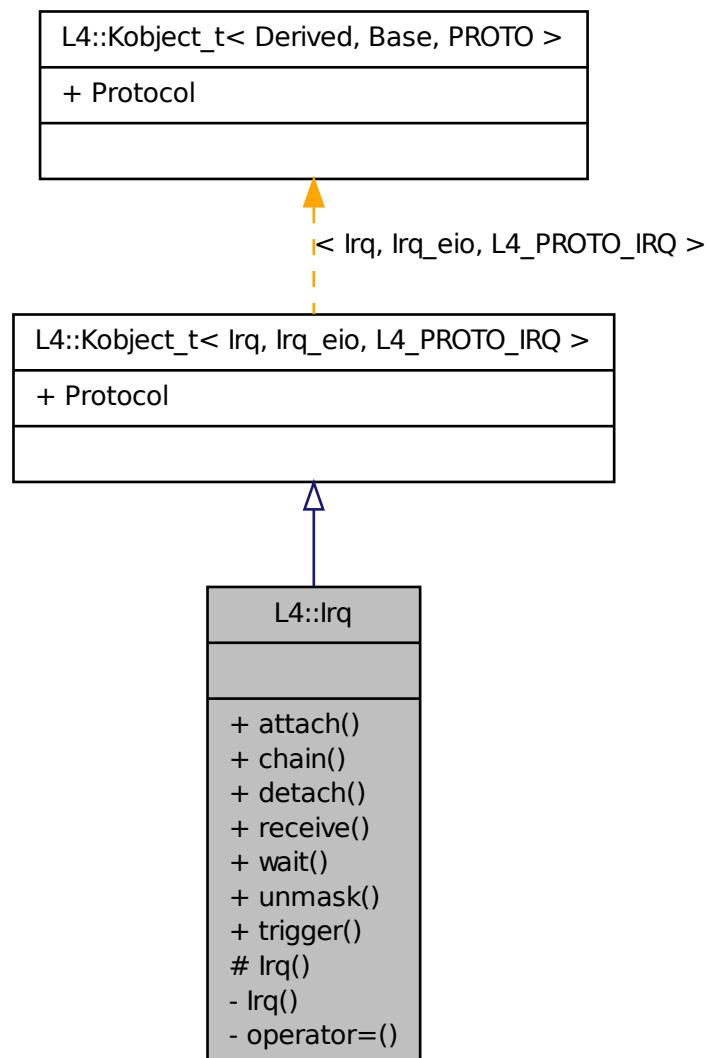
The documentation for this class was generated from the following file:

- l4/sys/ipc\_gate

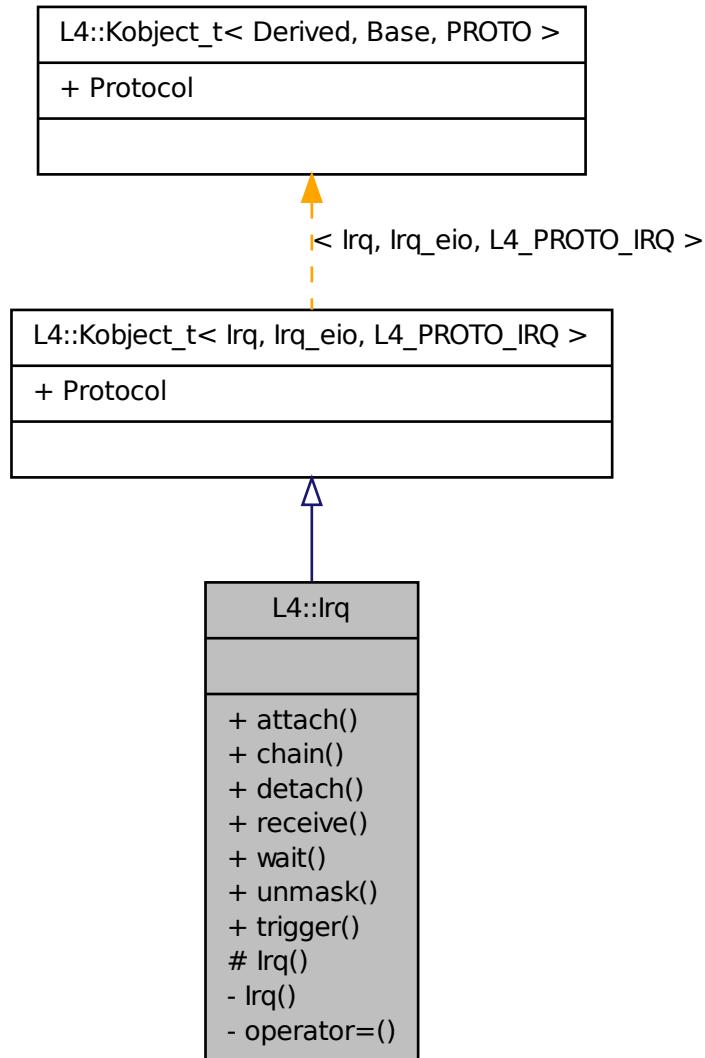
## 11.80 L4::Irq Class Reference

C++ version of an [L4](#) IRQ.

Inheritance diagram for L4::Irq:



Collaboration diagram for L4::Irq:



## Public Member Functions

- `l4_mshtag_t attach (l4_umword_t label, Cap< Thread > const &thread, l4_utcb_t *utcb=l4_utcb()) throw ()`
- `l4_mshtag_t chain (l4_umword_t label, Cap< Irq > const &slave, l4_utcb_t *utcb=l4_utcb()) throw ()`
- `l4_mshtag_t detach (l4_utcb_t *utcb=l4_utcb()) throw ()`
- `l4_mshtag_t receive (l4_timeout_t to=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb()) throw ()`

- `l4_mshtag_t wait (l4_umword_t *label, l4_timeout_t to=L4_IPC_NEVER, l4_utcb_t *utcb=l4_utcb()) throw ()`
- `l4_mshtag_t unmask (l4_utcb_t *utcb=l4_utcb()) throw ()`
- `l4_mshtag_t trigger (l4_utcb_t *utcb=l4_utcb()) throw ()`

### 11.80.1 Detailed Description

C++ version of an L4 IRQ. #include <l4/sys/irq>

#### See also

[IRQs](#) for an overview and C bindings.

#### Examples:

`examples/libs/l4re/c++/shared_ds/ds_clnt.cc.`

Definition at line 54 of file `irq`.

### 11.80.2 Member Function Documentation

#### 11.80.2.1 `l4_mshtag_t L4::Irq::attach ( l4_umword_t label, Cap< Thread > const & thread, l4_utcb_t * utcb = l4_utcb() ) throw () [inline]`

Attach to an interrupt source.

#### Parameters

*irq* IRQ to attach to.

*label* Identifier of the IRQ.

*flow\_type* Interrupt type, see `L4_irq_flow_type`

*thread* Thread to attach the interrupt to.

#### Returns

Syscall return tag

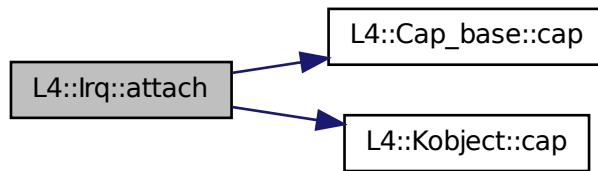
#### Note

*irq* is the implicit *this* pointer.

Definition at line 64 of file `irq`.

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



**11.80.2.2 l4\_mshtag\_t L4::Irq::chain ( l4\_umword\_t *label*, Cap< Irq > const & *slave*, l4\_utcb\_t \* *utcb* = *l4\_utcb()* ) throw () [inline]**

Chain an IRQ to another master IRQ source.

The chaining feature of IRQ objects allows to deal with shared IRQs. For chaining IRQs there must be a master IRQ object, bound to the real IRQ source. Note, the master IRQ must not have a thread attached to it. This function allows to add a limited number of slave IRQs to this master IRQ, with the semantics that each of the slave IRQs is triggered whenever the master IRQ is triggered. The master IRQ will be masked automatically when an IRQ is delivered and shall be unmasked when all attached slave IRQs are unmasked.

### Parameters

*irq* The master IRQ object.

*label* Identifier of the IRQ.

*flow\_type* Interrupt type, see L4\_irq\_flow\_type

*slave* The slave that shall be attached to the master.

### Returns

Syscall return tag

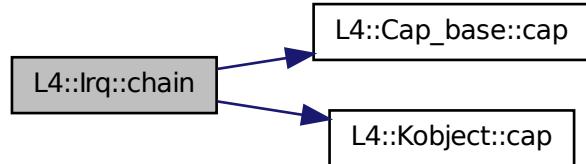
### Note

*irq* is the implicit *this* pointer.

Definition at line 72 of file [irq](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 11.80.2.3 l4\_mshtag\_t L4::Irq::detach ( l4\_utcb\_t \* utcb = [l4\\_utcb\(\)](#) ) throw () [inline]

Detach from an interrupt source.

#### Parameters

*irq* IRQ to detach from.

#### Returns

Syscall return tag

#### Note

*irq* is the implicit *this* pointer.

Definition at line 80 of file [irq](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 11.80.2.4 l4\_mshtag\_t L4::Irq::receive ( l4\_timeout\_t to = [L4\\_IPC\\_NEVER](#), l4\_utcb\_t \* utcb = [l4\\_utcb\(\)](#) ) throw () [inline]

Unmask and wait for specified IRQ.

### Parameters

*irq* IRQ to wait for.

*to* Timeout.

### Returns

Syscall return tag

### Note

*irq* is the implicit *this* pointer.

Definition at line 88 of file [irq](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



## 11.80.2.5 l4\_msgtag\_t L4::Irq::wait ( l4\_umword\_t \* label, l4\_timeout\_t to = L4\_IPC\_NEVER, l4\_utcb\_t \* utcb = l4\_utcb() ) throw () [inline]

Unmask IRQ and wait for any message.

### Parameters

*irq* IRQ to wait for.

*label* Receive label.

*to* Timeout.

### Returns

Syscall return tag

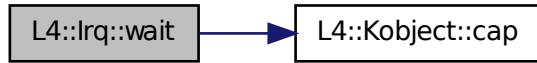
### Note

*irq* is the implicit *this* pointer.

Definition at line 96 of file [irq](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 11.80.2.6 l4\_mshtag\_t L4::Irq::unmask ( l4\_utcb\_t \* utcb = l4\_utcb() ) throw () [inline]

Unmask IRQ.

#### Parameters

*irq* IRQ to unmask.

#### Returns

Syscall return tag

#### Note

`l4_irq_wait` and `l4_irq_receive` are doing the unmask themselves.

#### Note

*irq* is the implicit *this* pointer.

Definition at line 104 of file `irq.h`.

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 11.80.2.7 l4\_mshtag\_t L4::Irq::trigger ( l4\_utcb\_t \* utcb = l4\_utcb() ) throw () [inline]

Trigger an IRQ.

### Parameters

*irq* IRQ to trigger.

### Precondition

*irq* must be a reference to an IRQ.

### Returns

Syscall return tag, note this is a send only operation, i.e. there is no return value except for failed sending.

### Note

*irq* is the implicit *this* pointer.

Definition at line 111 of file [irq](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



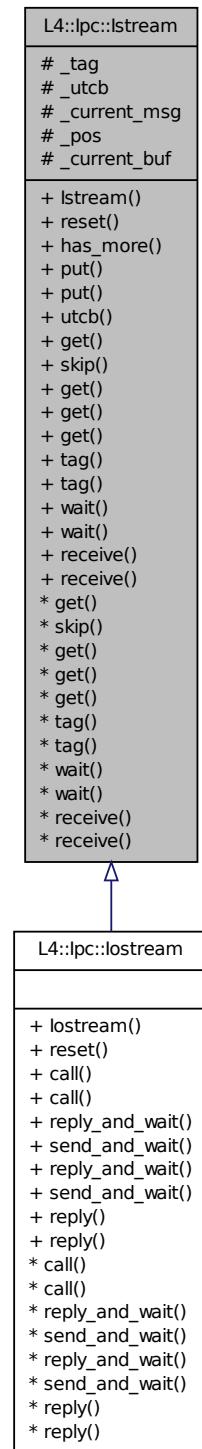
The documentation for this class was generated from the following file:

- [l4/sys/irq](#)

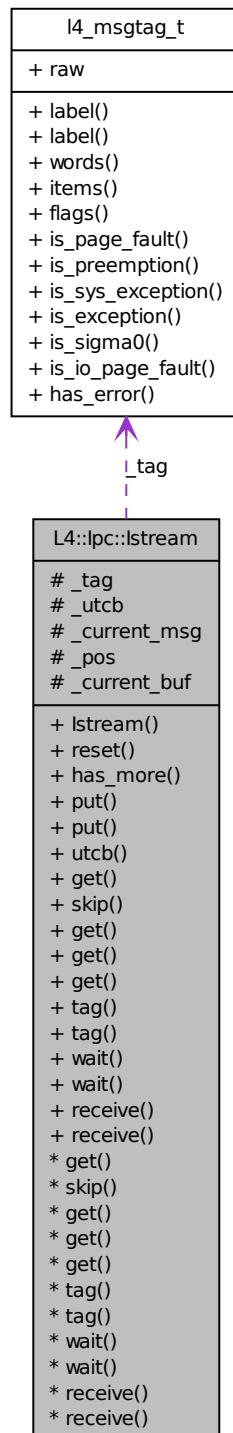
## 11.81 L4::Ipc::Istream Class Reference

Input stream for IPC unmarshalling.

Inheritance diagram for L4::Ipc::Istream:



Collaboration diagram for L4::Ipc::Istream:



## Public Member Functions

- **Istream (l4\_utcb\_t \*utcbl)**  
*Create an input stream for the given message buffer.*
- **void reset ()**  
*Reset the stream to empty, and ready for [receive\(\)](#)/wait().*
- template<typename T >  
**bool has\_more ()**  
*Check whether a value of type T can be obtained from the stream.*
- **l4\_utcb\_t \* utcb () const**  
*Return utcb pointer.*

### Get/Put Functions.

*These functions are basically used to implement the extraction operators (>>) and should not be called directly.*

*See [IPC stream operators](#).*

- template<typename T >  
**void get (T \*buf, unsigned long size)**  
*Copy out an array of type T with size elements.*
- template<typename T >  
**void skip (unsigned long size)**  
*Skip size elements of type T in the stream.*
- template<typename T >  
**void get (Msg\_ptr< T > const &buf, unsigned long size=1)**  
*Read one size elements of type T from the stream and return a pointer.*
- template<typename T >  
**void get (T &v)**  
*Extract a single element of type T from the stream.*
- **bool get (Ipc::Varg \*va)**
- **l4\_mshtag\_t tag () const**  
*Get the message tag of a received IPC.*
- **l4\_mshtag\_t & tag ()**  
*Get the message tag of a received IPC.*

### IPC operations.

- **l4\_mshtag\_t wait (l4\_umword\_t \*src)**  
*Wait for an incoming message from any sender.*
- **l4\_mshtag\_t wait (l4\_umword\_t \*src, l4\_timeout\_t timeout)**  
*Wait for an incoming message from any sender.*

- [l4\\_msntag\\_t receive \(l4\\_cap\\_idx\\_t src\)](#)  
*Wait for a message from the specified sender.*
- [l4\\_msntag\\_t receive \(l4\\_cap\\_idx\\_t src, l4\\_timeout\\_t timeout\)](#)

### 11.81.1 Detailed Description

Input stream for IPC unmarshalling. [Ipc::Istream](#) is part of the dynamic IPC marshalling infrastructure, as well as [Ipc::Ostream](#) and [Ipc::Iostream](#).

[Ipc::Istream](#) is an input stream supporting extraction of values from an IPC message buffer. A received IPC message can be unmarshalled using the usual extraction operator ( $>>$ ).

There exist some special wrapper classes to extract arrays (see [Ipc\\_buf\\_cp\\_in](#) and [Ipc\\_buf\\_in](#)) and indirect strings (see [Msg\\_in\\_buffer](#) and [Msg\\_io\\_buffer](#)).

Definition at line 572 of file [ipc\\_stream](#).

### 11.81.2 Constructor & Destructor Documentation

#### 11.81.2.1 L4::Ipc::Istream::Istream ( l4\_utcb\_t \* utcb ) [inline]

Create an input stream for the given message buffer.

The given message buffer is used for IPC operations [wait\(\)](#)/[receive\(\)](#) and received data can be extracted using the  $>>$  operator afterwards. In the case of indirect message parts a buffer of type [Msg\\_in\\_buffer](#) must be inserted into the stream before the IPC operation and contains received data afterwards.

#### Parameters

*msg* The message buffer to receive IPC messages.

Definition at line 586 of file [ipc\\_stream](#).

### 11.81.3 Member Function Documentation

#### 11.81.3.1 void L4::Ipc::Istream::reset ( ) [inline]

Reset the stream to empty, and ready for [receive\(\)](#)/[wait\(\)](#).

The stream is reset to the same state as on its creation.

Definition at line 596 of file [ipc\\_stream](#).

References [l4\\_msg\\_regs\\_t::mr](#).

#### 11.81.3.2 template<typename T> void L4::Ipc::Istream::get ( T \* buf, unsigned long size ) [inline]

Copy out an array of type *T* with *size* elements.

#### Parameters

*buf* Pointer to a buffer for *size* elements of type *T*.

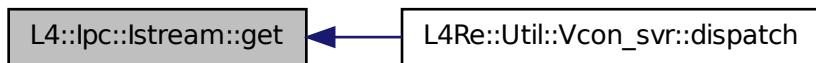
*size* number of elements of type T to copy out.

See [IPC stream operators](#).

Definition at line 631 of file [ipc\\_stream](#).

Referenced by [L4Re::Util::Vcon\\_svr< SVR >::dispatch\(\)](#).

Here is the caller graph for this function:



### 11.81.3.3 template<typename T> void L4::Ipc::Istream::skip ( unsigned long size ) [inline]

Skip size elements of type T in the stream.

#### Parameters

*size* number of elements to skip.

Definition at line 648 of file [ipc\\_stream](#).

### 11.81.3.4 template<typename T> void L4::Ipc::Istream::get ( Msg\_ptr< T > const & buf, unsigned long size = 1 ) [inline]

Read one size elements of type T from the stream and return a pointer.

In contrast to a normal get, this version does actually not copy the data but returns a pointer to the data.

#### Parameters

*buf* a [Msg\\_ptr](#) that is actually set to point to the element in the stream.

*size* number of elements to extract (default is 1).

See [IPC stream operators](#).

Definition at line 671 of file [ipc\\_stream](#).

### 11.81.3.5 template<typename T> void L4::Ipc::Istream::get ( T & v ) [inline]

Extract a single element of type T from the stream.

#### Parameters

*v* Output: the element.

See [IPC stream operators](#).

Definition at line 690 of file [ipc\\_stream](#).

#### 11.81.3.6 `l4_mshtag_t L4::Ipc::Istream::tag( ) const [inline]`

Get the message tag of a received IPC.

##### Returns

The [L4](#) message tag for the received IPC.

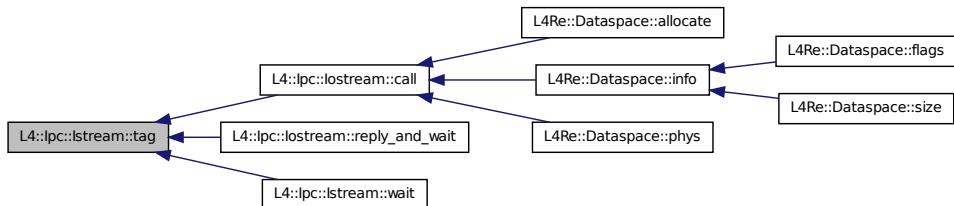
This is in particular useful for handling page faults or exceptions.

See [IPC stream operators](#).

Definition at line 726 of file [ipc\\_stream](#).

Referenced by [L4::Ipc::Iostream::call\(\)](#), [L4::Ipc::Iostream::reply\\_and\\_wait\(\)](#), and [wait\(\)](#).

Here is the caller graph for this function:



#### 11.81.3.7 `l4_mshtag_t& L4::Ipc::Istream::tag( ) [inline]`

Get the message tag of a received IPC.

##### Returns

A reference to the [L4](#) message tag for the received IPC.

This is in particular useful for handling page faults or exceptions.

See [IPC stream operators](#).

Definition at line 737 of file [ipc\\_stream](#).

#### 11.81.3.8 `l4_mshtag_t L4::Ipc::Istream::wait( l4_umword_t * src ) [inline]`

Wait for an incoming message from any sender.

##### Parameters

*src* contains the sender after a successful IPC operation.

## Returns

The IPC result dope ([l4\\_msgtag\\_t](#)).

This wait is actually known as 'open wait'.

Definition at line [766](#) of file [ipc\\_stream](#).

References [L4\\_IPC\\_NEVER](#), and [wait\(\)](#).

Referenced by [wait\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### **11.81.3.9 l4\_msgtag\_t L4::Ipc::Istream::wait ( l4\_umword\_t \* src, l4\_timeout\_t timeout ) [inline]**

Wait for an incoming message from any sender.

## Parameters

*src* contains the sender after a successful IPC operation.

*timeout* Timeout used for IPC.

## Returns

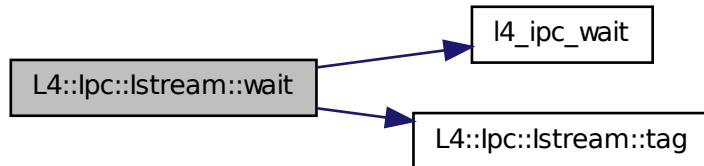
The IPC result dope ([l4\\_msgtag\\_t](#)).

This wait is actually known as 'open wait'.

Definition at line 1207 of file ipc\_stream.

References [l4\\_ipc\\_wait\(\)](#), and [tag\(\)](#).

Here is the call graph for this function:



### 11.81.3.10 l4\_mshtag\_t L4::Ipc::Istream::receive ( l4\_cap\_idx\_t src ) [inline]

Wait for a message from the specified sender.

#### Parameters

*src* The sender id to receive from.

#### Returns

The IPC result dope ([l4\\_mshtag\\_t](#)).

This is commonly known as 'closed wait'.

Definition at line 786 of file ipc\_stream.

References [L4\\_IPC\\_NEVER](#), and [receive\(\)](#).

Referenced by [receive\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



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

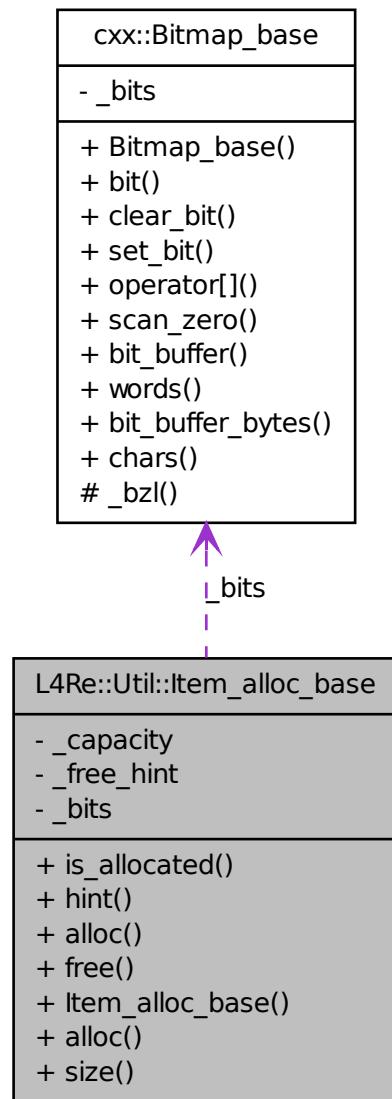
- l4/cxx/ipc\_stream

## 11.82 L4Re::Util::Item\_alloc\_base Class Reference

Item allocator.

Inherited by L4Re::Util::Item\_alloc< Bits >.

Collaboration diagram for L4Re::Util::Item\_alloc\_base:



### 11.82.1 Detailed Description

Item allocator.

Definition at line 38 of file [item\\_alloc](#).

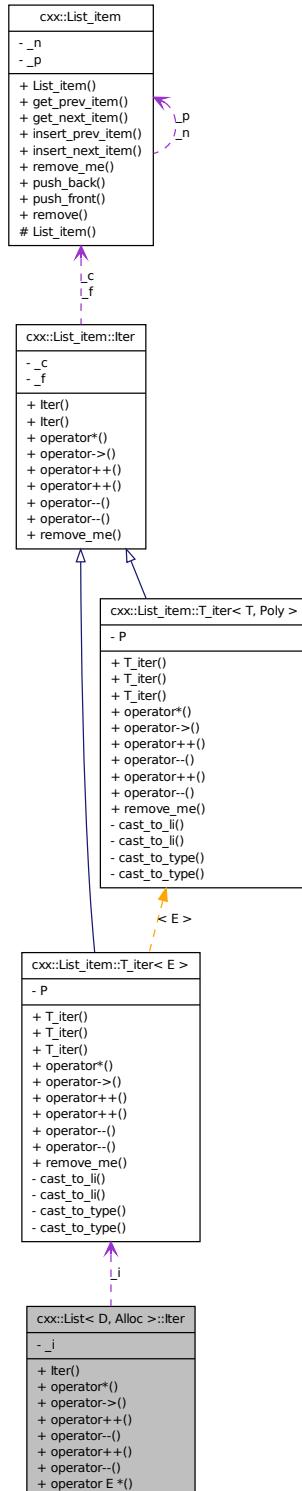
The documentation for this class was generated from the following file:

- [l4/re/util/item\\_alloc](#)

## 11.83 cxx::List< D, Alloc >::Iter Class Reference

Iterator.

Collaboration diagram for cxx::List< D, Alloc >::Iter:



## Public Member Functions

- [operator E \\* \(\) const throw \(\)](#)

*operator for testing validity (syntactiaclly equal to pointers)*

### 11.83.1 Detailed Description

**template<typename D, template< typename A > class Alloc = New\_allocator> class cxx::List< D, Alloc >::Iter**

Iterator. Forward and backward iteratable.

Definition at line [321](#) of file [list](#).

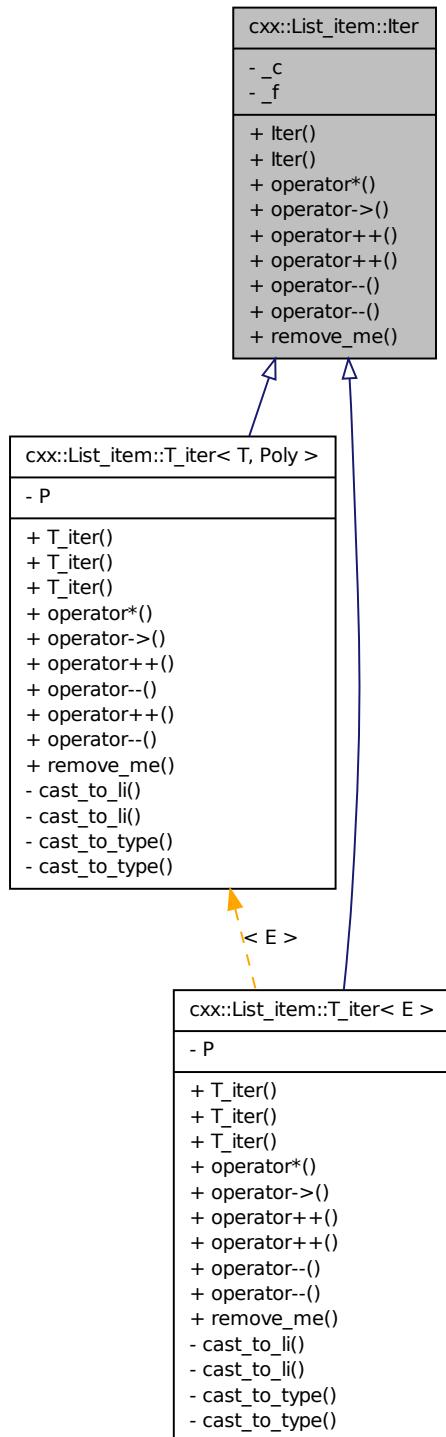
The documentation for this class was generated from the following file:

- [14/cxx/list](#)

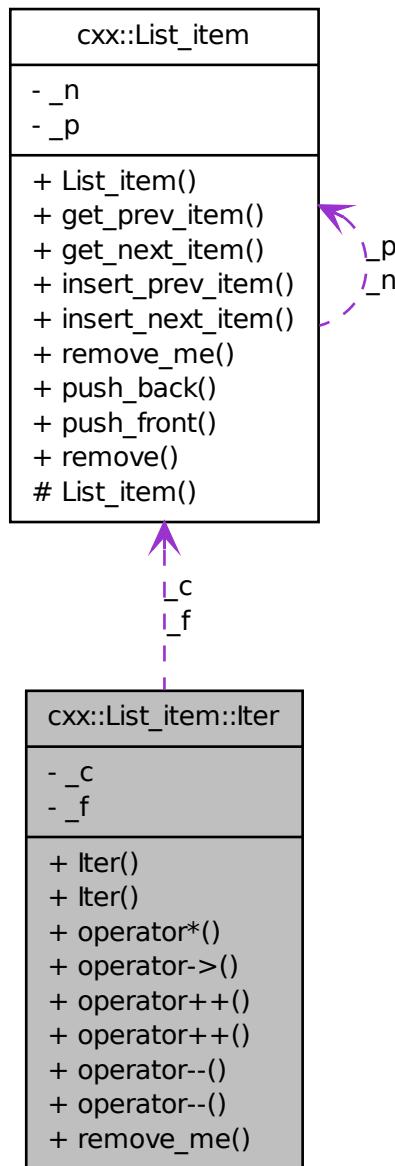
## 11.84 cxx::List\_item::Iter Class Reference

Iterator for a list of ListItem-s.

Inheritance diagram for cxx::List\_item::Iter:



Collaboration diagram for cxx::List\_item::Iter:



## Public Member Functions

- `List_item * remove_me () throw ()`

*Remove item pointed to by iterator, and return pointer to element.*

### 11.84.1 Detailed Description

Iterator for a list of ListItem-s. The Iterator iterates til it finds the first element again.

Definition at line 45 of file [list](#).

### 11.84.2 Member Function Documentation

#### 11.84.2.1 `List_item* cxx::List_item::Iter::remove_me( ) throw() [inline]`

Remove item pointed to by iterator, and return pointer to element.

Reimplemented in `cxx::List_item::T_iter< T, Poly >`, and `cxx::List_item::T_iter< E >`.

Definition at line 86 of file [list](#).

References `cxx::List_item::remove_me()`.

Referenced by `cxx::List_item::T_iter< T, Poly >::remove_me()`.

Here is the call graph for this function:



Here is the caller graph for this function:



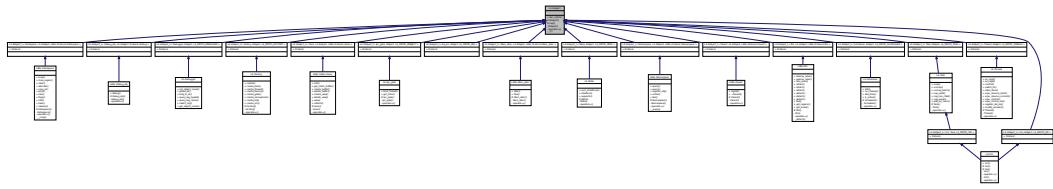
The documentation for this class was generated from the following file:

- l4/cxx/list

## 11.85 L4::Kobject Class Reference

Base class for all kinds of kernel objects, referred to by capabilities.

## Inheritance diagram for L4::Kobject:



## Public Member Functions

- `l4_msntag_t dec_refcnt (l4_mword_t diff, l4_utcb_t *utcb=l4_utcb())`  
*Decrement the in kernel reference counter for the object.*

## Protected Member Functions

- `l4_cap_idx_t cap () const throw ()`  
*Return capability selector.*

## **Friends**

- template<typename T >  
`Type_info const * kobject_typeid ()`  
Get the `L4::Type_info` for the `L4Re` interface given in T.

### **11.85.1 Detailed Description**

```
Base class for all kinds of kernel objects, referred to by capabilities. #include <4.1/sys/capability>
```

## Attention

Objects derived from [Kobject](#) must never add any data to those objects. Kobjects can act only as proxy object for encapsulating object invocations.

Definition at line 452 of file [capability](#).

## 11.85.2 Member Function Documentation

11.85.2.1 **l4\_cap\_idx\_t L4::Kobject::cap( ) const throw()** [inline, protected]

Return capability selector.

## Returns

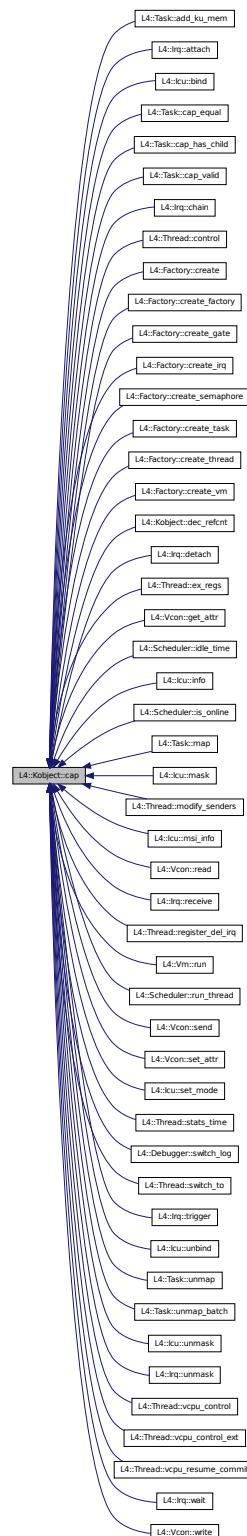
## Capability selector.

This method is for derived classes to gain access to the actual capability selector.

Definition at line 478 of file [capability](#).

Referenced by [L4::Task::add\\_ku\\_mem\(\)](#), [L4::Irq::attach\(\)](#), [L4::Icu::bind\(\)](#), [L4::Task::cap\\_equal\(\)](#), [L4::Task::cap\\_has\\_child\(\)](#), [L4::Task::cap\\_valid\(\)](#), [L4::Irq::chain\(\)](#), [L4::Thread::control\(\)](#), [L4::Factory::create\(\)](#), [L4::Factory::create\\_factory\(\)](#), [L4::Factory::create\\_gate\(\)](#), [L4::Factory::create\\_irq\(\)](#), [L4::Factory::create\\_semaphore\(\)](#), [L4::Factory::create\\_task\(\)](#), [L4::Factory::create\\_thread\(\)](#), [L4::Factory::create\\_vm\(\)](#), [dec\\_refcnt\(\)](#), [L4::Irq::detach\(\)](#), [L4::Thread::ex\\_regs\(\)](#), [L4::Vcon::get\\_attr\(\)](#), [L4::Scheduler::idle\\_time\(\)](#), [L4::Icu::info\(\)](#), [L4::Scheduler::is\\_online\(\)](#), [L4::Task::map\(\)](#), [L4::Icu::mask\(\)](#), [L4::Thread::modify\\_senders\(\)](#), [L4::Icu::msi\\_info\(\)](#), [L4::Vcon::read\(\)](#), [L4::Irq::receive\(\)](#), [L4::Thread::register\\_del\\_irq\(\)](#), [L4::Vm::run\(\)](#), [L4::Scheduler::run\\_thread\(\)](#), [L4::Vcon::send\(\)](#), [L4::Vcon::set\\_attr\(\)](#), [L4::Icu::set\\_mode\(\)](#), [L4::Thread::stats\\_time\(\)](#), [L4::Debugger::switch\\_log\(\)](#), [L4::Thread::switch\\_to\(\)](#), [L4::Irq::trigger\(\)](#), [L4::Icu::unbind\(\)](#), [L4::Task::unmap\(\)](#), [L4::Task::unmap\\_batch\(\)](#), [L4::Icu::unmask\(\)](#), [L4::Irq::unmask\(\)](#), [L4::Thread::vcpu\\_control\(\)](#), [L4::Thread::vcpu\\_control\\_ext\(\)](#), [L4::Thread::vcpu\\_resume\\_commit\(\)](#), [L4::Irq::wait\(\)](#), and [L4::Vcon::write\(\)](#).

Here is the caller graph for this function:



### 11.85.2.2 `l4_msntag_t L4::Kobject::dec_refcnt ( l4_mword_t diff, l4_utcb_t * utcb = 14_utcb() ) [inline]`

Decrement the in kernel reference counter for the object.

#### Parameters

`diff` is the delta that shall be subtracted from the reference count.

`utcb` is the utcb to use for the invocation.

This function is intended for servers to be able to remove the servers own capability from the counted references. This leads to the semantics that the kernel will delete the object even if the capability of the server is valid. The server can detect the deletion by polling its capabilities or by using the IPC-gate deletion IRQs. And to cleanup if the clients dropped the last reference (capability) to the object.

Definition at line 503 of file [capability](#).

References [cap\(\)](#).

Here is the call graph for this function:



### 11.85.3 Friends And Related Function Documentation

#### 11.85.3.1 `template<typename T > Type_info const* kobject_typeid ( ) [friend]`

Get the [L4::Type\\_info](#) for the [L4Re](#) interface given in *T*.

#### Parameters

*T* The type ([L4Re](#) interface) for which the information shall be returned.

#### Returns

A pointer to the [L4::Type\\_info](#) structure for *T*.

Reimplemented in [L4::Kobject\\_t< Ipc\\_gate, Kobject, L4\\_PROTO\\_KOBJECT >](#), [L4::Kobject\\_t< Namespace, L4::Kobject, L4Re::Protocol::Namespace >](#), [L4::Kobject\\_t< Parent, L4::Kobject, L4Re::Protocol::Parent >](#), [L4::Kobject\\_t< Irq\\_eio, Kobject, L4\\_PROTO\\_IRQ >](#), [L4::Kobject\\_t< Factory, Kobject, L4\\_PROTO\\_FACTORY >](#), [L4::Kobject\\_t< Task, Kobject, L4\\_PROTO\\_TASK >](#), [L4::Kobject\\_t< Debug\\_obj, L4::Kobject, Protocol::Debug >](#), [L4::Kobject\\_t< Vm, Kobject, L4\\_PROTO\\_VM >](#), [L4::Kobject\\_t< Event, L4::Icu, L4Re::Protocol::Event >](#), [L4::Kobject\\_t< Scheduler, Kobject, L4\\_PROTO\\_SCHEDULER >](#), [L4::Kobject\\_t< Rm, L4::Kobject, L4Re::Protocol::Rm >](#), [L4::Kobject\\_t< Log, L4::Vcon, L4\\_PROTO\\_LOG >](#), [L4::Kobject\\_t< Vcon, Icu, L4\\_PROTO\\_LOG >](#), [L4::Kobject\\_t< Meta, Kobject, L4\\_PROTO\\_META >](#), [L4::Kobject\\_t< Dataspace, L4::Kobject,](#)

`L4Re::Protocol::Dataspace >, L4::Kobject_t< Vm, Task, L4_PROTO_VM >, L4::Kobject_t< Debugger, Kobject, L4_PROTO_DEBUGGER >, L4::Kobject_t< Mem_alloc, L4::Kobject, L4Re::Protocol::Mem_alloc >, L4::Kobject_t< Thread, Kobject, L4_PROTO_THREAD >, L4::Kobject_t< Irq, Irq_eio, L4_PROTO_IRQ >, L4::Kobject_t< Icu, Irq_eio, L4_PROTO_IRQ >, and L4::Kobject_t< Goos, L4::Kobject, L4Re::Protocol::Goos >.`

Definition at line 87 of file `__typeinfo.h`.

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

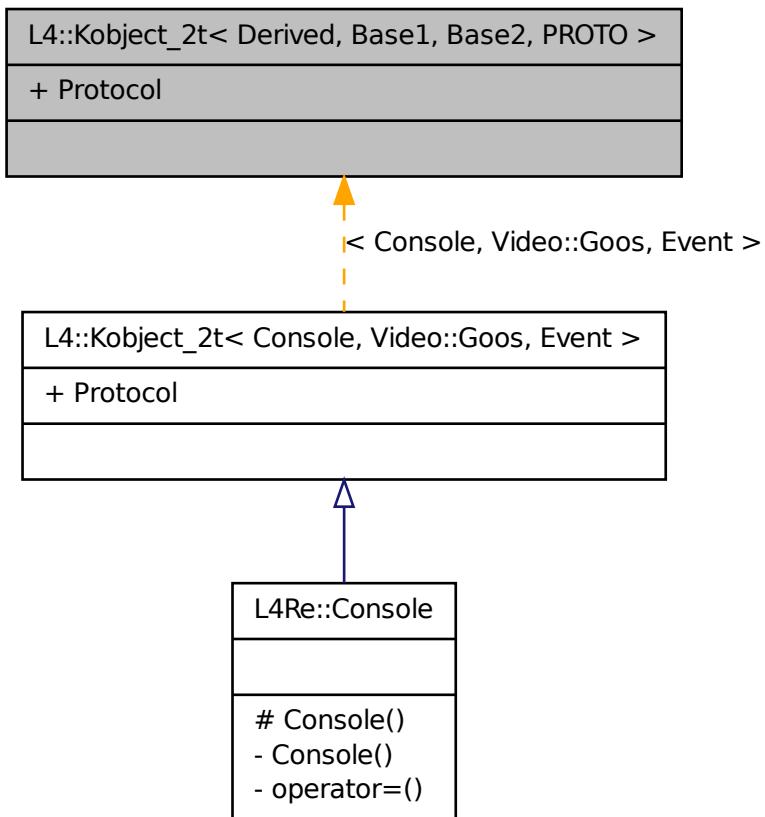
- `l4/sys/capability`

## 11.86 `L4::Kobject_2t< Derived, Base1, Base2, PROTO >` Class Template Reference

Helper class to create an `L4Re` interface class that is derived from two base classes.

```
#include <__typeinfo.h>
```

Inheritance diagram for `L4::Kobject_2t< Derived, Base1, Base2, PROTO >`:



## Friends

- template<typename T >  
`Type_info const * kobject_typeid ()`

*Get the L4::Type\_info for the L4Re interface given in T.*

### 11.86.1 Detailed Description

```
template<typename Derived, typename Base1, typename Base2, long PROTO = 0> class
L4::Kobject_2t< Derived, Base1, Base2, PROTO >
```

Helper class to create an L4Re interface class that is derived from two base classes.

#### Parameters

- Derived** is the name of the new interface.
- Base1** is the name of the interfaces first base class.
- Base2** is the name of the interfaces second base class.
- PROTO** may be set to the statically assigned protocol number used to communicate with this interface.

The typical usage pattern is shown in the following code snippet. The semantics of this example is an interface My\_iface that is derived from L4::Icu and L4Re::Dataspace.

```
class My_iface : public L4::Kobject_2t<My_iface, L4::Icu, L4Re::Dataspace>
{
    ...
};
```

Definition at line 175 of file [\\_\\_typeinfo.h](#).

### 11.86.2 Friends And Related Function Documentation

#### 11.86.2.1 template<typename Derived, typename Base1, typename Base2, long PROTO = 0> template<typename T > Type\_info const\* kobject\_typeid ( ) [friend]

Get the L4::Type\_info for the L4Re interface given in T.

#### Parameters

- T** The type (L4Re interface) for which the information shall be returned.

#### Returns

A pointer to the L4::Type\_info structure for T.

Definition at line 87 of file [\\_\\_typeinfo.h](#).

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

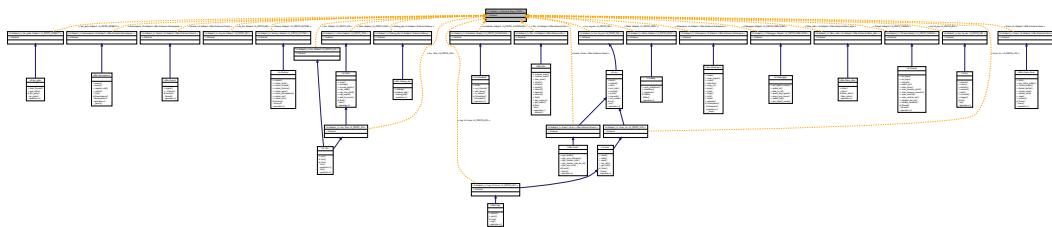
- l4/sys/\_\_typeinfo.h

## 11.87 L4::Kobject\_t< Derived, Base, PROTO > Class Template Reference

Helper class to create an L4Re interface class that is derived from a single base class.

```
#include <__typeinfo.h>
```

Inheritance diagram for L4::Kobject\_t< Derived, Base, PROTO >:



### Friends

- template<typename T >  
`Type_info const * kobject_typeid ()`

*Get the L4::Type\_info for the L4Re interface given in T.*

#### 11.87.1 Detailed Description

**template<typename Derived, typename Base, long PROTO = 0> class L4::Kobject\_t< Derived, Base, PROTO >**

Helper class to create an L4Re interface class that is derived from a single base class.

#### Parameters

**Derived** is the name of the new interface.

**Base** is the name of the interfaces single base class.

**PROTO** may be set to the statically assigned protocol number used to communicate with this interface.

The typical usage pattern is shown in the following code snippet. The semantics of this example is an interface My\_iface that is derived from L4::Kobject.

```
class My_iface : public L4::Kobject_t<My_iface, L4::Kobject>
{
    ...
};
```

Definition at line 137 of file [\\_\\_typeinfo.h](#).

## 11.87.2 Friends And Related Function Documentation

**11.87.2.1 template<typename Derived, typename Base, long PROTO = 0> template<typename T > Type\_info const\* kobject\_typeid ( ) [friend]**

Get the [L4::Type\\_info](#) for the [L4Re](#) interface given in *T*.

### Parameters

*T* The type ([L4Re](#) interface) for which the information shall be returned.

### Returns

A pointer to the [L4::Type\\_info](#) structure for *T*.

Reimplemented in [L4::Kobject\\_t< Event, L4::Icu, L4Re::Protocol::Event >](#), [L4::Kobject\\_t< Log, L4::Vcon, L4\\_PROTO\\_LOG >](#), [L4::Kobject\\_t< Vcon, Icu, L4\\_PROTO\\_LOG >](#), [L4::Kobject\\_t< Vm, Task, L4\\_PROTO\\_VM >](#), [L4::Kobject\\_t< Irq, Irq\\_eio, L4\\_PROTO\\_IRQ >](#), and [L4::Kobject\\_t< Icu, Irq\\_eio, L4\\_PROTO\\_IRQ >](#).

Definition at line 87 of file [\\_\\_typeinfo.h](#).

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

- [14/sys/\\_\\_typeinfo.h](#)

## 11.88 l4\_buf\_regs\_t Struct Reference

Encapsulation of the buffer-registers block in the UTCB.

```
#include <14/sys/utcbl.h>
```

### Data Fields

- [l4\\_umword\\_t bdr](#)  
*Buffer descriptor.*
- [l4\\_umword\\_t br \[L4\\_UTCB\\_GENERIC\\_BUFFERS\\_SIZE\]](#)  
*Buffer registers.*

## 11.88.1 Detailed Description

Encapsulation of the buffer-registers block in the UTCB.

Definition at line 95 of file [utcbl.h](#).

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

- [14/sys/utcbl.h](#)

## 11.89 l4\_exc\_regs\_t Struct Reference

UTCB structure for exceptions.

```
#include <utcb.h>
```

### Data Fields

- `l4_umword_t pfa`  
*page fault address*
- `l4_umword_t err`  
*error code*
- `l4_umword_t r [13]`  
*registers*
- `l4_umword_t sp`  
*stack pointer*
- `l4_umword_t ulr`  
*ulr*
- `l4_umword_t _dummy1`  
*dummy*
- `l4_umword_t pc`  
*pc*
- `l4_umword_t cpsr`  
*cpsr*
- `l4_umword_t r15`  
*r15*
- `l4_umword_t r14`  
*r14*
- `l4_umword_t r13`  
*r13*
- `l4_umword_t r12`  
*r12*
- `l4_umword_t r11`  
*r11*
- `l4_umword_t r10`  
*r10*

- `l4_umword_t r9`  
*r9*
- `l4_umword_t r8`  
*r8*
- `l4_umword_t rdi`  
*rdi*
- `l4_umword_t rsi`  
*rsi*
- `l4_umword_t rbp`  
*rbp*
- `l4_umword_t rbx`  
*rbx*
- `l4_umword_t rdx`  
*rdx*
- `l4_umword_t rcx`  
*rcx*
- `l4_umword_t rax`  
*rax*
- `l4_umword_t trapno`  
*trap number*
- `l4_umword_t ip`  
*instruction pointer*
- `l4_umword_t dummy1`  
*dummy*
- `l4_umword_t flags`  
*rflags*
- `l4_umword_t ss`  
*stack segment register*
- `l4_umword_t gs`  
*gs register*
- `l4_umword_t fs`  
*fs register*
- `l4_umword_t edi`  
*edi register*

- [l4\\_umword\\_t esi](#)  
*esi register*
- [l4\\_umword\\_t ebp](#)  
*ebp register*
- [l4\\_umword\\_t ebx](#)  
*ebx register*
- [l4\\_umword\\_t edx](#)  
*edx register*
- [l4\\_umword\\_t ecx](#)  
*ecx register*
- [l4\\_umword\\_t eax](#)  
*eax register*

### 11.89.1 Detailed Description

UTCB structure for exceptions.

**Examples:**

[examples/sys/aliens/main.c](#), [examples/sys/singlestep/main.c](#), and [examples/sys/start-with-exc/main.c](#).

Definition at line 58 of file [utcbl.h](#).

### 11.89.2 Field Documentation

#### 11.89.2.1 l4\_umword\_t l4\_exc\_regs\_t::flags

rflags

eflags

Definition at line 80 of file [utcbl.h](#).

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

- [arm/l4/sys/utcbl.h](#)
- [amd64/l4/sys/utcbl.h](#)
- [x86/l4/sys/utcbl.h](#)

## 11.90 l4\_fpage\_t Union Reference

[L4](#) flexpage type.

```
#include <__l4_fpage.h>
```

## Data Fields

- `l4_umword_t fpage`

*Raw value.*

- `l4_umword_t raw`

*Raw value.*

### 11.90.1 Detailed Description

[L4](#) flexpage type.

Definition at line [78](#) of file [\\_\\_l4\\_fpage.h](#).

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

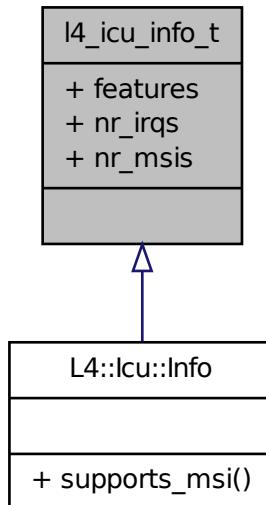
- `l4/sys/__l4_fpage.h`

## 11.91 l4\_icu\_info\_t Struct Reference

Info structure for an ICU.

```
#include <icu.h>
```

Inheritance diagram for l4\_icu\_info\_t:



## Data Fields

- unsigned [features](#)  
*Feature flags.*
- unsigned [nr\\_irqs](#)  
*The number of IRQ lines supported by the ICU.*
- unsigned [nr\\_msis](#)  
*The number of MSI vectors supported by the ICU.*

### 11.91.1 Detailed Description

Info structure for an ICU. This structure contains information about the features of an ICU.

#### See also

[l4\\_icu\\_info\(\)](#).

Definition at line 128 of file [icu.h](#).

### 11.91.2 Field Documentation

#### 11.91.2.1 unsigned l4\_icu\_info\_t::features

Feature flags.

If [L4\\_ICU\\_FLAG\\_MSI](#) is set the ICU supports MSIs.

Definition at line 135 of file [icu.h](#).

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

- [l4/sys/icu.h](#)

## 11.92 l4\_kernel\_info\_mem\_desc\_t Struct Reference

Memory descriptor data structure.

```
#include <memdesc.h>
```

### 11.92.1 Detailed Description

Memory descriptor data structure.

#### Note

This data type is opaque, and must be accessed by the accessor functions defined in this module.

Definition at line 64 of file [memdesc.h](#).

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

- l4/sys/memdesc.h

## 11.93 l4\_kernel\_info\_t Struct Reference

[L4 Kernel Interface Page.](#)

```
#include <__kip-32bit.h>
```

### Data Fields

- **l4\_uint32\_t magic**  
*Kernel Info Page identifier ("L4tK").*
- **l4\_uint32\_t version**  
*Kernel version.*
- **l4\_uint8\_t offset\_version\_strings**  
*offset to version string*
- **l4\_uint8\_t fill0 [3]**  
*reserved*
- **l4\_uint8\_t kip\_sys\_calls**  
*pointer to system calls*
- **l4\_uint8\_t fill1 [3]**  
*reserved*
- **l4\_umword\_t init\_default\_kdebug**  
*Kdebug init function.*
- **l4\_umword\_t default\_kdebug\_exception**  
*Kdebug exception handler.*
- **l4\_umword\_t scheduler\_granularity**  
*for rounding time slices*
- **l4\_umword\_t default\_kdebug\_end**  
*default\_kdebug\_end*
- **l4\_umword\_t sigma0\_esp**  
*Sigma0 start stack pointer.*
- **l4\_umword\_t sigma0\_eip**  
*Sigma0 instruction pointer.*
- **l4\_umword\_t \_res01 [2]**  
*reserved*

- `l4_umword_t sigma1_esp`  
*Sigma1 start stack pointer.*
- `l4_umword_t sigma1_eip`  
*Sigma1 instruction pointer.*
- `l4_umword_t _res02 [2]`  
*reserved*
- `l4_umword_t root_esp`  
*Root task stack pointer.*
- `l4_umword_t root_eip`  
*Root task instruction pointer.*
- `l4_umword_t _res03 [2]`  
*reserved*
- `l4_umword_t l4_config`  
*L4 kernel configuration.*
- `l4_umword_t mem_info`  
*memory information*
- `l4_umword_t kdebug_config`  
*Kernel debugger config.*
- `l4_umword_t kdebug_permission`  
*Kernel debugger permissions.*
- `l4_umword_t total_ram`  
*Size of RAM in bytes.*
- `l4_umword_t processor_info`  
*CPU info.*
- `l4_umword_t _res04 [14]`  
*reserved*
- `volatile l4_cpu_time_t clock`  
*L4 system clock (fs).*
- `l4_umword_t _res05 [2]`  
*reserved*
- `l4_umword_t frequency_cpu`  
*CPU frequency in kHz.*
- `l4_umword_t frequency_bus`  
*Bus frequency.*

- `l4_umword_t _res06` [10]  
*reserved*
- `l4_umword_t user_ptr`  
*user\_ptr*
- `l4_umword_t vhw_offset`  
*offset to vhw structure*
- `l4_umword_t _res07` [2]  
*reserved*
- `l4_uint64_t magic`  
*Kernel Info Page identifier ("L4tK").*
- `l4_uint64_t version`  
*Kernel version.*
- `l4_uint8_t fill2` [7]  
*reserved*
- `l4_uint8_t fill3` [7]  
*reserved*

### 11.93.1 Detailed Description

[L4](#) Kernel Interface Page.

**Examples:**

[examples/sys/ux-vhw/main.c](#).

Definition at line 38 of file [\\_kip-32bit.h](#).

### 11.93.2 Field Documentation

#### 11.93.2.1 `l4_umword_t l4_kernel_info_t::l4_config`

[L4](#) kernel configuration.

Values:

- bits 0-7: set the number of page table entries to allocate
- bits 8-15: set the number of mapping nodes.

Definition at line 82 of file [\\_kip-32bit.h](#).

### 11.93.2.2 l4\_umword\_t l4\_kernel\_info\_t::kdebug\_config

Kernel debugger config.

Values:

- bits 0-7: set the number of pages to allocate for the trace buffer
- bit 8: if set to 1, the kernel enters kdebug before starting the root task
- bits 16-19: set the port speed to use with serial line (1..115.2KBd, 2..57.6KBd, 3..38.4KBd, 6..19.2KBd, 12..9.6KBD)
- bits 20-31: set the I/O port to use with serial line, 0 indicates that no serial output should be used

Definition at line 92 of file [\\_kip-32bit.h](#).

### 11.93.2.3 l4\_umword\_t l4\_kernel\_info\_t::kdebug\_permission

Kernel debugger permissions.

Values:

- bits 0-7: if 0 all tasks can enter the kernel debugger, otherwise only tasks with a number lower than the set value can enter kdebug, other tasks will be shut down.
- bit 8: if set, kdebug may display mappings
- bit 9: if set, kdebug may display user registers
- bit 10: if set, kdebug may display user memory
- bit 11: if set, kdebug may modify memory, registers, mappings and tcbs
- bit 12: if set, kdebug may read/write I/O ports
- bit 13: if set, kdebug may protocol page faults and IPC

Definition at line 113 of file [\\_kip-32bit.h](#).

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

- l4/sys/\_kip-32bit.h
- l4/sys/\_kip-64bit.h

## 11.94 l4\_msg\_regs\_t Struct Reference

Encapsulation of the message-register block in the UTCB.

```
#include <l4/sys/utcbl.h>
```

### Data Fields

- [l4\\_umword\\_t mr](#) [L4\_UTCB\_GENERIC\_DATA\_SIZE]  
*Message registers.*

### 11.94.1 Detailed Description

Encapsulation of the message-register block in the UTCB.

#### Examples:

[examples/sys/utcb-ipc/main.c](#)

Definition at line 80 of file [utcb.h](#).

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

- [l4/sys/utcb.h](#)

## 11.95 l4\_mshtag\_t Struct Reference

Message tag data structure.

```
#include <types.h>
```

### Public Member Functions

- long [label](#) () const throw ()  
*Get the protocol value.*
- void [label](#) (long v) throw ()  
*Set the protocol value.*
- unsigned [words](#) () const throw ()  
*Get the number of untyped words.*
- unsigned [items](#) () const throw ()  
*Get the number of typed items.*
- unsigned [flags](#) () const throw ()  
*Get the flags value.*
- bool [is\\_page\\_fault](#) () const throw ()  
*Test if protocol indicates page-fault protocol.*
- bool [is\\_preemption](#) () const throw ()  
*Test if protocol indicates preemption protocol.*
- bool [is\\_sys\\_exception](#) () const throw ()  
*Test if protocol indicates system-exception protocol.*
- bool [is\\_exception](#) () const throw ()  
*Test if protocol indicates exception protocol.*
- bool [is\\_sigma0](#) () const throw ()

*Test if protocol indicates sigma0 protocol.*

- bool `is_io_page_fault () const throw ()`

*Test if protocol indicates IO-page-fault protocol.*

- unsigned `has_error () const throw ()`

*Test if flags indicate an error.*

## Data Fields

- `l4_mword_t raw`

*raw value*

### 11.95.1 Detailed Description

Message tag data structure. #include <l4/sys/types.h>

Describes the details of an IPC operation, in particular which parts of the UTCB have to be transmitted, and also flags to enable real-time and FPU extensions.

The message tag also contains a user-defined label that could be used to specify a protocol ID. Some negative values are reserved for kernel protocols such as page faults and exceptions.

The type must be treated completely opaque.

#### Examples:

`examples/clntsrv/client.cc`, `examples/clntsrv/server.cc`, `examples/libs/l4re/c++/shared_ds/ds_srv.cc`, `examples/libs/l4re/streammap/client.cc`, `examples/libs/l4re/streammap/server.cc`, `examples/sys/aliens/main.c`, `examples/sys/ipc/ipc_example.c`, `examples/sys/isr/main.c`, `examples/sys/singlestep/main.c`, `examples/sys/start-with-exc/main.c`, and `examples/sys/utcb-ipc/main.c`.

Definition at line 158 of file `types.h`.

### 11.95.2 Member Function Documentation

#### 11.95.2.1 `unsigned l4_mshtag_t::flags ( ) const throw () [inline]`

Get the flags value.

The flags are a combination of the flags defined by `l4_mshtag_flags`.

Definition at line 176 of file `types.h`.

References `raw`.

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

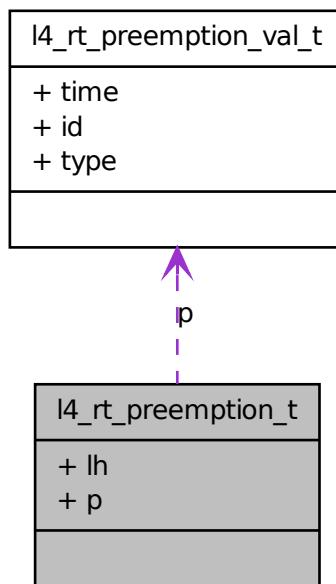
- `l4/sys/types.h`

## 11.96 l4\_rt\_preemption\_t Union Reference

Struct.

```
#include <rt_sched-proto.h>
```

Collaboration diagram for l4\_rt\_preemption\_t:



### Data Fields

- l4\_low\_high\_t **lh**  
*lh value*
- l4\_rt\_preemption\_val\_t **p**  
*p value*

#### 11.96.1 Detailed Description

Struct.

Definition at line 56 of file [rt\\_sched-proto.h](#).

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

- x86/l4/sys/rt\_sched-proto.h

## 11.97 l4\_rt\_preemption\_val32\_t Struct Reference

Struct.

```
#include <rt_sched-proto.h>
```

### Data Fields

- `l4_uint32_t time_high:24`

*time\_high*

- `l4_uint32_t id:7`

*id*

- `l4_uint32_t type:1`

*type*

### 11.97.1 Detailed Description

Struct.

Definition at line 50 of file `rt_sched-proto.h`.

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

- x86/l4/sys/rt\_sched-proto.h

## 11.98 l4\_rt\_preemption\_val\_t Struct Reference

Struct.

```
#include <rt_sched-proto.h>
```

### Data Fields

- `l4_uint64_t time:56`

*time*

- `l4_uint64_t id:7`

*id*

- `l4_uint64_t type:1`

*id*

### 11.98.1 Detailed Description

Struct.

Definition at line 44 of file [rt\\_sched Proto.h](#).

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

- [x86/l4/sys/rt\\_sched Proto.h](#)

## 11.99 l4\_sched\_cpu\_set\_t Struct Reference

CPU sets.

```
#include <scheduler.h>
```

### Data Fields

- [l4\\_umword\\_t offset](#)  
*First CPU of interest (must be aligned to  $2^{\wedge}$  granularity).*
- [l4\\_umword\\_t map](#)  
*Bitmap of online CPUs.*
- [unsigned char granularity](#)  
*One bit in map represents  $2^{\wedge}$  granularity CPUs.*

### 11.99.1 Detailed Description

CPU sets.

#### Examples:

[examples/sys/migrate/thread\\_migrate.cc](#).

Definition at line 40 of file [scheduler.h](#).

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

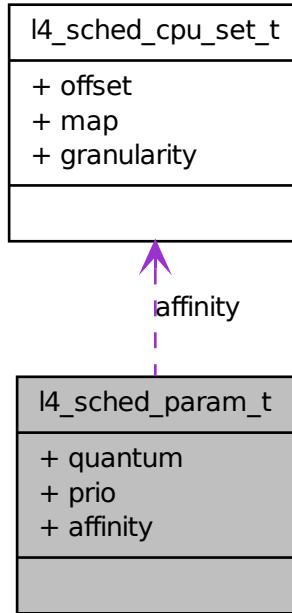
- [l4/sys/scheduler.h](#)

## 11.100 l4\_sched\_param\_t Struct Reference

Scheduler parameter set.

```
#include <scheduler.h>
```

Collaboration diagram for l4\_sched\_param\_t:



## Data Fields

- [l4\\_cpu\\_time\\_t quantum](#)  
*Timeslice in micro seconds.*
- [unsigned prio](#)  
*Priority for scheduling.*
- [l4\\_sched\\_cpu\\_set\\_t affinity](#)  
*CPU affinity.*

### 11.100.1 Detailed Description

Scheduler parameter set.

#### Examples:

[examples/sys/migrate/thread\\_migrate.cc](#).

Definition at line 101 of file [scheduler.h](#).

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

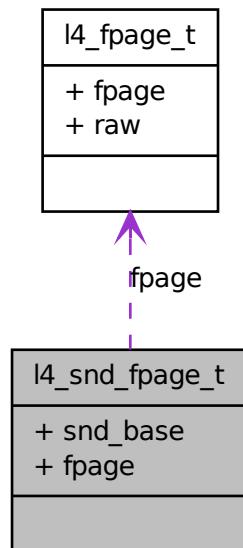
- l4/sys/scheduler.h

## 11.101 l4\_snd\_fpage\_t Struct Reference

Send-flex-page types.

```
#include <__l4_fpage.h>
```

Collaboration diagram for l4\_snd\_fpage\_t:



### Data Fields

- `l4_umword_t snd_base`  
*Offset in receive window (send base).*
- `l4_fpage_t fpage`  
*Source flex-page descriptor.*

#### 11.101.1 Detailed Description

Send-flex-page types.

Definition at line 95 of file [\\_\\_l4\\_fpage.h](#).

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

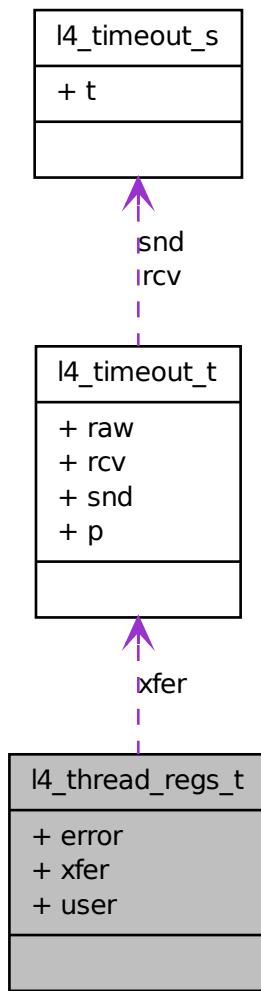
- l4/sys/\_l4\_fpage.h

## 11.102 l4\_thread\_regs\_t Struct Reference

Encapsulation of the thread-control-register block of the UTCB.

```
#include <l4/sys/utcbl.h>
```

Collaboration diagram for l4\_thread\_regs\_t:



### Data Fields

- `l4_umword_t` `error`

*System call error codes.*

- [l4\\_timeout\\_t xfer](#)  
*Message transfer timeout.*
- [l4\\_umword\\_t user](#) [3]  
*User values (ignored and preserved by the kernel).*

### 11.102.1 Detailed Description

Encapsulation of the thread-control-register block of the UTCB.

Definition at line 113 of file [utcb.h](#).

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

- [l4/sys/utcb.h](#)

## 11.103 l4\_timeout\_s Struct Reference

Basic timeout specification.

```
#include <__timeout.h>
```

### Data Fields

- [l4\\_uint16\\_t t](#)  
*timeout value*

### 11.103.1 Detailed Description

Basic timeout specification. Basically a floating point number with 10 bits mantissa and 5 bits exponent ( $t = m \cdot 2^e$ ).

The timeout can also specify an absolute point in time (bit 16 == 1).

Definition at line 45 of file [\\_\\_timeout.h](#).

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

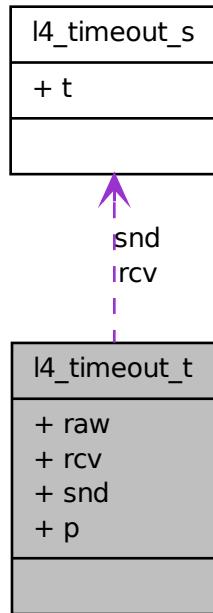
- [l4/sys/\\_\\_timeout.h](#)

## 11.104 l4\_timeout\_t Union Reference

Timeout pair.

```
#include <__timeout.h>
```

Collaboration diagram for l4\_timeout\_t:



## Data Fields

- `l4_uint32_t raw`  
*raw value*
- `struct {`  
    `l4_timeout_s rcv`  
        *receive timeout*  
    `l4_timeout_s snd`  
        *send timeout*  
    `} p`  
  
*combined timeout*

### 11.104.1 Detailed Description

Timeout pair. For IPC there are usually a send and a receive timeout. So this structure contains a pair of timeouts.

Definition at line 57 of file [\\_timeout.h](#).

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

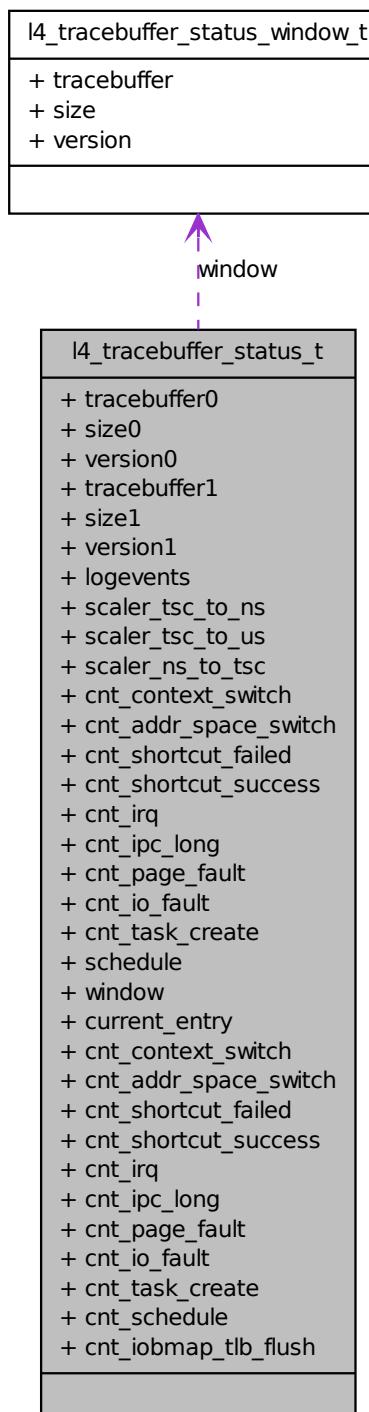
- l4/sys/\_\_timeout.h

## **11.105 l4\_tracebuffer\_status\_t Struct Reference**

Trace buffer status.

```
#include <ktrace.h>
```

Collaboration diagram for l4\_tracebuffer\_status\_t:



## Data Fields

- `l4_umword_t tracebuffer0`  
*Address of trace buffer 0.*
- `l4_umword_t size0`  
*Size of trace buffer 0.*
- `l4_umword_t version0`  
*Version number of trace buffer 0 (incremented if tb0 overruns).*
- `l4_umword_t tracebuffer1`  
*Address of trace buffer 1 (there is no gap between tb0 and tb1).*
- `l4_umword_t size1`  
*Size of trace buffer 1 (same as tb0).*
- `l4_umword_t version1`  
*Version number of trace buffer 1 (incremented if tb1 overruns).*
- `l4_umword_t logevents [16]`  
*Available LOG events.*
- `l4_umword_t scaler_tsc_to_ns`  
*Scaler used for translation of CPU cycles to nano seconds.*
- `l4_umword_t scaler_tsc_to_us`  
*Scaler used for translation of CPU cycles to micro seconds.*
- `l4_umword_t scaler_ns_to_tsc`  
*Scaler used for translation of nano seconds to CPU cycles.*
- `l4_umword_t cnt_context_switch`  
*Number of context switches (intra AS or inter AS).*
- `l4_umword_t cnt_addr_space_switch`  
*Number of inter AS context switches.*
- `l4_umword_t cnt_shortcut_failed`  
*How often was the IPC shortcut not taken.*
- `l4_umword_t cnt_shortcut_success`  
*How often was the IPC shortcut taken.*
- `l4_umword_t cnt_irq`  
*Number of hardware interrupts (without kernel scheduling interrupt).*
- `l4_umword_t cnt_ipc_long`  
*Number of long IPCs.*

- **l4\_umword\_t cnt\_page\_fault**  
*Number of page faults.*
- **l4\_umword\_t cnt\_io\_fault**  
*Number of faults (application runs at IOPL 0 and tries to execute cli, sti, in, or out but does not have a sufficient right in the I/O bitmap).*
- **l4\_umword\_t cnt\_task\_create**  
*Number of tasks created.*
- **l4\_umword\_t schedule**  
*Number of reschedules.*
- volatile **l4\_tracebuffer\_entry\_t \* current\_entry**  
*Address of the most current event in trace-buffer.*
- volatile **l4\_umword\_t cnt\_context\_switch**  
*Number of context switches (intra AS or inter AS).*
- volatile **l4\_umword\_t cnt\_addr\_space\_switch**  
*Number of inter AS context switches.*
- volatile **l4\_umword\_t cnt\_shortcut\_failed**  
*How often was the IPC shortcut taken.*
- volatile **l4\_umword\_t cnt\_shortcut\_success**  
*How often was the IPC shortcut not taken.*
- volatile **l4\_umword\_t cnt\_irq**  
*Number of hardware interrupts (without kernel scheduling interrupt).*
- volatile **l4\_umword\_t cnt\_ipc\_long**  
*Number of long IPCs.*
- volatile **l4\_umword\_t cnt\_page\_fault**  
*Number of page faults.*
- volatile **l4\_umword\_t cnt\_io\_fault**  
*Number of faults (application runs at IOPL 0 and tries to execute cli, sti, in, or out but does not have a sufficient right in the I/O bitmap).*
- volatile **l4\_umword\_t cnt\_task\_create**  
*Number of tasks created.*
- volatile **l4\_umword\_t cnt\_schedule**  
*Number of reschedules.*
- volatile **l4\_umword\_t cnt\_iobmap\_tlb\_flush**  
*Number of flushes of the I/O bitmap.*

### 11.105.1 Detailed Description

Trace buffer status. Trace-buffer status.

Tracebuffer status.

Definition at line 55 of file [ktrace.h](#).

### 11.105.2 Field Documentation

#### 11.105.2.1 l4\_umword\_t l4\_tracebuffer\_status\_t::tracebuffer0

Address of trace buffer 0.

Address of tracebuffer 0.

Definition at line 58 of file [ktrace.h](#).

#### 11.105.2.2 l4\_umword\_t l4\_tracebuffer\_status\_t::size0

Size of trace buffer 0.

Size of tracebuffer 0.

Definition at line 60 of file [ktrace.h](#).

#### 11.105.2.3 l4\_umword\_t l4\_tracebuffer\_status\_t::version0

Version number of trace buffer 0 (incremented if tb0 overruns).

Version number of tracebuffer 0 (incremented if tb0 overruns).

Definition at line 62 of file [ktrace.h](#).

#### 11.105.2.4 l4\_umword\_t l4\_tracebuffer\_status\_t::tracebuffer1

Address of trace buffer 1 (there is no gap between tb0 and tb1).

Address of tracebuffer 1 (there is no gap between tb0 and tb1).

Definition at line 64 of file [ktrace.h](#).

#### 11.105.2.5 l4\_umword\_t l4\_tracebuffer\_status\_t::size1

Size of trace buffer 1 (same as tb0).

Size of tracebuffer 1 (same as tb0).

Definition at line 66 of file [ktrace.h](#).

#### 11.105.2.6 l4\_umword\_t l4\_tracebuffer\_status\_t::version1

Version number of trace buffer 1 (incremented if tb1 overruns).

Version number of tracebuffer 1 (incremented if tb1 overruns).

Definition at line 68 of file [ktrace.h](#).

### 11.105.2.7 volatile l4\_umword\_t l4\_tracebuffer\_status\_t::cnt\_iobmap\_tlb\_flush

Number of flushes of the I/O bitmap.

Increases on context switches between two small address spaces if at least one of the spaces has an I/O bitmap allocated.

Definition at line 99 of file [ktrace.h](#).

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

- arm/l4/sys/ktrace.h
- amd64/l4f/l4/sys/ktrace.h
- x86/l4/sys/ktrace.h

## 11.106 l4\_tracebuffer\_status\_window\_t Struct Reference

Trace-buffer status window descriptor.

```
#include <ktrace.h>
```

### Data Fields

- `l4_tracebuffer_entry_t * tracebuffer`  
*Address of trace-buffer.*
- `l4_umword_t size`  
*Size of trace-buffer.*
- `volatile l4_uint64_t version`  
*Version number of trace-buffer (incremented if trace-buffer overruns).*

### 11.106.1 Detailed Description

Trace-buffer status window descriptor.

Definition at line 45 of file [ktrace.h](#).

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

- x86/l4/sys/ktrace.h

## 11.107 l4\_vcon\_attr\_t Struct Reference

Vcon attribute structure.

```
#include <vcon.h>
```

## Data Fields

- [l4\\_umword\\_t i\\_flags](#)

*input flags*

- [l4\\_umword\\_t o\\_flags](#)

*output flags*

- [l4\\_umword\\_t l\\_flags](#)

*local flags*

### 11.107.1 Detailed Description

Vcon attribute structure.

Definition at line [115](#) of file [vcon.h](#).

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

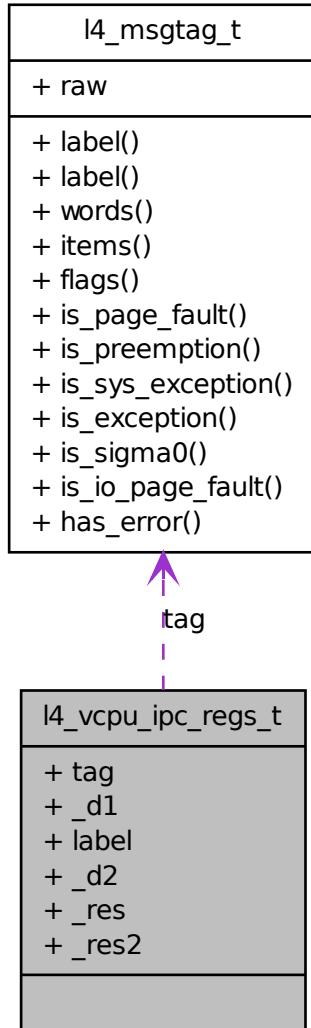
- [l4/sys/vcon.h](#)

## 11.108 l4\_vcpu\_ipc\_regs\_t Struct Reference

vCPU message registers.

```
#include <__vcpu-arch.h>
```

Collaboration diagram for l4\_vcpu\_ipc\_regs\_t:



### 11.108.1 Detailed Description

vCPU message registers.

Definition at line 45 of file [\\_\\_vcpu-arch.h](#).

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

- arm/l4/sys/\_\_vcpu-arch.h
- amd64/l4/sys/\_\_vcpu-arch.h
- x86/l4/sys/\_\_vcpu-arch.h

## 11.109 l4\_vcpu\_regs\_t Struct Reference

vCPU registers.

```
#include <__vcpu-arch.h>
```

### Data Fields

- `l4_umword_t pfa`  
*page fault address*
- `l4_umword_t err`  
*error code*
- `l4_umword_t sp`  
*stack pointer*
- `l4_umword_t ip`  
*instruction pointer*
- `l4_umword_t flags`  
*eflags*
- `l4_umword_t r15`  
*r15 register*
- `l4_umword_t r14`  
*r14 register*
- `l4_umword_t r13`  
*r13 register*
- `l4_umword_t r12`  
*r12 register*
- `l4_umword_t r11`  
*r11 register*
- `l4_umword_t r10`  
*r10 register*
- `l4_umword_t r9`  
*r9 register*
- `l4_umword_t r8`  
*r8 register*
- `l4_umword_t di`  
*rdi register*

- `l4_umword_t si`  
*rsi register*
- `l4_umword_t bp`  
*rbp register*
- `l4_umword_t bx`  
*rbx register*
- `l4_umword_t dx`  
*rdx register*
- `l4_umword_t cx`  
*rcx register*
- `l4_umword_t ax`  
*rax register*
- `l4_umword_t trapno`  
*trap number*
- `l4_umword_t dummy1`  
*dummy*
- `l4_umword_t ss`  
*ss register*
- `l4_umword_t es`  
*gs register*
- `l4_umword_t ds`  
*fs register*
- `l4_umword_t gs`  
*gs register*
- `l4_umword_t fs`  
*fs register*

### 11.109.1 Detailed Description

vCPU registers.

Definition at line [27](#) of file [\\_\\_vcpu-arch.h](#).

## 11.109.2 Field Documentation

### 11.109.2.1 l4\_umword\_t l4\_vcpu\_regs\_t::di

rdi register

edi register

Definition at line 44 of file [\\_\\_vcpu-arch.h](#).

### 11.109.2.2 l4\_umword\_t l4\_vcpu\_regs\_t::si

rsi register

esi register

Definition at line 45 of file [\\_\\_vcpu-arch.h](#).

### 11.109.2.3 l4\_umword\_t l4\_vcpu\_regs\_t::bp

rbp register

ebp register

Definition at line 46 of file [\\_\\_vcpu-arch.h](#).

### 11.109.2.4 l4\_umword\_t l4\_vcpu\_regs\_t::bx

rbx register

ebx register

Definition at line 48 of file [\\_\\_vcpu-arch.h](#).

### 11.109.2.5 l4\_umword\_t l4\_vcpu\_regs\_t::dx

rdx register

edx register

Definition at line 49 of file [\\_\\_vcpu-arch.h](#).

### 11.109.2.6 l4\_umword\_t l4\_vcpu\_regs\_t::cx

rcx register

ecx register

Definition at line 50 of file [\\_\\_vcpu-arch.h](#).

### 11.109.2.7 l4\_umword\_t l4\_vcpu\_regs\_t::ax

rax register

eax register

Definition at line 51 of file [\\_\\_vcpu-arch.h](#).

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

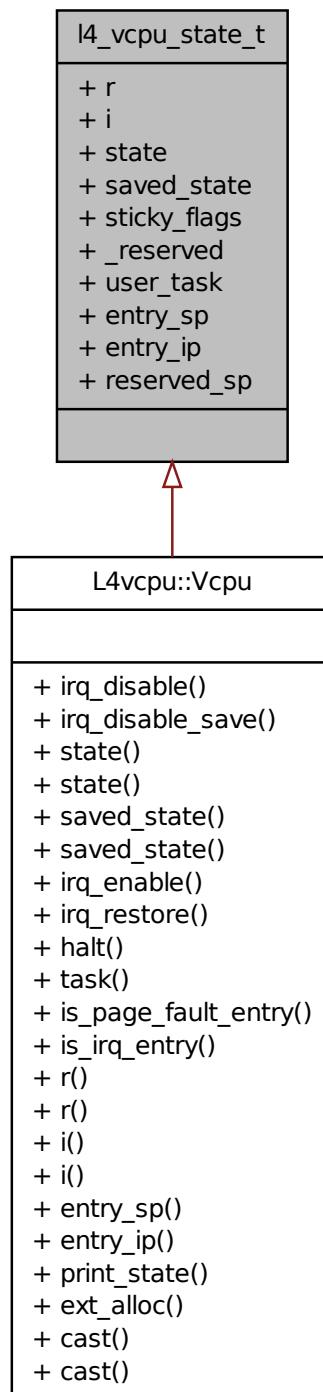
- arm/l4/sys/\_\_vcpu-arch.h
- amd64/l4/sys/\_\_vcpu-arch.h
- x86/l4/sys/\_\_vcpu-arch.h

## 11.110 l4\_vcpu\_state\_t Struct Reference

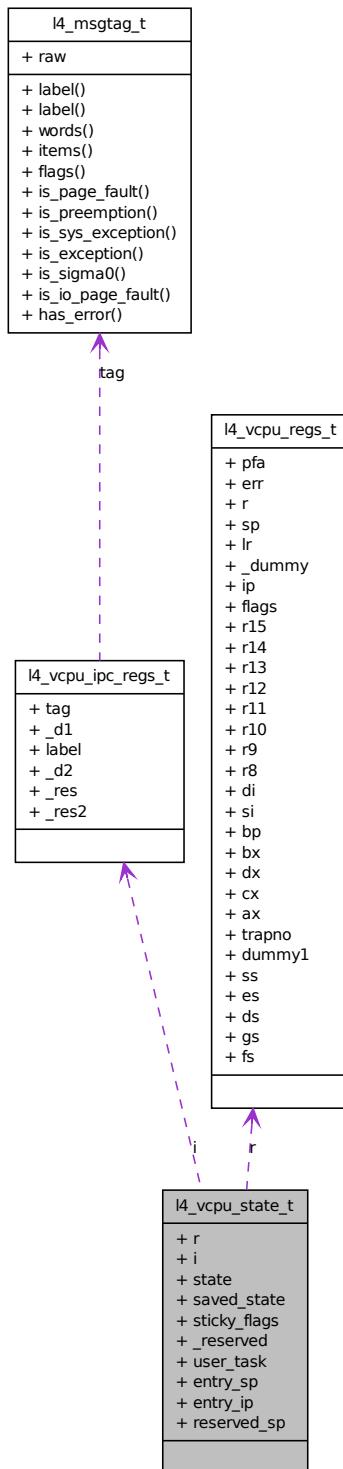
State of a vCPU.

```
#include <vcpu.h>
```

Inheritance diagram for l4\_vcpu\_state\_t:



Collaboration diagram for l4\_vcpu\_state\_t:



## Data Fields

- [l4\\_vcpu\\_regs\\_t r](#)  
*Register state.*
- [l4\\_vcpu\\_ipc\\_regs\\_t i](#)  
*IPC state.*
- [l4\\_uint16\\_t state](#)  
*Current vCPU state.*
- [l4\\_uint16\\_t saved\\_state](#)  
*Saved vCPU state.*
- [l4\\_uint16\\_t sticky\\_flags](#)  
*Pending flags.*
- [l4\\_cap\\_idx\\_t user\\_task](#)  
*User task to use.*
- [l4\\_umword\\_t entry\\_sp](#)  
*Stack pointer for entry (when coming from user task).*
- [l4\\_umword\\_t entry\\_ip](#)  
*IP for entry.*

### 11.110.1 Detailed Description

State of a vCPU.

Definition at line 34 of file [vcpu.h](#).

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

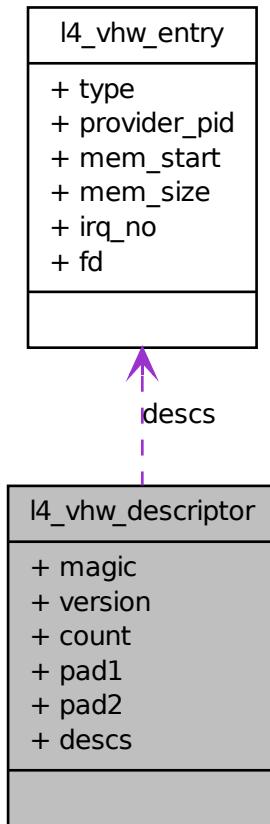
- [l4/sys/vcpu.h](#)

## 11.111 l4\_vhw\_descriptor Struct Reference

Virtual hardware devices description.

```
#include <vhw.h>
```

Collaboration diagram for l4\_vhw\_descriptor:



## Data Fields

- `l4_uint32_t` `magic`  
*Magic.*
- `l4_uint8_t` `version`  
*Version of the descriptor.*
- `l4_uint8_t` `count`  
*Number of entries.*
- `l4_uint8_t` `pad1`  
*padding*
- `l4_uint8_t` `pad2`

*padding*

- struct l4\_vhw\_entry\_descs [ ]

*Array of device descriptions.*

### 11.111.1 Detailed Description

Virtual hardware devices description.

**Examples:**

[examples/sys/ux-vhw/main.c](#).

Definition at line [70](#) of file [vhw.h](#).

### 11.111.2 Field Documentation

#### 11.111.2.1 l4\_uint32\_t l4\_vhw\_descriptor::magic

Magic.

**Examples:**

[examples/sys/ux-vhw/main.c](#).

Definition at line [71](#) of file [vhw.h](#).

#### 11.111.2.2 l4\_uint8\_t l4\_vhw\_descriptor::version

Version of the descriptor.

**Examples:**

[examples/sys/ux-vhw/main.c](#).

Definition at line [72](#) of file [vhw.h](#).

#### 11.111.2.3 l4\_uint8\_t l4\_vhw\_descriptor::count

Number of entries.

**Examples:**

[examples/sys/ux-vhw/main.c](#).

Definition at line [73](#) of file [vhw.h](#).

#### 11.111.2.4 struct l4\_vhw\_entry l4\_vhw\_descriptor::descs[ ]

Array of device descriptions.

Definition at line 77 of file [vhw.h](#).

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

- [l4/sys/vhw.h](#)

### 11.112 l4\_vhw\_entry Struct Reference

Description of a device.

```
#include <vhw.h>
```

#### Data Fields

- enum [l4\\_vhw\\_entry\\_type](#) type  
*Type of virtual hardware.*
- [l4\\_uint32\\_t](#) provider\_pid  
*Host PID of the VHW provider.*
- [l4\\_addr\\_t](#) mem\_start  
*Start of memory region.*
- [l4\\_addr\\_t](#) mem\_size  
*Size of memory region.*
- [l4\\_uint32\\_t](#) irq\_no  
*IRQ number.*
- [l4\\_uint32\\_t](#) fd  
*File descriptor.*

#### 11.112.1 Detailed Description

Description of a device.

#### Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line 55 of file [vhw.h](#).

## 11.112.2 Field Documentation

### 11.112.2.1 enum l4\_vhw\_entry\_type l4\_vhw\_entry::type

Type of virtual hardware.

#### Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line [56](#) of file [vhw.h](#).

### 11.112.2.2 l4\_uint32\_t l4\_vhw\_entry::provider\_pid

Host PID of the VHW provider.

#### Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line [57](#) of file [vhw.h](#).

### 11.112.2.3 l4\_addr\_t l4\_vhw\_entry::mem\_start

Start of memory region.

#### Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line [59](#) of file [vhw.h](#).

### 11.112.2.4 l4\_addr\_t l4\_vhw\_entry::mem\_size

Size of memory region.

#### Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line [60](#) of file [vhw.h](#).

### 11.112.2.5 l4\_uint32\_t l4\_vhw\_entry::irq\_no

IRQ number.

#### Examples:

[examples/sys/ux-vhw/main.c](#).

Definition at line [62](#) of file [vhw.h](#).

### 11.112.2.6 l4\_uint32\_t l4\_vhw\_entry::fd

File descriptor.

Definition at line 63 of file [vhw.h](#).

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

- l4/sys/vhw.h

## 11.113 l4\_vm\_state Struct Reference

state structure for TrustZone VMs

```
#include <vm.h>
```

### 11.113.1 Detailed Description

state structure for TrustZone VMs

Definition at line 38 of file [vm.h](#).

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

- arm/l4/sys/vm.h

## 11.114 l4\_vm\_svm\_vmcb\_control\_area Struct Reference

VMCB structure for SVM VMs.

```
#include <__vm-svm.h>
```

### 11.114.1 Detailed Description

VMCB structure for SVM VMs.

Definition at line 40 of file [\\_\\_vm-svm.h](#).

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

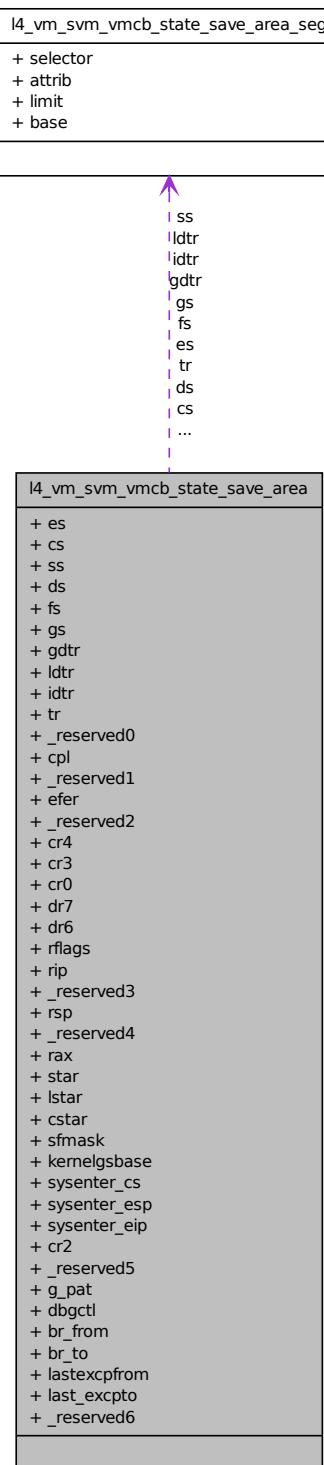
- l4/sys/\_\_vm-svm.h

## 11.115 l4\_vm\_svm\_vmcb\_state\_save\_area Struct Reference

State save area structure for SVM VMs.

```
#include <__vm-svm.h>
```

Collaboration diagram for l4\_vm\_svm\_vmcb\_state\_save\_area:



### 11.115.1 Detailed Description

State save area structure for SVM VMs.

Definition at line [92](#) of file [\\_\\_vm-svm.h](#).

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

- l4/sys/\_\_vm-svm.h

## 11.116 l4\_vm\_svm\_vmcb\_state\_save\_area\_seg Struct Reference

State save area segment selector struct.

```
#include <__vm-svm.h>
```

### 11.116.1 Detailed Description

State save area segment selector struct.

Definition at line [80](#) of file [\\_\\_vm-svm.h](#).

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

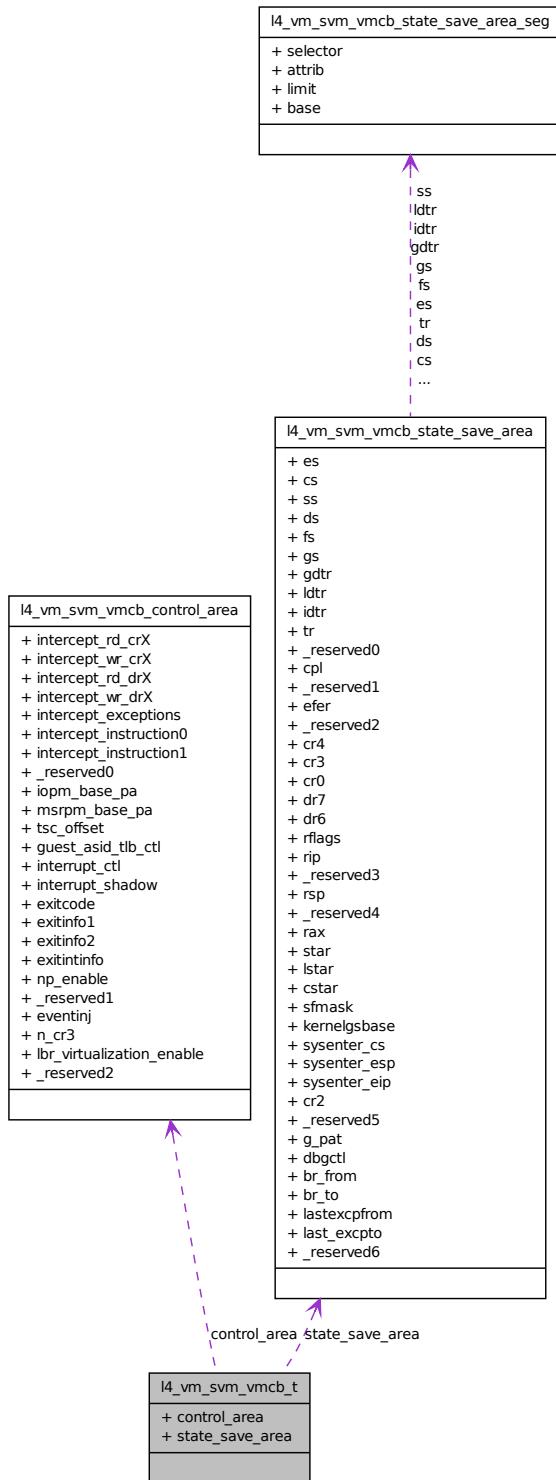
- l4/sys/\_\_vm-svm.h

## 11.117 l4\_vm\_svm\_vmcb\_t Struct Reference

Control structure for SVM VMs.

```
#include <__vm-svm.h>
```

Collaboration diagram for l4\_vm\_svm\_vmcb\_t:



### 11.117.1 Detailed Description

Control structure for SVM VMs.

Definition at line 157 of file [\\_\\_vm-svm.h](#).

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

- l4/sys/\_\_vm-svm.h

## 11.118 l4re\_aux\_t Struct Reference

Auxiliary descriptor.

```
#include <l4aux.h>
```

### Data Fields

- char const \* [binary](#)  
*Binary name.*
- [l4\\_cap\\_idx\\_t kip\\_ds](#)  
*Data space of the KIP.*
- [l4\\_umword\\_t dbg\\_lvl](#)  
*Debug levels for l4re.*
- [l4\\_umword\\_t ldr\\_flags](#)  
*Flags for l4re, see [l4re\\_aux\\_ldr\\_flags\\_t](#).*

### 11.118.1 Detailed Description

Auxiliary descriptor.

Definition at line 51 of file [l4aux.h](#).

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

- l4/re/l4aux.h

## 11.119 l4re\_ds\_stats\_t Struct Reference

Information about the data space.

```
#include <dataspace.h>
```

### Data Fields

- unsigned long [size](#)

*size*

- unsigned long [flags](#)

*flags*

### 11.119.1 Detailed Description

Information about the data space.

Definition at line [45](#) of file [dataspace.h](#).

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

- [l4/re/c/dataspace.h](#)

## 11.120 l4re\_elf\_aux\_mword\_t Struct Reference

Auxiliary vector element for a single unsigned data word.

```
#include <elf_aux.h>
```

### 11.120.1 Detailed Description

Auxiliary vector element for a single unsigned data word.

Definition at line [125](#) of file [elf\\_aux.h](#).

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

- [l4/re/elf\\_aux.h](#)

## 11.121 l4re\_elf\_aux\_t Struct Reference

Generic header for each auxiliary vector element.

```
#include <elf_aux.h>
```

### 11.121.1 Detailed Description

Generic header for each auxiliary vector element.

Definition at line [105](#) of file [elf\\_aux.h](#).

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

- [l4/re/elf\\_aux.h](#)

## 11.122 l4re\_elf\_aux\_vma\_t Struct Reference

Auxiliary vector element for a reserved virtual memory area.

```
#include <elf_aux.h>
```

### 11.122.1 Detailed Description

Auxiliary vector element for a reserved virtual memory area.

Definition at line 114 of file [elf\\_aux.h](#).

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

- [14/re/elf\\_aux.h](#)

## 11.123 l4re\_env\_cap\_entry\_t Struct Reference

Entry in the [L4Re](#) environment array for the named initial objects.

```
#include <env.h>
```

### Public Member Functions

- [l4re\\_env\\_cap\\_entry\\_t \(\)](#)  
*Create an invalid entry.*
- [l4re\\_env\\_cap\\_entry\\_t \(char const \\*n, l4\\_cap\\_idx\\_t c, l4\\_umword\\_t f=0\)](#)  
*Create an entry with the name n, capability c, and flags f.*

### Data Fields

- [l4\\_cap\\_idx\\_t cap](#)  
*The capability selector for the object.*
- [l4\\_umword\\_t flags](#)  
*Some flags for the object.*
- [char name \[16\]](#)  
*The name of the object.*

### 11.123.1 Detailed Description

Entry in the [L4Re](#) environment array for the named initial objects.

Definition at line 35 of file [env.h](#).

### 11.123.2 Constructor & Destructor Documentation

**11.123.2.1 l4re\_env\_cap\_entry\_t::l4re\_env\_cap\_entry\_t ( `char const * n, l4_cap_idx_t c,`  
`l4_umword_t f = 0` ) [inline]**

Create an entry with the name *n*, capability *c*, and flags *f*.

#### Parameters

*n* is the name of the initial object.

*c* is the capability selector that refers the initial object.

*f* are the additional flags for the object.

Definition at line 67 of file [env.h](#).

References [name](#).

### 11.123.3 Field Documentation

**11.123.3.1 l4\_umword\_t l4re\_env\_cap\_entry\_t::flags**

Some flags for the object.

#### Note

Currently unused.

Definition at line 46 of file [env.h](#).

Referenced by [l4re\\_get\\_env\\_cap\\_l\(\)](#).

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

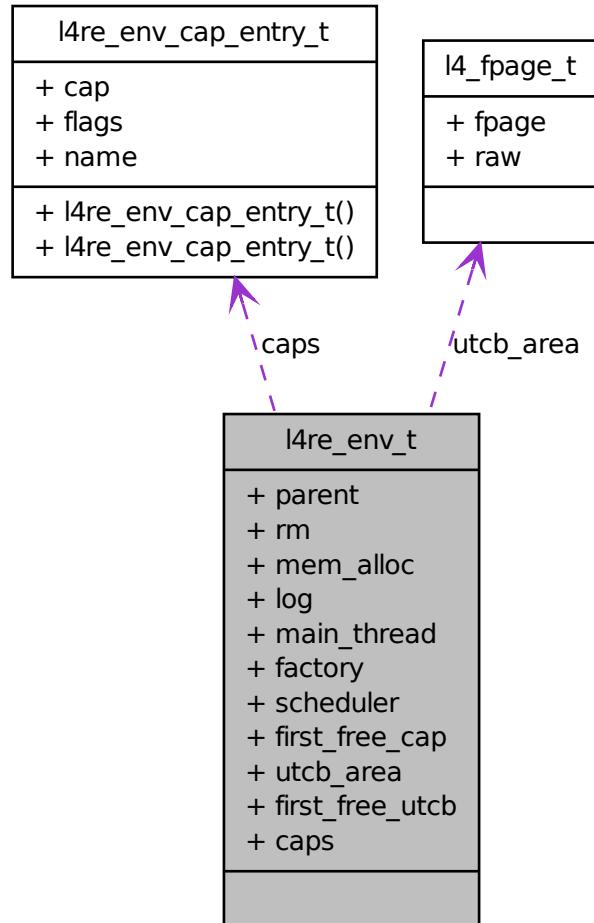
- l4/re/env.h

## 11.124 l4re\_env\_t Struct Reference

Initial Environment structure (C version).

```
#include <env.h>
```

Collaboration diagram for l4re\_env\_t:



## Data Fields

- [l4\\_cap\\_idx\\_t parent](#)  
*Parent object-capability.*
- [l4\\_cap\\_idx\\_t rm](#)  
*Region map object-capability.*
- [l4\\_cap\\_idx\\_t mem\\_alloc](#)  
*Memory allocator object-capability.*
- [l4\\_cap\\_idx\\_t log](#)

*Logging object-capability.*

- [l4\\_cap\\_idx\\_t main\\_thread](#)  
*Object-capability of the first user thread.*
- [l4\\_cap\\_idx\\_t factory](#)  
*Object-capability of the factory available to the task.*
- [l4\\_cap\\_idx\\_t scheduler](#)  
*Object capability for the scheduler set to use.*
- [l4\\_cap\\_idx\\_t first\\_free\\_cap](#)  
*First capability index available to the application.*
- [l4\\_fpage\\_t utcb\\_area](#)  
*UTCB area of the task.*
- [l4\\_addr\\_t first\\_free\\_utcb](#)  
*First UTCB within the UTCB area available to the application.*

### 11.124.1 Detailed Description

Initial Environment structure (C version).

#### See also

[Initial environment](#)

Definition at line 87 of file [env.h](#).

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

- [l4/re/env.h](#)

## 11.125 l4re\_event\_t Struct Reference

Event structure used in buffer.

```
#include <event.h>
```

### Data Fields

- [long long time](#)  
*Time stamp of the event.*
- [unsigned short type](#)  
*Type of the event.*
- [unsigned short code](#)

*Code of the event.*

- int **value**

*Value of the event.*

- **l4\_umword\_t stream\_id**

*Stream ID.*

### 11.125.1 Detailed Description

Event structure used in buffer.

Definition at line 39 of file [event.h](#).

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

- [l4/re/c/event.h](#)

## 11.126 l4re\_video\_color\_component\_t Struct Reference

Color component structure.

```
#include <colors.h>
```

### Data Fields

- unsigned char **size**

*Size in bits.*

- unsigned char **shift**

*offset in pixel*

### 11.126.1 Detailed Description

Color component structure.

Definition at line 29 of file [colors.h](#).

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

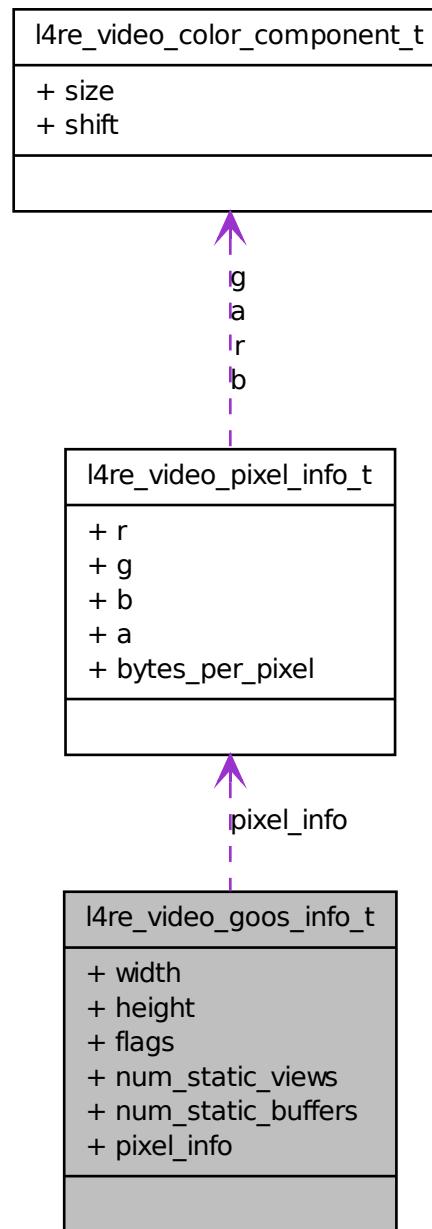
- [l4/re/c/video/colors.h](#)

## 11.127 l4re\_video\_goos\_info\_t Struct Reference

Goos information structure.

```
#include <goos.h>
```

## Collaboration diagram for 14re\_video\_goops\_info\_t:



## Data Fields

- `unsigned long width`  
*Width of the goos.*

- unsigned long [height](#)

*Height of the goos.*

- unsigned [flags](#)

*Flags of the framebuffer.*

- unsigned [num\\_static\\_views](#)

*Number of static views.*

- unsigned [num\\_static\\_buffers](#)

*Number of static buffers.*

- [l4re\\_video\\_pixel\\_info\\_t pixel\\_info](#)

*Pixel layout of the goos.*

## 11.127.1 Detailed Description

Goos information structure.

Definition at line [51](#) of file [goos.h](#).

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

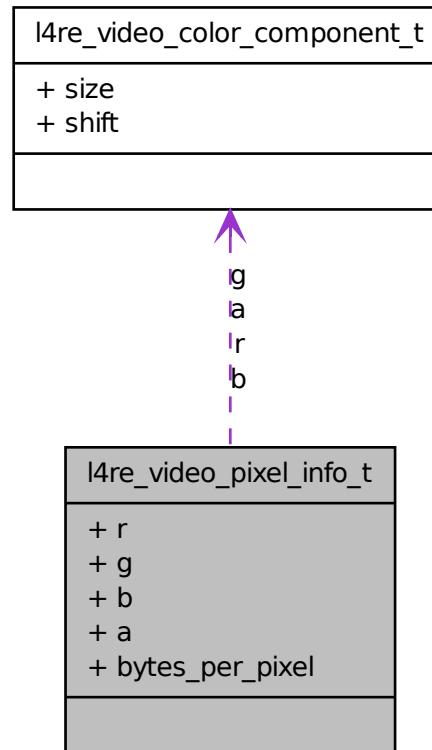
- [l4/re/c/video/goos.h](#)

## 11.128 l4re\_video\_pixel\_info\_t Struct Reference

Pixel\_info structure.

```
#include <colors.h>
```

Collaboration diagram for l4re\_video\_pixel\_info\_t:



## Data Fields

- l4re\_video\_color\_component\_t a  
*Colors.*
- unsigned char bytes\_per\_pixel  
*Bytes per pixel.*

### 11.128.1 Detailed Description

Pixel\_info structure.

Definition at line 39 of file [colors.h](#).

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

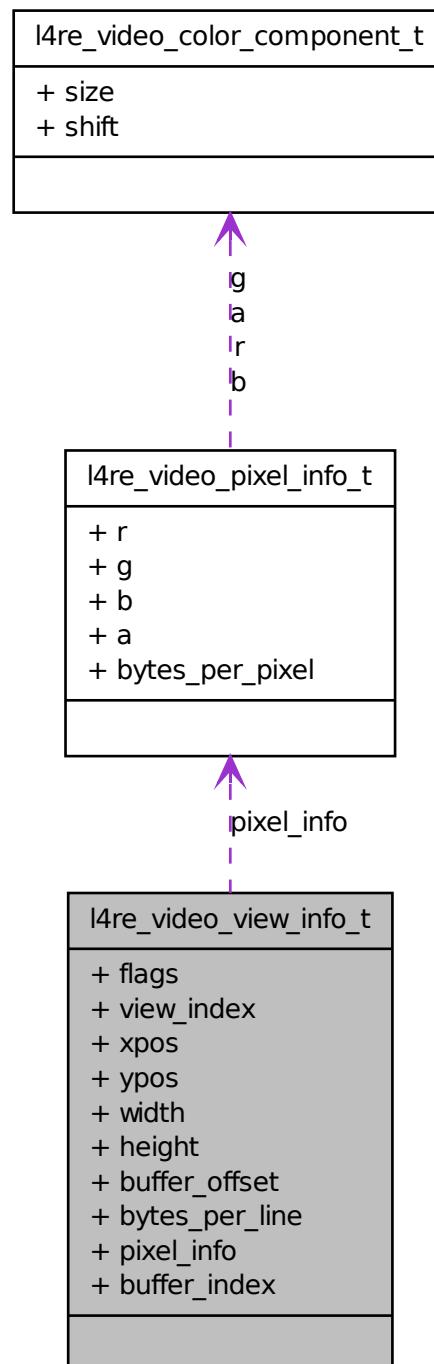
- l4/re/c/video/colors.h

## 11.129 l4re\_video\_view\_info\_t Struct Reference

View information structure.

```
#include <view.h>
```

Collaboration diagram for l4re\_video\_view\_info\_t:



## Data Fields

- `unsigned flags`  
*Flags.*
- `unsigned view_index`  
*Number of view in the goos.*
- `unsigned long height`  
*Position in goos and size of view.*
- `unsigned long buffer_offset`  
*Memory offset in goos buffer.*
- `unsigned long bytes_per_line`  
*Size of line in view.*
- `l4re_video_pixel_info_t pixel_info`  
*Pixel info.*
- `unsigned buffer_index`  
*Number of buffer of goos.*

### 11.129.1 Detailed Description

View information structure.

Definition at line 59 of file `view.h`.

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

- `l4/re/c/video/view.h`

## 11.130 l4re\_video\_view\_t Struct Reference

C representation of a goos view.

```
#include <view.h>
```

### 11.130.1 Detailed Description

C representation of a goos view. A view is a visible rectangle that provides a view to the contents of a buffer (frame buffer) memory object and is placed on a real screen.

Definition at line 78 of file `view.h`.

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

- `l4/re/c/video/view.h`

## 11.131 l4util\_idt\_desc\_t Struct Reference

IDT entry.

```
#include <idt.h>
```

### Data Fields

- [l4\\_uint64\\_t b](#)

*see Intel doc*

- [l4\\_uint32\\_t b](#)

*see Intel doc*

### 11.131.1 Detailed Description

IDT entry.

Definition at line [33](#) of file [idt.h](#).

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

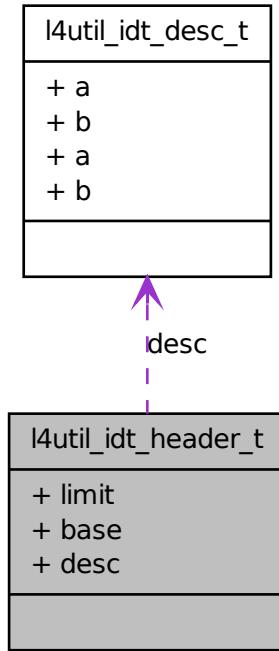
- [amd64/l4/util/idt.h](#)
- [x86/l4/util/idt.h](#)

## 11.132 l4util\_idt\_header\_t Struct Reference

Header of an IDT table.

```
#include <idt.h>
```

Collaboration diagram for l4util\_idt\_header\_t:



## Data Fields

- [l4\\_uint16\\_t limit](#)  
*limit field (see Intel doc)*
- [void \\* base](#)  
*idt base (see Intel doc)*

### 11.132.1 Detailed Description

Header of an IDT table.

Definition at line [40](#) of file [idt.h](#).

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

- [amd64/l4/util/idt.h](#)
- [x86/l4/util/idt.h](#)

## 11.133 l4util\_mb\_addr\_range\_t Struct Reference

INT-15, AX=E820 style "AddressRangeDescriptor" ...with a "size" parameter on the front which is the structure size - 4, pointing to the next one, up until the full buffer length of the memory map has been reached.

```
#include <mb_info.h>
```

### Data Fields

- `l4_uint64_t addr`  
*<Size of structure*
- `l4_uint64_t size`  
*<Start address*
- `l4_uint32_t type`  
*<Size of memory range*

### 11.133.1 Detailed Description

INT-15, AX=E820 style "AddressRangeDescriptor" ...with a "size" parameter on the front which is the structure size - 4, pointing to the next one, up until the full buffer length of the memory map has been reached.

Definition at line 43 of file [mb\\_info.h](#).

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

- `l4/util/mb_info.h`

## 11.134 l4util\_mb\_apm\_t Struct Reference

APM BIOS info.

```
#include <mb_info.h>
```

### 11.134.1 Detailed Description

APM BIOS info.

Definition at line 90 of file [mb\\_info.h](#).

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

- `l4/util/mb_info.h`

## 11.135 l4util\_mb\_drive\_t Struct Reference

Drive Info structure.

```
#include <mb_info.h>
```

### Data Fields

- `l4_uint8_t drive_number`  
*<The size of this structure.*
- `l4_uint8_t drive_mode`  
*<The BIOS drive number.*
- `l4_uint16_t drive_cylinders`  
*<The access mode (see below).*
- `l4_uint8_t drive_heads`  
*<number of cylinders*
- `l4_uint8_t drive_sectors`  
*<number of heads*
- `l4_uint16_t drive_ports [0]`  
*<number of sectors per track*

### 11.135.1 Detailed Description

Drive Info structure.

Definition at line [73](#) of file `mb_info.h`.

### 11.135.2 Field Documentation

#### 11.135.2.1 l4\_uint8\_t l4util\_mb\_drive\_t::drive\_number

*<The size of this structure.*

Definition at line [76](#) of file `mb_info.h`.

#### 11.135.2.2 l4\_uint8\_t l4util\_mb\_drive\_t::drive\_mode

*<The BIOS drive number.*

Definition at line [77](#) of file `mb_info.h`.

### 11.135.2.3 l4\_uint16\_t l4util\_mb\_drive\_t::drive\_cylinders

<The access mode (see below).

Definition at line 78 of file [mb\\_info.h](#).

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

- [l4/util\(mb\\_info.h\)](#)

## 11.136 l4util\_mb\_info\_t Struct Reference

```
#include <mb_info.h>
```

### Data Fields

- [l4\\_uint32\\_t flags](#)  
*MultiBoot info version number.*
- [l4\\_uint32\\_t mem\\_lower](#)  
*available memory below 1MB*
- [l4\\_uint32\\_t mem\\_upper](#)  
*available memory starting from 1MB [kB]*
- [l4\\_uint32\\_t boot\\_device](#)  
*"root" partition*
- [l4\\_uint32\\_t cmdline](#)  
*Kernel command line.*
- [l4\\_uint32\\_t mods\\_count](#)  
*number of modules*
- [l4\\_uint32\\_t mods\\_addr](#)  
*module list*
- [l4\\_uint32\\_t mmap\\_length](#)  
*size of memory mapping buffer*
- [l4\\_uint32\\_t mmap\\_addr](#)  
*address of memory mapping buffer*
- [l4\\_uint32\\_t drives\\_length](#)  
*size of drive info buffer*
- [l4\\_uint32\\_t drives\\_addr](#)  
*address of driver info buffer*
- [l4\\_uint32\\_t config\\_table](#)

*ROM configuration table.*

- `l4_uint32_t boot_loader_name`  
*Boot Loader Name.*
- `l4_uint32_t apm_table`  
*APM table.*
- `l4_uint32_t vbe_ctrl_info`  
*VESA video controller info.*
- `l4_uint32_t vbe_mode_info`  
*VESA video mode info.*
- `l4_uint16_t vbe_mode`  
*VESA video mode number.*
- `l4_uint16_t vbe_interface_seg`  
*VESA segment of prot BIOS interface.*
- `l4_uint16_t vbe_interface_off`  
*VESA offset of prot BIOS interface.*
- `l4_uint16_t vbe_interface_len`  
*VESA lenght of prot BIOS interface.*
- `l4_uint32_t tabsize`  
*(a.out) Kernel symbol table info*
- `l4_uint32_t num`  
*(ELF) Kernel section header table*

### 11.136.1 Detailed Description

MultiBoot Info description

This is the struct passed to the boot image. This is done by placing its address in the EAX register.

Definition at line 202 of file `mb_info.h`.

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

- `l4/util/mb_info.h`

### 11.137 l4util\_mb\_mod\_t Struct Reference

```
#include <mb_info.h>
```

## Data Fields

- `l4_uint32_t mod_start`  
*Starting address of module in memory.*
- `l4_uint32_t mod_end`  
*End address of module in memory.*
- `l4_uint32_t cmdline`  
*Module command line.*
- `l4_uint32_t pad`  
*padding to take it to 16 bytes*

### 11.137.1 Detailed Description

The structure type "mod\_list" is used by the [multiboot\\_info](#) structure.

Definition at line [27](#) of file [mb\\_info.h](#).

### 11.137.2 Field Documentation

#### 11.137.2.1 l4\_uint32\_t l4util\_mb\_mod\_t::mod\_start

Starting address of module in memory.

Definition at line [29](#) of file [mb\\_info.h](#).

#### 11.137.2.2 l4\_uint32\_t l4util\_mb\_mod\_t::mod\_end

End address of module in memory.

Definition at line [30](#) of file [mb\\_info.h](#).

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

- `l4/util/mb_info.h`

## 11.138 l4util\_mb\_vbe\_ctrl\_t Struct Reference

VBE controller information.

```
#include <mb_info.h>
```

### 11.138.1 Detailed Description

VBE controller information.

Definition at line [104](#) of file [mb\\_info.h](#).

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

- l4/util(mb\_info.h

## 11.139 l4util\_mb\_vbe\_mode\_t Struct Reference

VBE mode information.

```
#include <mb_info.h>
```

### Data Fields

all VESA versions

- l4\_uint16\_t mode\_attributes
- l4\_uint8\_t win\_a\_attributes
- l4\_uint8\_t win\_b\_attributes
- l4\_uint16\_t win\_granularity
- l4\_uint16\_t win\_size
- l4\_uint16\_t win\_a\_segment
- l4\_uint16\_t win\_b\_segment
- l4\_uint32\_t win\_func
- l4\_uint16\_t bytes\_per\_scanline

>= VESA version 1.2

- l4\_uint16\_t x\_resolution
- l4\_uint16\_t y\_resolution
- l4\_uint8\_t x\_char\_size
- l4\_uint8\_t y\_char\_size
- l4\_uint8\_t number\_of\_planes
- l4\_uint8\_t bits\_per\_pixel
- l4\_uint8\_t number\_of\_banks
- l4\_uint8\_t memory\_model
- l4\_uint8\_t bank\_size
- l4\_uint8\_t number\_of\_image\_pages
- l4\_uint8\_t reserved0

direct color

- l4\_uint8\_t red\_mask\_size
- l4\_uint8\_t red\_field\_position
- l4\_uint8\_t green\_mask\_size
- l4\_uint8\_t green\_field\_position
- l4\_uint8\_t blue\_mask\_size
- l4\_uint8\_t blue\_field\_position
- l4\_uint8\_t reserved\_mask\_size
- l4\_uint8\_t reserved\_field\_position
- l4\_uint8\_t direct\_color\_mode\_info

>= VESA version 2.0

- l4\_uint32\_t phys\_base
- l4\_uint32\_t reserved1
- l4\_uint16\_t reversed2

>= VESA version 3.0

- `l4_uint16_t linear_bytes_per_scanline`
- `l4_uint8_t banked_number_of_image_pages`
- `l4_uint8_t linear_number_of_image_pages`
- `l4_uint8_t linear_red_mask_size`
- `l4_uint8_t linear_red_field_position`
- `l4_uint8_t linear_green_mask_size`
- `l4_uint8_t linear_green_field_position`
- `l4_uint8_t linear_blue_mask_size`
- `l4_uint8_t linear_blue_field_position`
- `l4_uint8_t linear_reserved_mask_size`
- `l4_uint8_t linear_reserved_field_position`
- `l4_uint32_t max_pixel_clock`
- `l4_uint8_t reserved3 [189+1]`

### 11.139.1 Detailed Description

VBE mode information.

Definition at line 122 of file `mb_info.h`.

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

- `l4/util/mb_info.h`

## 11.140 cxx::List< D, Alloc > Class Template Reference

Doubly linked list, with internal allocation.

### Data Structures

- class `Iter`

*Iterator.*

### Public Member Functions

- `void push_back (D const &d) throw ()`  
*Add element at the end of the list.*
- `void push_front (D const &d) throw ()`  
*Add element at the beginning of the list.*
- `void remove (Iter const &i) throw ()`  
*Remove element pointed to by the iterator.*
- `unsigned long size () const throw ()`  
*Get the length of the list.*
- `D const & operator[ ] (unsigned long idx) const throw ()`  
*Random access.*

- `D & operator[ ]` (unsigned long idx) throw ()

*Random access.*

- `Iter items ()` throw ()

*Get iterator for the list elements.*

### 11.140.1 Detailed Description

```
template<typename D, template< typename A > class Alloc = New_allocator> class cxx::List< D, Alloc >
```

Doubly linked list, with internal allocation. Container for items of type D, implemented by a doubly linked list. Alloc defines the allocator policy.

Definition at line 300 of file [list](#).

### 11.140.2 Member Function Documentation

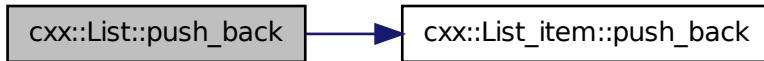
**11.140.2.1** `template<typename D , template< typename A > class Alloc = New_allocator> void cxx::List< D, Alloc >::push_back ( D const & d ) throw () [inline]`

Add element at the end of the list.

Definition at line 346 of file [list](#).

References [cxx::List\\_item::push\\_back\(\)](#).

Here is the call graph for this function:



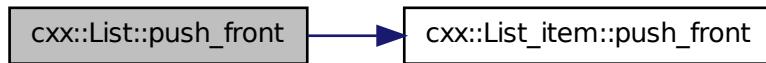
**11.140.2.2** `template<typename D , template< typename A > class Alloc = New_allocator> void cxx::List< D, Alloc >::push_front ( D const & d ) throw () [inline]`

Add element at the beginning of the list.

Definition at line 355 of file [list](#).

References [cxx::List\\_item::push\\_front\(\)](#).

Here is the call graph for this function:



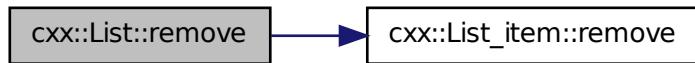
#### 11.140.2.3 template<typename D , template< typename A > class Alloc = New\_allocator> void cxx::List< D, Alloc >::remove ( Iter const & i ) throw () [inline]

Remove element pointed to by the iterator.

Definition at line 364 of file [list](#).

References [cxx::List\\_item::remove\(\)](#).

Here is the call graph for this function:



#### 11.140.2.4 template<typename D , template< typename A > class Alloc = New\_allocator> unsigned long cxx::List< D, Alloc >::size ( ) const throw () [inline]

Get the length of the list.

Definition at line 368 of file [list](#).

#### 11.140.2.5 template<typename D , template< typename A > class Alloc = New\_allocator> D const& cxx::List< D, Alloc >::operator[] ( unsigned long idx ) const throw () [inline]

Random access.

Complexity is O(n).

Definition at line 371 of file [list](#).

---

**11.140.2.6 template<typename D , template< typename A > class Alloc = New\_allocator> D& cxx::List< D, Alloc >::operator[ ]( unsigned long idx ) throw () [inline]**

Random access.

Complexity is O(n).

Definition at line [375](#) of file [list](#).

**11.140.2.7 template<typename D , template< typename A > class Alloc = New\_allocator> Iter cxx::List< D, Alloc >::items( ) throw () [inline]**

Get iterator for the list elements.

Definition at line [379](#) of file [list](#).

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

- l4/cxx/list

## 11.141 cxx::List\_alloc Class Reference

Standard list-based allocator.

### Public Member Functions

- **List\_alloc()**  
*Initializes an empty list allocator.*
- **void free( void \*block, unsigned long size, bool initial\_free=false)**  
*Return a free memory block to the allocator.*
- **void \* alloc( unsigned long size, unsigned align)**  
*Alloc a memory block.*
- **unsigned long avail()**  
*Get the amount of available memory.*

### 11.141.1 Detailed Description

Standard list-based allocator.

Definition at line [29](#) of file [list\\_alloc](#).

### 11.141.2 Constructor & Destructor Documentation

**11.141.2.1 cxx::List\_alloc::List\_alloc( ) [inline]**

Initializes an empty list allocator.

**Note**

To initialize the allocator with available memory use the [free\(\)](#) function.

Definition at line 48 of file [list\\_alloc](#).

### 11.141.3 Member Function Documentation

#### 11.141.3.1 void cxx::List\_alloc::free ( void \* *block*, unsigned long *size*, bool *initial\_free* = *false* ) [inline]

Return a free memory block to the allocator.

**Parameters**

*block* pointer to memory block

*size* size of memory block

*initial\_free* Set to true for putting fresh memory to the allocator. This will enforce alignment on that memory.

Definition at line 199 of file [list\\_alloc](#).

#### 11.141.3.2 void \* cxx::List\_alloc::alloc ( unsigned long *size*, unsigned *align* ) [inline]

Alloc a memory block.

**Parameters**

*size* Size of the memory block

*align* Alignment constraint

**Returns**

Pointer to memory block

Definition at line 237 of file [list\\_alloc](#).

#### 11.141.3.3 unsigned long cxx::List\_alloc::avail ( ) [inline]

Get the amount of available memory.

**Returns**

Available memory in bytes

Definition at line 308 of file [list\\_alloc](#).

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

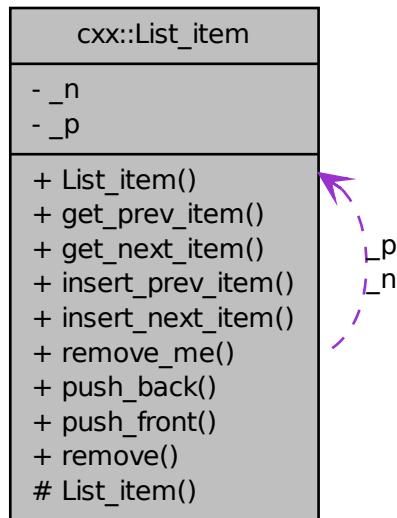
- l4/cxx/list\_alloc

## 11.142 cxx::List\_item Class Reference

Basic list item.

Inherited by cxx::Base\_slab< Obj\_size, Slab\_size, Max\_free, Alloc >::Slab\_head, and cxx::List< D, Alloc >::E.

Collaboration diagram for cxx::List\_item:



## Data Structures

- class [Iter](#)

*Iterator for a list of ListItem-s.*

- class [T\\_iter](#)

*Iterator for derived classes from ListItem.*

## Public Member Functions

- [List\\_item \\* get\\_prev\\_item \(\) const throw \(\)](#)

*Get previous item.*

- [List\\_item \\* get\\_next\\_item \(\) const throw \(\)](#)

*Get next item.*

- [void insert\\_prev\\_item \(List\\_item \\*p\) throw \(\)](#)

*Insert item p before this item.*

- void [insert\\_next\\_item](#) (List\_item \*p) throw ()

*Insert item p after this item.*

- void [remove\\_me](#) () throw ()

*Remove this item from the list.*

## Static Public Member Functions

- template<typename C , typename N >  
static C \* [push\\_back](#) (C \*head, N \*p) throw ()

*Append item to a list.*

- template<typename C , typename N >  
static C \* [push\\_front](#) (C \*head, N \*p) throw ()

*Prepend item to a list.*

- template<typename C , typename N >  
static C \* [remove](#) (C \*head, N \*p) throw ()

*Remove item from a list.*

### 11.142.1 Detailed Description

Basic list item. Basic item that can be member of a doubly linked, cyclic list.

Definition at line [37](#) of file [list](#).

### 11.142.2 Member Function Documentation

#### 11.142.2.1 List\_item\* cxx::List\_item::get\_prev\_item ( ) const throw () [inline]

Get previous item.

Definition at line [174](#) of file [list](#).

#### 11.142.2.2 List\_item\* cxx::List\_item::get\_next\_item ( ) const throw () [inline]

Get next item.

Definition at line [177](#) of file [list](#).

#### 11.142.2.3 void cxx::List\_item::insert\_prev\_item ( List\_item \* p ) throw () [inline]

Insert item p before this item.

Definition at line [180](#) of file [list](#).

#### **11.142.2.4 void cxx::List\_item::insert\_next\_item ( List\_item \* p ) throw () [inline]**

Insert item p after this item.

Definition at line 190 of file [list](#).

#### **11.142.2.5 void cxx::List\_item::remove\_me ( ) throw () [inline]**

Remove this item from the list.

Definition at line 199 of file [list](#).

Referenced by [cxx::List\\_item::Iter::remove\\_me\(\)](#).

Here is the caller graph for this function:



#### **11.142.2.6 template<typename C , typename N > C \* cxx::List\_item::push\_back ( C \* head, N \* p ) throw () [inline, static]**

Append item to a list.

Convinience function for empty-head corner case.

##### **Parameters**

*h* pointer to the current list head.

*p* pointer to new item.

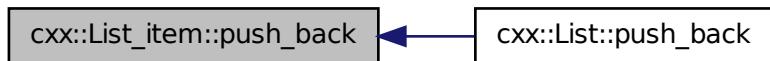
##### **Returns**

the pointer to the new head.

Definition at line 249 of file [list](#).

Referenced by [cxx::List< D, Alloc >::push\\_back\(\)](#).

Here is the caller graph for this function:



**11.142.2.7 template<typename C , typename N > C \* cxx::List\_item::push\_front ( C \* *head*, N \* *p* ) throw () [inline, static]**

Prepend item to a list.

Convinience function for empty-head corner case.

#### Parameters

*head* pointer to the current list head.

*p* pointer to new item.

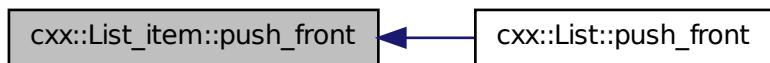
#### Returns

the pointer to the new head.

Definition at line 260 of file [list](#).

Referenced by [cxx::List< D, Alloc >::push\\_front\(\)](#).

Here is the caller graph for this function:



**11.142.2.8 template<typename C , typename N > C \* cxx::List\_item::remove ( C \* *head*, N \* *p* ) throw () [inline, static]**

Remove item from a list.

Convinience function for remove-head corner case.

#### Parameters

*head* pointer to the current list head.

*p* pointer to the item to remove.

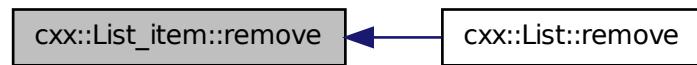
#### Returns

the pointer to the new head.

Definition at line 270 of file [list](#).

Referenced by [cxx::List< D, Alloc >::remove\(\)](#).

Here is the caller graph for this function:



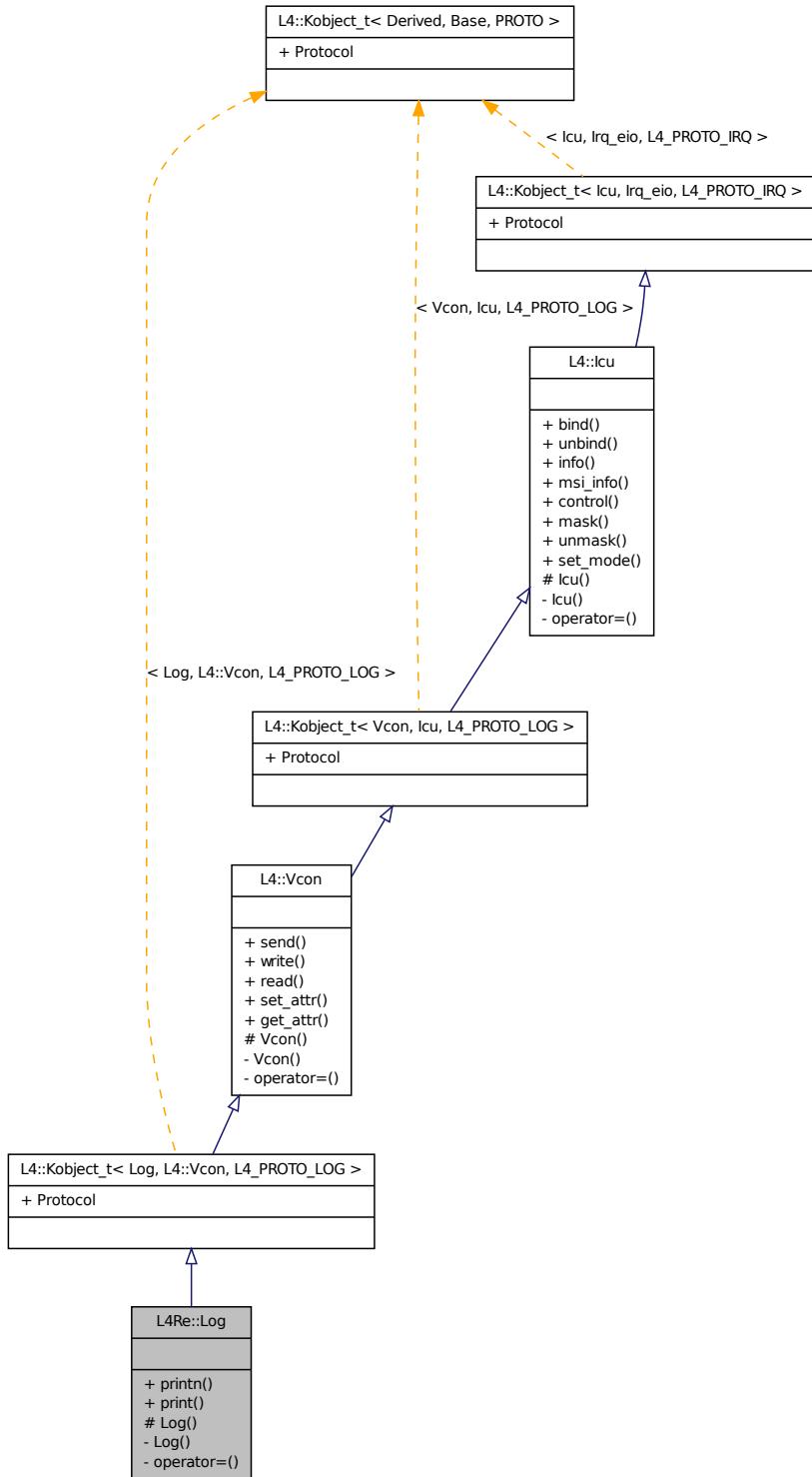
The documentation for this class was generated from the following file:

- l4/cxx/list

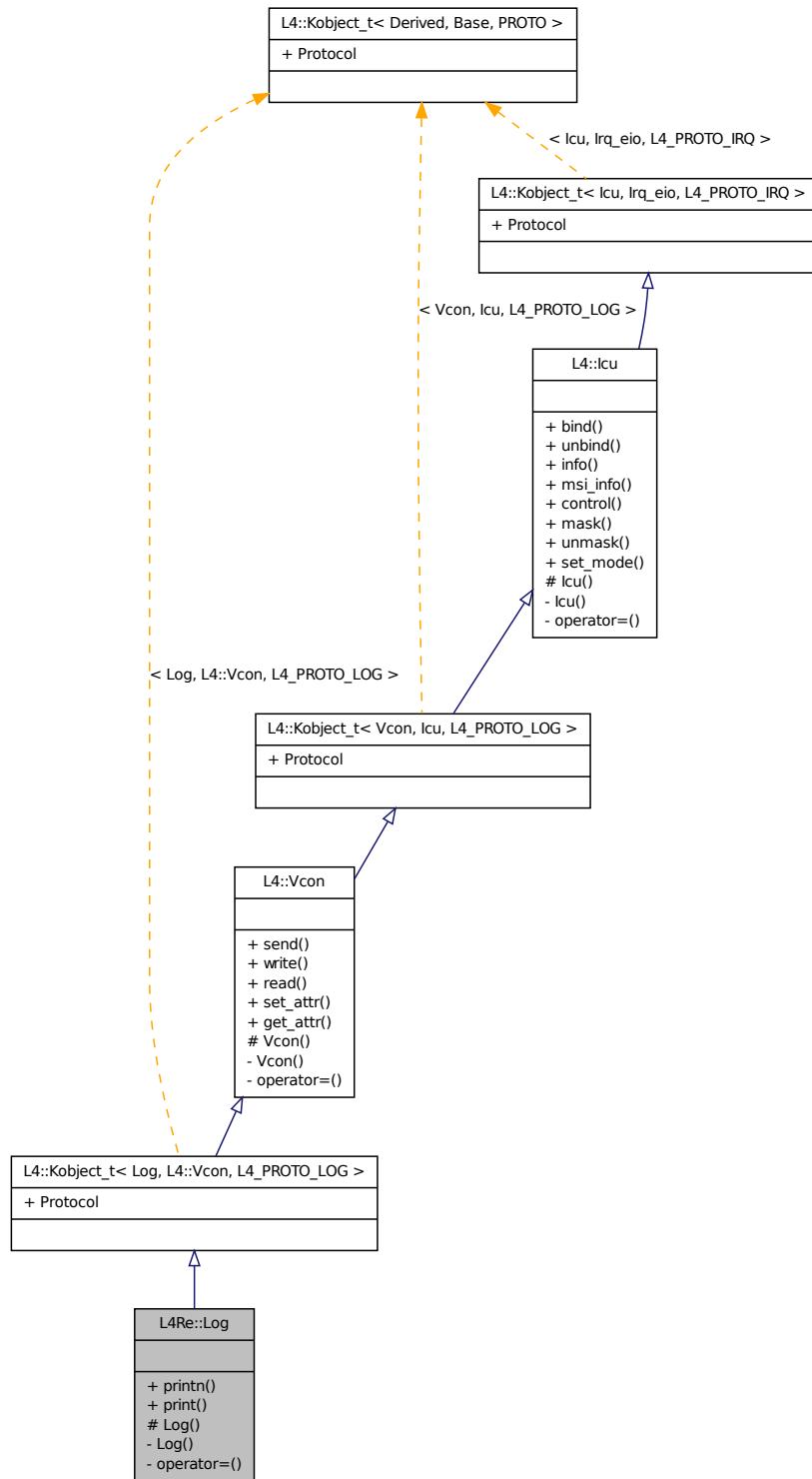
## 11.143 L4Re::Log Class Reference

[Log](#) interface class.

Inheritance diagram for L4Re::Log:



Collaboration diagram for L4Re::Log:



## Public Member Functions

- void `printn` (char const \**string*, int *len*) const throw ()  
*Print string with length len, NULL characters don't matter.*
- void `print` (char const \**string*) const throw ()  
*Print NULL-terminated string.*

### 11.143.1 Detailed Description

[Log](#) interface class.

Definition at line 44 of file [log](#).

### 11.143.2 Member Function Documentation

#### 11.143.2.1 void L4Re::Log::printn ( char const \* *string*, int *len* ) const throw ()

Print string with length len, NULL characters don't matter.

##### Parameters

- string* string to print  
*len* length of string

#### 11.143.2.2 void L4Re::Log::print ( char const \* *string* ) const throw ()

Print NULL-terminated string.

##### Parameters

- string* string to print

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

- [l4/re/log](#)

## 11.144 L4::Factory::Lstr Struct Reference

Special type to add a pascal string into the factory create stream.

### Data Fields

- char const \* `s`  
*The character buffer.*
- int `len`  
*The number of characters in the buffer.*

### 11.144.1 Detailed Description

Special type to add a pascal string into the factory create stream. This encapsulates a string that has an explicit length.

Definition at line 61 of file [factory](#).

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

- 14/sys/factory

## 11.145 cxx::Lt\_functor< Obj > Struct Template Reference

Generic comparator class that defaults to the less-than operator.

### 11.145.1 Detailed Description

```
template<typename Obj> struct cxx::Lt_functor< Obj >
```

Generic comparator class that defaults to the less-than operator.

Definition at line 29 of file [std\\_ops](#).

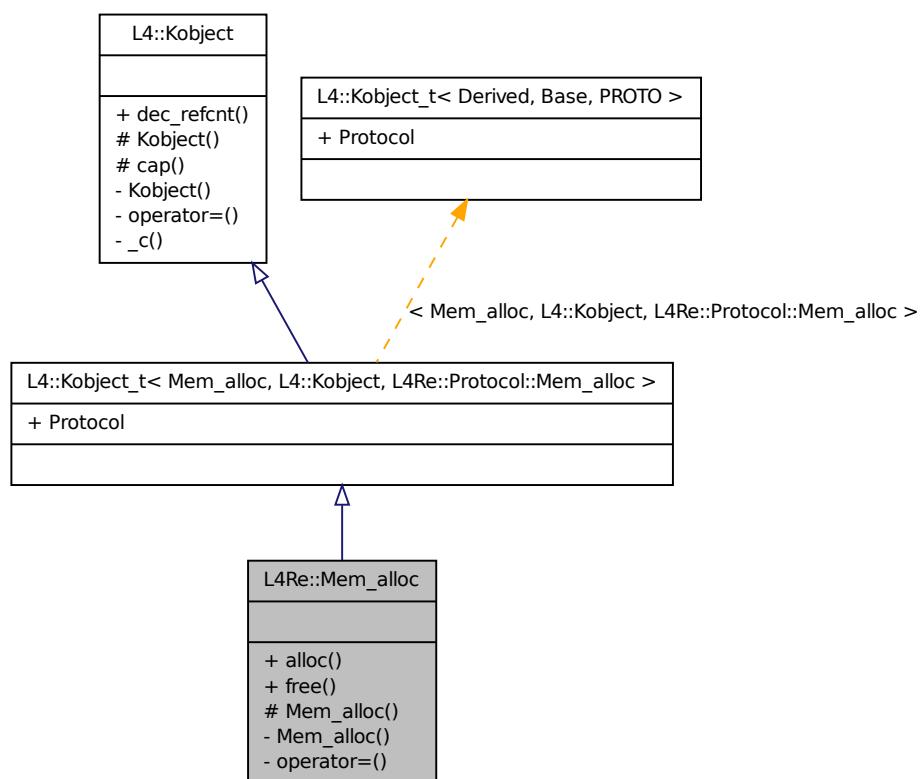
The documentation for this struct was generated from the following file:

- 14/cxx/std\_ops

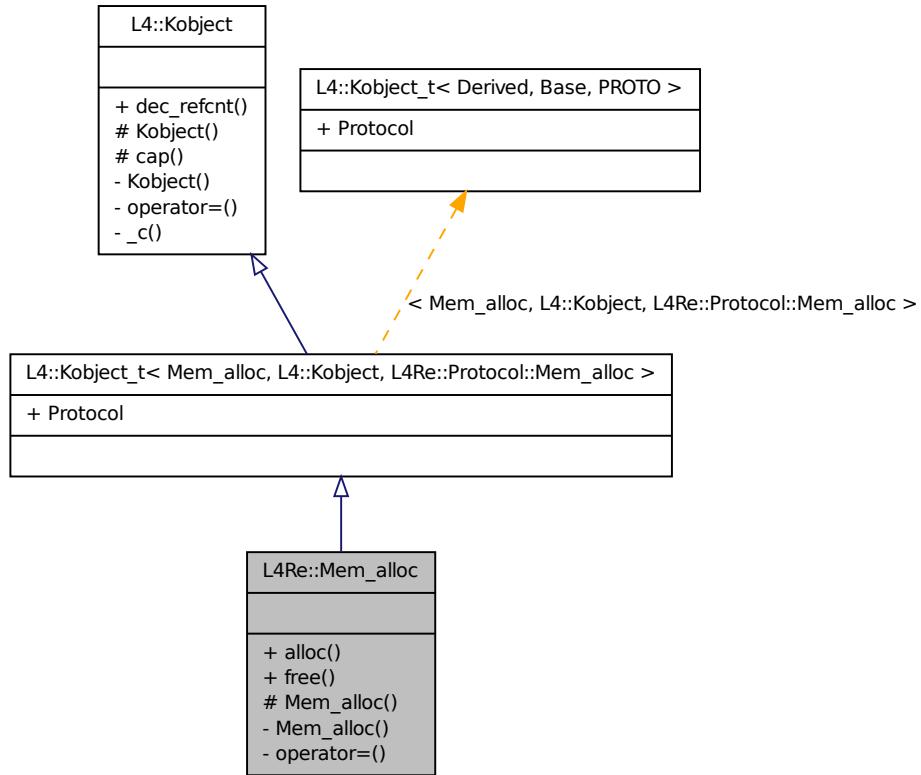
## 11.146 L4Re::Mem\_alloc Class Reference

Memory allocator.

Inheritance diagram for L4Re::Mem\_alloc:



Collaboration diagram for L4Re::Mem\_alloc:



## Public Types

- enum `Mem_alloc_flags` { `Continuous` = 0x01, `Pinned` = 0x02, `Super_pages` = 0x04 }

*Flags for the allocator.*

## Public Member Functions

- long `alloc` (unsigned long size, `L4::Cap< Dataspace >` mem, unsigned long flags=0) const throw ()
 

*Allocate anonymous memory.*
- long `free` (`L4::Cap< Dataspace >` mem) const throw ()
 

*Free data space.*

## 11.146.1 Detailed Description

Memory allocator. Memory-allocator interface, for more information see [Memory-allocator API](#).

Definition at line 69 of file [mem\\_alloc](#).

## 11.146.2 Member Enumeration Documentation

### 11.146.2.1 enum L4Re::Mem\_alloc::Mem\_alloc\_flags

Flags for the allocator.

#### Enumerator:

*Continuous* Allocate physically contiguous data space, if supported by the allocator.

*Pinned* Allocate pinned data space, if supported by the allocator.

*Super\_pages* Allocate super pages, if supported by the allocator.

Definition at line 78 of file [mem\\_alloc](#).

## 11.146.3 Member Function Documentation

### 11.146.3.1 long L4Re::Mem\_alloc::alloc ( unsigned long size, L4::Cap< Dataspace > mem, unsigned long flags = 0 ) const throw ()

Allocate anonymous memory.

#### Parameters

*size* Size to be requested in bytes (granularity is (super)pages and the size is rounded up to this granularity).

*mem* Object capability for the data space to be allocated.

*flags* Flags, see [Mem\\_alloc\\_flags](#), default none

#### Returns

0 on success, <0 on error

- -L4\_ENOMEM
- IPC errors

Definition at line 35 of file [mem\\_alloc\\_impl.h](#).

References [l4\\_error\(\)](#).

Here is the call graph for this function:



### 11.146.3.2 long L4Re::Mem\_alloc::free ( L4::Cap< Dataspace > mem ) const throw ()

Free data space.

#### Parameters

*mem* Data space that contains the memory.

#### Returns

0 on success, <0 on error

- -L4\_EINVAL
- IPC errors

Definition at line 45 of file [mem\\_alloc\\_impl.h](#).

The documentation for this class was generated from the following files:

- [l4/re/mem\\_alloc](#)
- [l4/re/impl/mem\\_alloc\\_impl.h](#)

## 11.147 L4::Kip::Mem\_desc Class Reference

Memory descriptors stored in the kernel interface page.

#### Public Types

- enum [Mem\\_type](#)

*Memory types.*

#### Public Member Functions

- [Mem\\_desc](#) (unsigned long start, unsigned long end, [Mem\\_type](#) t, unsigned char st=0, bool virt=false)  
throw ()

*Initialize memory descriptor.*

- `unsigned long start () const throw ()`  
*Return start address of memory descriptor.*
- `unsigned long end () const throw ()`  
*Return end address of memory descriptor.*
- `unsigned long size () const throw ()`  
*Return size of region described by the memory descriptor.*
- `Mem_type type () const throw ()`  
*Return type of the memory descriptor.*
- `unsigned char sub_type () const throw ()`  
*Return sub-type of the memory descriptor.*
- `unsigned is_virtual () const throw ()`  
*Return whether the memory descriptor describes a virtual or physical region.*
- `void set (unsigned long start, unsigned long end, Mem_type t, unsigned char st=0, bool virt=false) throw ()`  
*Set values of a memory descriptor.*

## Static Public Member Functions

- `static Mem_desc * first (void *kip) throw ()`  
*Get first memory descriptor.*
- `static unsigned long count (void const *kip) throw ()`  
*Return number of memory descriptors stored in the kernel info page.*
- `static void count (void *kip, unsigned count) throw ()`  
*Set number of memory descriptors.*

### 11.147.1 Detailed Description

Memory descriptors stored in the kernel interface page. #include <l4/sys/kip>  
 Definition at line 51 of file [kip](#).

### 11.147.2 Constructor & Destructor Documentation

#### 11.147.2.1 L4::Kip::Mem\_desc::Mem\_desc ( `unsigned long start, unsigned long end, Mem_type t, unsigned char st = 0, bool virt = false` ) `throw () [inline]`

Initialize memory descriptor.

**Parameters**

*start* Start address  
*end* End address  
*t* Memory type  
*st* Memory subtype, defaults to 0  
*virt* True for virtual memory, false for physical memory, defaults to physical

Definition at line 125 of file [kip](#).

### 11.147.3 Member Function Documentation

**11.147.3.1 static Mem\_desc\* L4::Kip::Mem\_desc::first ( void \* *kip* ) throw () [inline, static]**

Get first memory descriptor.

**Parameters**

*kip* Pointer to the kernel info page

**Returns**

First memory descriptor stored in the kernel info page

Definition at line 83 of file [kip](#).

**11.147.3.2 static unsigned long L4::Kip::Mem\_desc::count ( void const \* *kip* ) throw () [inline, static]**

Return number of memory descriptors stored in the kernel info page.

**Parameters**

*kip* Pointer to the kernel info page

**Returns**

Number of memory descriptors in the kernel info page.

Definition at line 100 of file [kip](#).

**11.147.3.3 static void L4::Kip::Mem\_desc::count ( void \* *kip*, unsigned *count* ) throw () [inline, static]**

Set number of memory descriptors.

**Parameters**

*kip* Pointer to the kernel info page  
*count* Number of memory descriptors

Definition at line 110 of file [kip](#).

**11.147.3.4 unsigned long L4::Kip::Mem\_desc::start ( ) const throw () [inline]**

Return start address of memory descriptor.

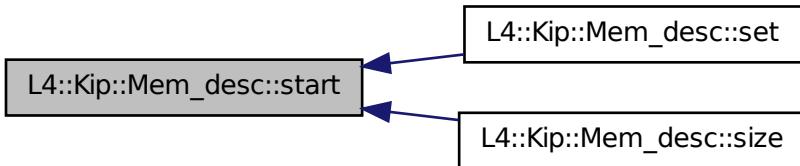
**Returns**

Start address of memory descriptor

Definition at line 135 of file [kip](#).

Referenced by [set\(\)](#), and [size\(\)](#).

Here is the caller graph for this function:

**11.147.3.5 unsigned long L4::Kip::Mem\_desc::end ( ) const throw () [inline]**

Return end address of memory descriptor.

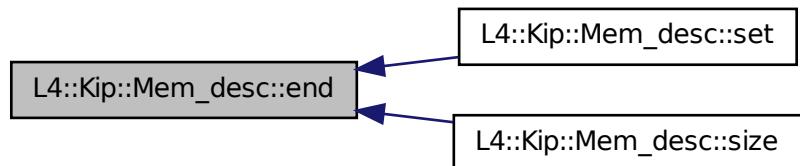
**Returns**

End address of memory descriptor

Definition at line 141 of file [kip](#).

Referenced by [set\(\)](#), and [size\(\)](#).

Here is the caller graph for this function:



**11.147.3.6 unsigned long L4::Kip::Mem\_desc::size ( ) const throw () [inline]**

Return size of region described by the memory descriptor.

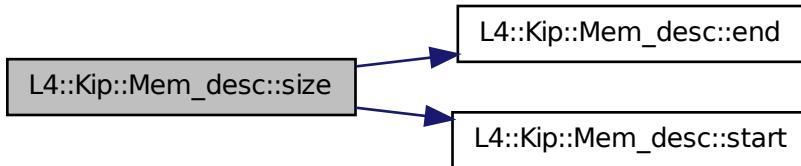
**Returns**

Size of the region described by the memory descriptor

Definition at line 147 of file [kip](#).

References [end\(\)](#), and [start\(\)](#).

Here is the call graph for this function:

**11.147.3.7 Mem\_type L4::Kip::Mem\_desc::type ( ) const throw () [inline]**

Return type of the memory descriptor.

**Returns**

Type of the memory descriptor

Definition at line 153 of file [kip](#).

**11.147.3.8 unsigned char L4::Kip::Mem\_desc::sub\_type ( ) const throw () [inline]**

Return sub-type of the memory descriptor.

**Returns**

Sub-type of the memory descriptor

Definition at line 159 of file [kip](#).

**11.147.3.9 unsigned L4::Kip::Mem\_desc::is\_virtual ( ) const throw () [inline]**

Return whether the memory descriptor describes a virtual or physical region.

**Returns**

True for virtual region, false for physical region.

Definition at line 166 of file [kip](#).

**11.147.3.10 void L4::Kip::Mem\_desc::set ( unsigned long start, unsigned long end, Mem\_type t, unsigned char st = 0, bool virt = false ) throw () [inline]**

Set values of a memory descriptor.

**Parameters**

*start* Start address

*end* End address

*t* Memory type

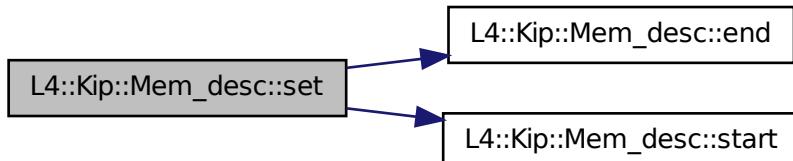
*st* Sub-type, defaults to 0

*virt* Virtual or physical memory region, defaults to physical

Definition at line 176 of file [kip](#).

References [end\(\)](#), and [start\(\)](#).

Here is the call graph for this function:



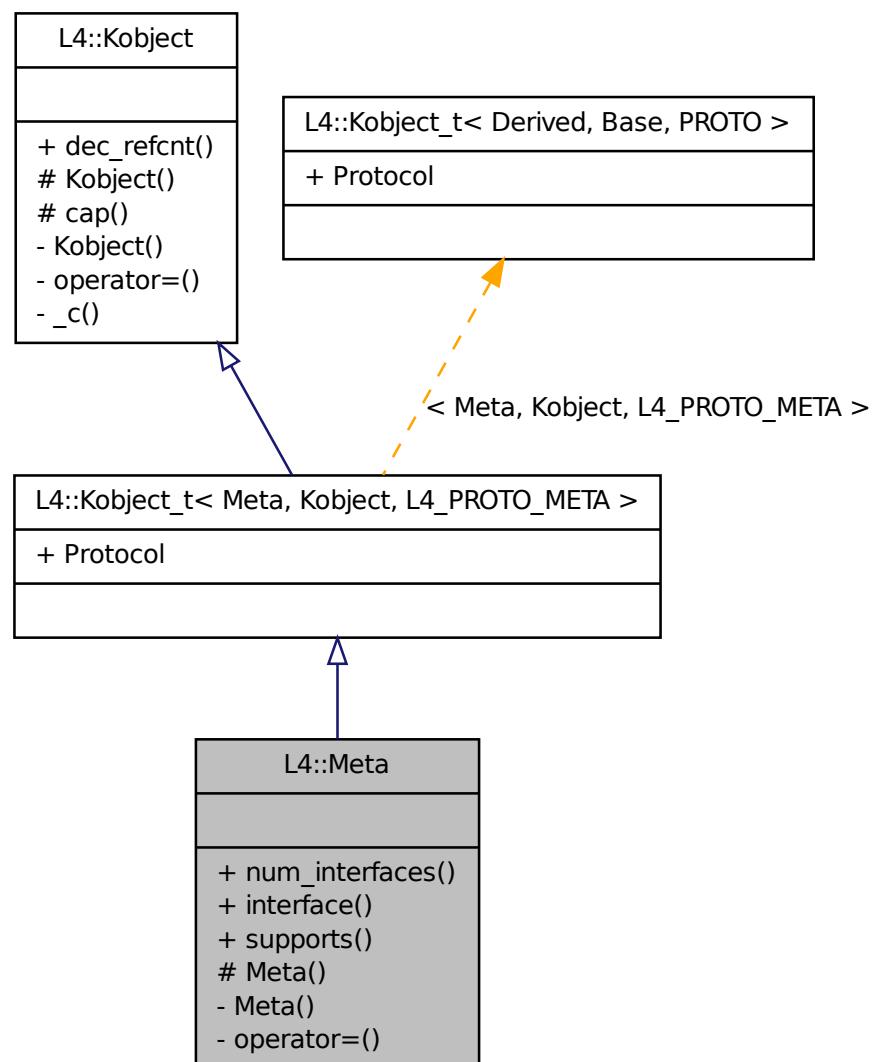
The documentation for this class was generated from the following file:

- [14/sys/kip](#)

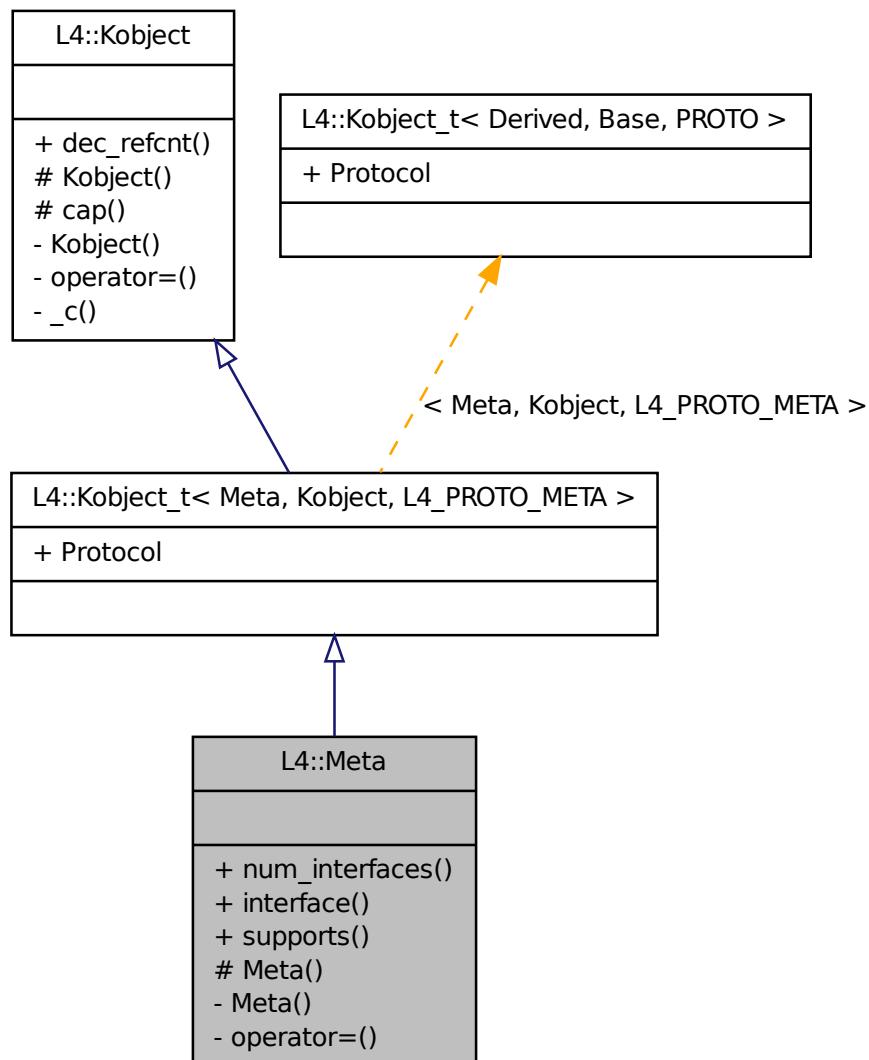
## 11.148 L4::Meta Class Reference

[Meta](#) interface that shall be implemented by each [L4Re](#) object and gives access to the dynamic type information for [L4Re](#) objects.

Inheritance diagram for L4::Meta:



Collaboration diagram for L4::Meta:



## Public Member Functions

- `l4_msgtag_t num_interfaces (l4_utcb_t *utcb=l4_utcb()) throw ()`  
*Get the number of interfaces implemented by this object.*
- `l4_msgtag_t interface (int idx, l4_utcb_t *u=l4_utcb()) throw ()`  
*Get the protocol number that must be used for the interface with the index idx.*
- `l4_msgtag_t supports (long protocol, l4_utcb_t *u=l4_utcb()) throw ()`

*Figure out if the object supports the given protocol (number).*

### 11.148.1 Detailed Description

Meta interface that shall be implemented by each L4Re object and gives access to the dynamic type information for L4Re objects.

Definition at line 41 of file [meta](#).

### 11.148.2 Member Function Documentation

#### 11.148.2.1 l4\_mshtag\_t L4::Meta::num\_interfaces ( l4\_utcb\_t \* *utcb* = **l4\_utcb()** ) throw () [inline]

Get the number of interfaces implemented by this object.

##### Parameters

*utcb* is the utcb to use for sending the message.

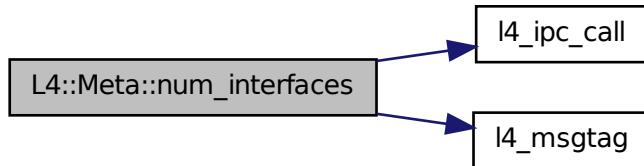
##### Returns

The message tag for the operation, the label ([l4\\_mshtag\\_t::label\(\)](#)) is set to the number of interfaces if successful, or to -error when an error occurred.

Definition at line 89 of file [meta](#).

References [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_mshtag\(\)](#), and [l4\\_msg\\_regs\\_t::mr](#).

Here is the call graph for this function:



#### 11.148.2.2 l4\_mshtag\_t L4::Meta::interface ( int *idx*, l4\_utcb\_t \* *u* = **l4\_utcb()** ) throw () [inline]

Get the protocol number that must be used for the interface with the index *idx*.

##### Parameters

*idx* is the index of the interface to get the protocol number for. *idx* must be  $\geq 0$  and  $<$  the return value of [num\\_interfaces\(\)](#).

*utcb* is the utcb to use for sending the message.

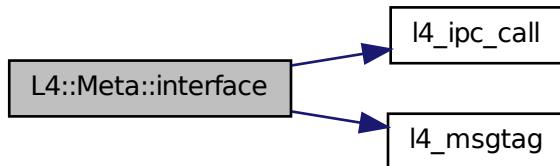
### Returns

The message tag for the operation, the label (`l4_mshtag_t::label()`) is set to the protocol number of interface *idx*.

Definition at line 98 of file `meta`.

References `l4_ipc_call()`, `L4_IPC_NEVER`, `l4_mshtag()`, and `l4_msg_regs_t::mr`.

Here is the call graph for this function:



### 11.148.2.3 `l4_mshtag_t L4::Meta::supports( long protocol, l4_utcb_t * u = 14_utcb() ) throw()` [inline]

Figure out if the object supports the given *protocol* (number).

### Parameters

*protocol* is the protocol number to check for.

*utcb* is the utcb to use for sending the message.

### Returns

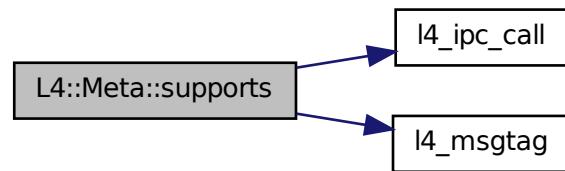
The message tag for the operation, the label (`l4_mshtag_t::label()`) is set to 1 if *protocol* is supported to 0 if not.

This method is intended to be used for statically assigned protocol numbers.

Definition at line 108 of file `meta`.

References `l4_ipc_call()`, `L4_IPC_NEVER`, `l4_mshtag()`, and `l4_msg_regs_t::mr`.

Here is the call graph for this function:



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

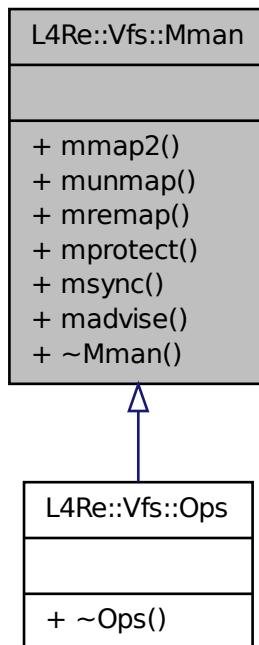
- l4/sys/meta

## 11.149 L4Re::Vfs::Mman Class Reference

Interface for the POSIX memory management.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Mman:



## Public Member Functions

- virtual int `mmap2` (void \*start, size\_t len, int prot, int flags, int fd, off\_t offset, void \*\*ptr)=0 throw ()
 

*Backend for the mmap2 system call.*
- virtual int `munmap` (void \*start, size\_t len)=0 throw ()
 

*Backend for the munmap system call.*
- virtual int `mremap` (void \*old, size\_t old\_sz, size\_t new\_sz, int flags, void \*\*new\_adr)=0 throw ()
 

*Backend for the mremap system call.*
- virtual int `mprotect` (const void \*a, size\_t sz, int prot)=0 throw ()
 

*Backend for the mprotect system call.*
- virtual int `msync` (void \*addr, size\_t len, int flags)=0 throw ()
 

*Backend for the msync system call.*
- virtual int `madvise` (void \*addr, size\_t len, int advice)=0 throw ()
 

*Backend for the madvice system call.*

### 11.149.1 Detailed Description

Interface for the POSIX memory management.

#### Note

This interface exists usually as a singleton as superclass of [L4Re::Vfs::Ops](#).

An implementation for this interface is in [l4/l4re\\_vfs/impl/vfs\\_impl.h](#) and used by the l4re\_vfs library or by the VFS implementation in lsdo.

Definition at line [757](#) of file [vfs.h](#).

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

- [l4/l4re\\_vfs/vfs.h](#)

## 11.150 L4::Thread::Modify\_senders Class Reference

Wrapper class for modifying senders.

### Public Member Functions

- `int add (l4_umword_t match_mask, l4_umword_t match, l4_umword_t del_bits, l4_umword_t add_bits) throw ()`

*Add a rule.*

### 11.150.1 Detailed Description

Wrapper class for modifying senders. Use the `add()` function to add modification rules, and use `modify_senders()` to commit. Do not use the UTCB inbetween as it is used by `add()` and `modify_senders()`.

Definition at line [220](#) of file [thread](#).

### 11.150.2 Member Function Documentation

#### 11.150.2.1 `int L4::Thread::Modify_senders::add ( l4_umword_t match_mask, l4_umword_t match, l4_umword_t del_bits, l4_umword_t add_bits ) throw () [inline]`

Add a rule.

#### Parameters

`match_mask` Bitmask of bits to match the label.

`match` Bitmask that must be equal to the label after applying `match_mask`.

`del_bits` Bits to be deleted from the label.

`add_bits` Bits to be added to the label.

#### Returns

0 on sucess, <0 on error

Only the first match is applied.

#### See also

[l4\\_thread\\_modify\\_sender\\_add\(\)](#)

Definition at line 249 of file [thread](#).

References [L4\\_UTCB\\_GENERIC\\_DATA\\_SIZE](#), and [l4\\_msg\\_regs\\_t::mr](#).

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

- l4/sys/thread

## 11.151 L4::Ipc::Msg\_ptr< T > Class Template Reference

Pointer to an element of type T in an [Ipc::Istream](#).

### Public Member Functions

- [Msg\\_ptr \(T \\*&p\)](#)  
*Create a [Msg\\_ptr](#) object that set pointer p to point into the message buffer.*

#### 11.151.1 Detailed Description

**template<typename T> class L4::Ipc::Msg\_ptr< T >**

Pointer to an element of type T in an [Ipc::Istream](#). This wrapper can be used to extract an element of type T from an [Ipc::Istream](#), whereas the data is not copied out, but a pointer into the message buffer itself is returned. With this mechanism it is possible to avoid an extra copy of large data structures from a received IPC message, instead the returned pointer gives direct access to the data in the message.

See [msg\\_ptr\(\)](#).

Definition at line 169 of file [ipc\\_stream](#).

#### 11.151.2 Constructor & Destructor Documentation

**11.151.2.1 template<typename T> L4::Ipc::Msg\_ptr< T >::Msg\_ptr ( T \*& p ) [inline, explicit]**

Create a [Msg\\_ptr](#) object that set pointer p to point into the message buffer.

#### Parameters

- p* The pointer that is adjusted to point into the message buffer.

Definition at line 179 of file [ipc\\_stream](#).

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

- l4/cxx/ipc\_stream

## 11.152 L4Re::Util::Names::Name Class Reference

[Name](#) class.

Inherits [cxx::String](#).

### 11.152.1 Detailed Description

[Name](#) class.

Definition at line [36](#) of file [name\\_space\\_svr](#).

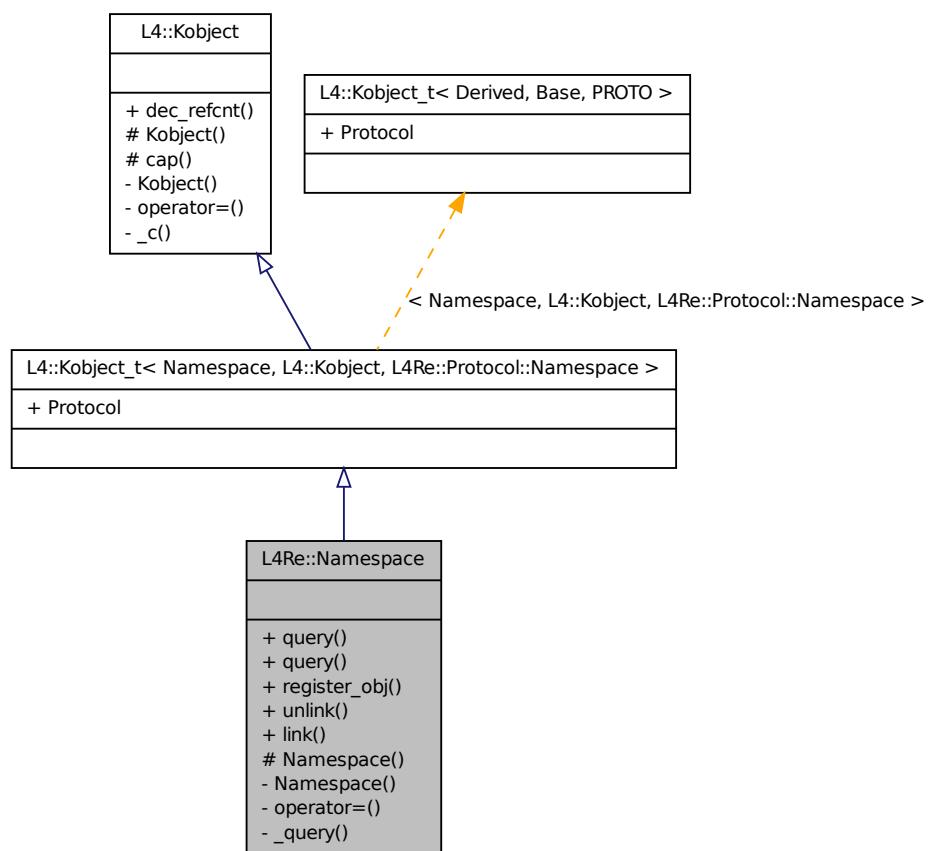
The documentation for this class was generated from the following file:

- [l4/re/util/name\\_space\\_svr](#)

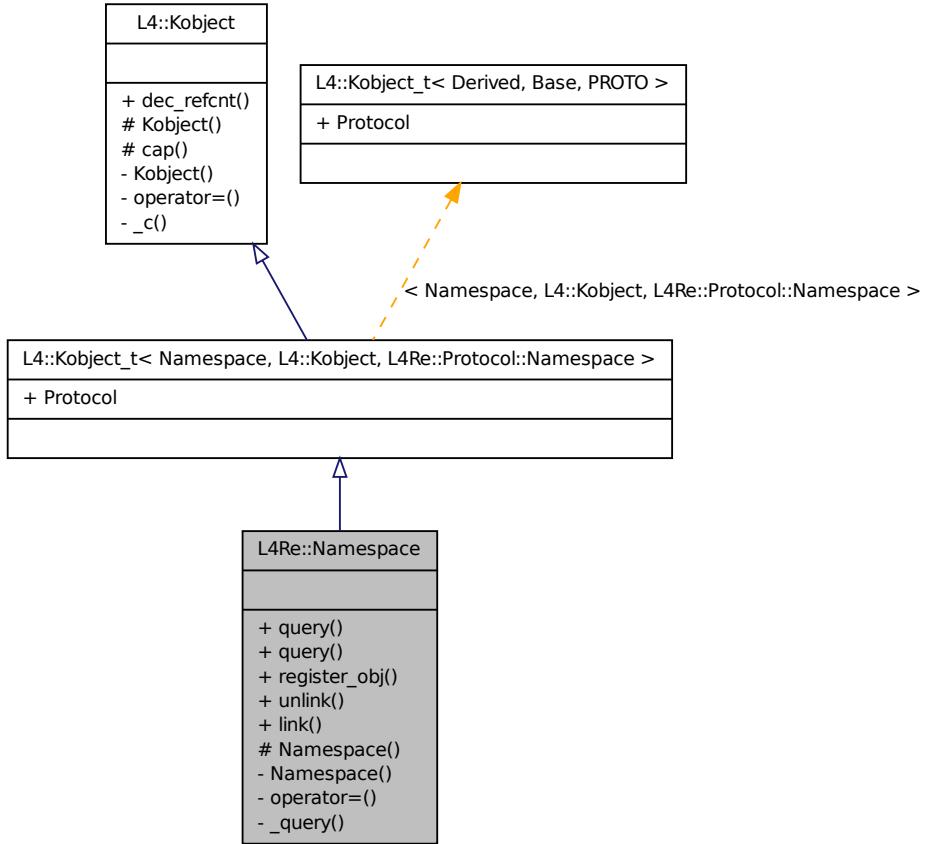
## 11.153 L4Re::Namespace Class Reference

Name-space interface.

Inheritance diagram for L4Re::Namespace:



Collaboration diagram for L4Re::Namespace:



## Public Types

- enum `Register_flags` { `Ro` = `L4_CAP_FPAGE_RO`, `Rw` = `L4_CAP_FPAGE_RW` , `Strong` = `L4_-CAP_FPAGE_S` }

*Flags for registering name spaces.*

## Public Member Functions

- long `query` (char const \*name, `L4::Cap< void >` const &cap, int timeout=`To_default`, `l4_umword_t` \*local\_id=0, bool iterate=true) const throw ()
 

*Query a name.*
- long `query` (char const \*name, unsigned len, `L4::Cap< void >` const &cap, int timeout=`To_default`, `l4_umword_t` \*local\_id=0, bool iterate=true) const throw ()
 

*Query a name.*

- long `register_obj` (char const \*name, L4::Cap< void > const &obj, unsigned flags=Rw) const throw ()

*Register an object with a name.*

### 11.153.1 Detailed Description

Name-space interface. All name space objects must provide this interface. However, it is not mandatory that a name space object allows to register new capabilities.

The name lookup is done iteratively, this means the hierarchical names are resolved component wise by the client itself.

Definition at line 57 of file [namespace](#).

### 11.153.2 Member Enumeration Documentation

#### 11.153.2.1 enum L4Re::Namespace::Register\_flags

Flags for registering name spaces.

**Enumerator:**

***Ro*** Read-only.

***Rw*** Read-write.

***Strong*** Strong.

Definition at line 66 of file [namespace](#).

### 11.153.3 Member Function Documentation

#### 11.153.3.1 long L4Re::Namespace::query ( char const \* *name*, L4::Cap< void > const & *cap*, int *timeout* = *To\_default*, l4\_umword\_t \* *local\_id* = 0, bool *iterate* = *true* ) const throw ()

Query a name.

**Parameters**

*name* String to query

*cap* Capability slot to put object into.

*timeout* Timeout of query in milliseconds.

**Return values**

*local\_id* Local id.

**Returns**

<0 on failure, see [l4\\_error\\_code\\_t](#).

- [-L4\\_ENOENT](#)

- IPC errors == 0 if name could be fully resolved > 0 if name could not be fully resolved

Definition at line 109 of file [namespace\\_impl.h](#).

**11.153.3.2 long L4Re::Namespace::query ( char const \* *name*, unsigned *len*, L4::Cap< void > const & *cap*, int *timeout* = *To\_default*, l4\_umword\_t \* *local\_id* = 0, bool *iterate* = *true* ) const throw ()**

Query a name.

#### Parameters

*name* String to query  
*len* Length of the string to query.  
*cap* Capability slot to put object into.  
*timeout* Timeout of query in milliseconds.

#### Return values

*local\_id* Local id.

#### Returns

<0 on failure, see [l4\\_error\\_code\\_t](#).  
 • -L4\_ENOENT  
 • IPC errors == 0 if name could be fully resolved > 0 if name could not be fully resolved

Definition at line 75 of file [namespace\\_impl.h](#).

**11.153.3.3 long L4Re::Namespace::register\_obj ( char const \* *name*, L4::Cap< void > const & *obj*, unsigned *flags* = *Rw* ) const throw ()**

Register an object with a name.

#### Parameters

*name* String to register.  
*obj* Object to register.  
*flags* Flags to use, see [Register\\_flags](#), default is rw.

#### Returns

0 on success, <0 on failure, see [l4\\_error\\_code\\_t](#).  
 • -L4\_EEXIST  
 • -L4\_EPERM  
 • -L4\_ENOMEM  
 • -L4\_EINVAL  
 • IPC errors

Definition at line 115 of file [namespace\\_impl.h](#).

The documentation for this class was generated from the following files:

- [l4/re/namespace](#)
- [l4/re/impl/namespace\\_impl.h](#)

## 11.154 cxx::New\_allocator< \_Type > Class Template Reference

Standard allocator based on `operator new ()`.

### 11.154.1 Detailed Description

```
template<typename _Type> class cxx::New_allocator< _Type >
```

Standard allocator based on `operator new ()`. This allocator is the default allocator used for the *cxx Containers*, such as [cxx::Avl\\_set](#) and [cxx::Avl\\_map](#), to allocate the internal data structures.

Definition at line 60 of file [std\\_alloc](#).

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

- 14/cxx/std\_alloc

## 11.155 L4::Factory::Nil Struct Reference

Special type to add a void argument into the factory create stream.

### 11.155.1 Detailed Description

Special type to add a void argument into the factory create stream.

Definition at line 53 of file [factory](#).

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

- 14/sys/factory

## 11.156 cxx::Avl\_set< Item, Compare, Alloc >::Node Class Reference

A smart pointer to a tree item.

### Public Member Functions

- `Node()`  
*Default construction for NIL pointer.*
- `Node & operator= (Node const &o)`  
*Default assignment.*
- `Item const & operator* ()`  
*Dereference the pointer.*
- `Item const * operator-> ()`

*Dereferenced member access.*

- `bool valid () const`  
*Validity check.*
- `operator Item const * ()`  
*Cast to a real item pointer.*

### 11.156.1 Detailed Description

`template<typename Item, class Compare = Lt_functor<Item>, template< typename A > class Alloc = New_allocator> class cxx::Avl_set< Item, Compare, Alloc >::Node`

A smart pointer to a tree item.

Definition at line 148 of file [avl\\_set](#).

### 11.156.2 Member Function Documentation

**11.156.2.1** `template<typename Item, class Compare = Lt_functor<Item>, template< typename A > class Alloc = New_allocator> bool cxx::Avl_set< Item, Compare, Alloc >::Node::valid ( ) const [inline]`

Validity check.

#### Returns

false if the pointer is NIL, true if valid.

Definition at line 172 of file [avl\\_set](#).

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

- l4/cxx/avl\_set

### 11.157 cxx::Nothrow Class Reference

Helper type to distinguish the oeprator new version that does not throw exceptions.

#### 11.157.1 Detailed Description

Helper type to distinguish the oeprator new version that does not throw exceptions.

Definition at line 30 of file [std\\_alloc](#).

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

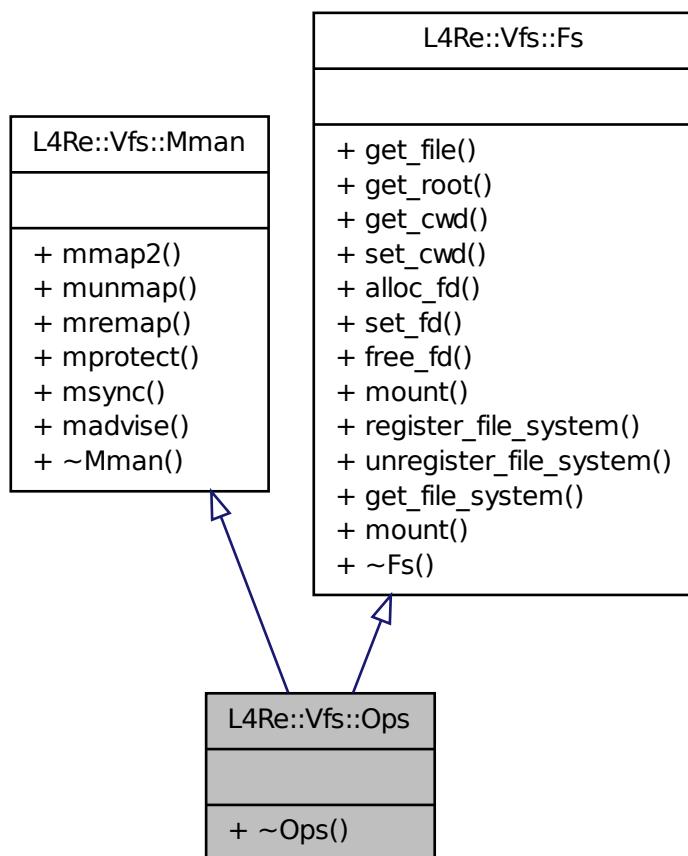
- l4/cxx/std\_alloc

## 11.158 L4Re::Vfs::Ops Class Reference

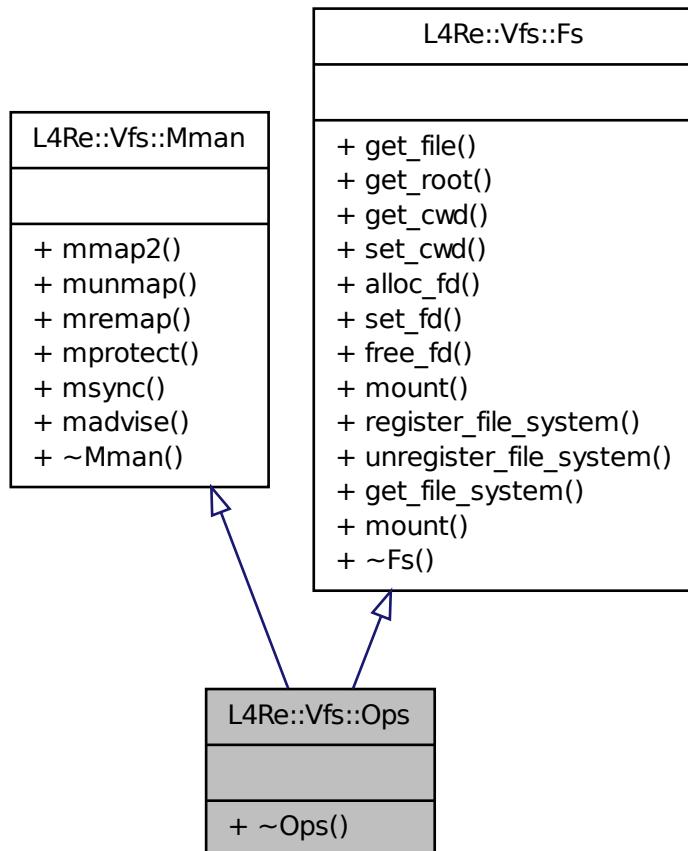
Interface for the POSIX backends for an application.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Ops:



Collaboration diagram for L4Re::Vfs::Ops:



### 11.158.1 Detailed Description

Interface for the POSIX backends for an application.

#### Note

There usually exists a single instance of this interface available via L4Re::Vfs::vfs\_ops that is used for all kinds of C-Library functions.

Definition at line 988 of file [vfs.h](#).

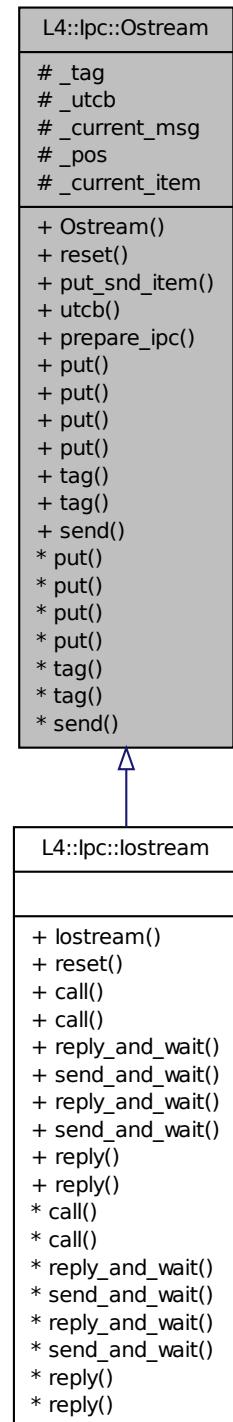
The documentation for this class was generated from the following file:

- l4/l4re\_vfs/vfs.h

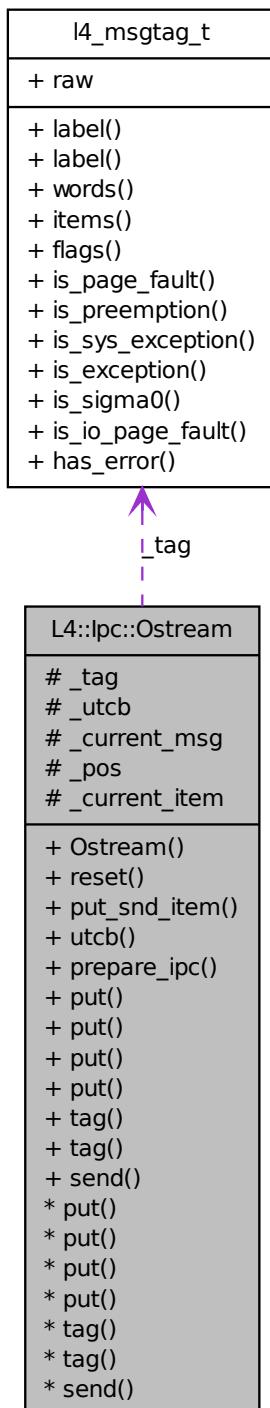
## **11.159 L4::Ipc::Ostream Class Reference**

Output stream for IPC marshalling.

Inheritance diagram for L4::Ipc::Ostream:



Collaboration diagram for L4::Ipc::Ostream:



## Public Member Functions

- **Ostream (l4\_utcb\_t \*utcb)**  
*Create an IPC output stream using the given message buffer msg.*
- **void reset ()**  
*Reset the stream to empty, same state as a newly created stream.*
- **l4\_utcb\_t \* utcb () const**  
*Return utcb pointer.*

### Get/Put functions.

*These functions are basically used to implement the insertion operators (<<) and should not be called directly.*

*See [IPC stream operators](#).*

- template<typename T >  
**void put (T \*buf, unsigned long size)**  
*Put an array with size elements of type T into the stream.*
- template<typename T >  
**bool put (T const &v)**  
*Insert an element of type T into the stream.*
- **int put (Varg const &va)**
- template<typename T >  
**int put (Varg\_t< T > const &va)**
- **l4\_mshtag\_t tag () const**  
*Extract the L4 message tag from the stream.*
- **l4\_mshtag\_t & tag ()**  
*Extract a reference to the L4 message tag from the stream.*

### IPC operations.

- **l4\_mshtag\_t send (l4\_cap\_idx\_t dst, long proto=0, unsigned flags=0)**  
*Send the message via IPC to the given receiver.*

## 11.159.1 Detailed Description

Output stream for IPC mar shalling. [Ipc::Ostream](#) is part of the dynamic IPC mar shalling infrastructure, as well as [Ipc::Istream](#) and [Ipc::Iostream](#).

[Ipc::Ostream](#) is an output stream supporting insertion of values into an IPC message buffer. A IPC message can be mar shalled using the usual insertion operator <<, see [IPC stream operators](#).

There exist some special wrapper classes to insert arrays (see [Ipc::Buf\\_cp\\_out](#)) and indirect strings (see [Msg\\_out\\_buffer](#) and [Msg\\_io\\_buffer](#)).

Definition at line 840 of file [ipc\\_stream](#).

## 11.159.2 Member Function Documentation

### 11.159.2.1 template<typename T > void L4::Ipc::Ostream::put ( T \* buf, unsigned long size ) [inline]

Put an array with *size* elements of type *T* into the stream.

#### Parameters

*buf* A pointer to the array to insert into the buffer.

*size* The number of elements in the array.

Definition at line 878 of file [ipc\\_stream](#).

### 11.159.2.2 template<typename T > bool L4::Ipc::Ostream::put ( T const & v ) [inline]

Insert an element of type *T* into the stream.

#### Parameters

*v* The element to insert.

Definition at line 894 of file [ipc\\_stream](#).

### 11.159.2.3 l4\_mshtag\_t L4::Ipc::Ostream::tag ( ) const [inline]

Extract the [L4](#) message tag from the stream.

#### Returns

the extracted [L4](#) message tag.

Definition at line 921 of file [ipc\\_stream](#).

Referenced by [send\(\)](#).

Here is the caller graph for this function:



### 11.159.2.4 l4\_mshtag\_t& L4::Ipc::Ostream::tag ( ) [inline]

Extract a reference to the [L4](#) message tag from the stream.

**Returns**

A reference to the [L4](#) message tag.

Definition at line [927](#) of file [ipc\\_stream](#).

**11.159.2.5 l4\_mshtag\_t L4::Ipc::Ostream::send ( l4\_cap\_idx\_t dst, long proto = 0, unsigned flags = 0 ) [inline]**

Send the message via IPC to the given receiver.

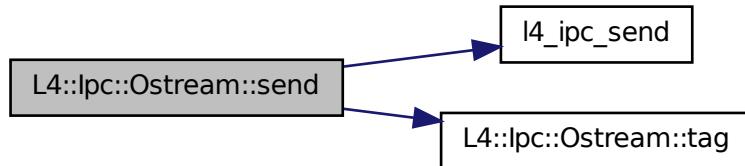
**Parameters**

*dst* The destination for the message.

Definition at line [1147](#) of file [ipc\\_stream](#).

References [L4\\_IPC\\_NEVER](#), [l4\\_ipc\\_send\(\)](#), [L4\\_MSGTAG\\_FLAGS](#), and [tag\(\)](#).

Here is the call graph for this function:



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

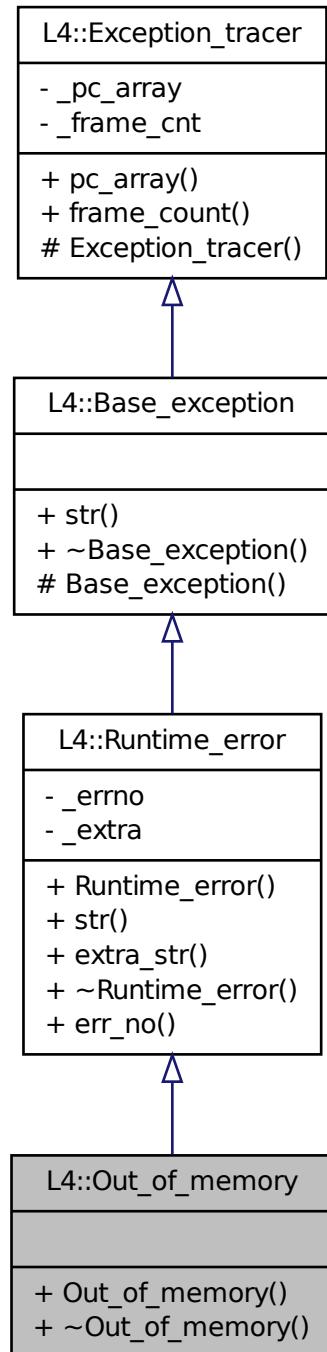
- [l4/cxx/ipc\\_stream](#)

## 11.160 L4::Out\_of\_memory Class Reference

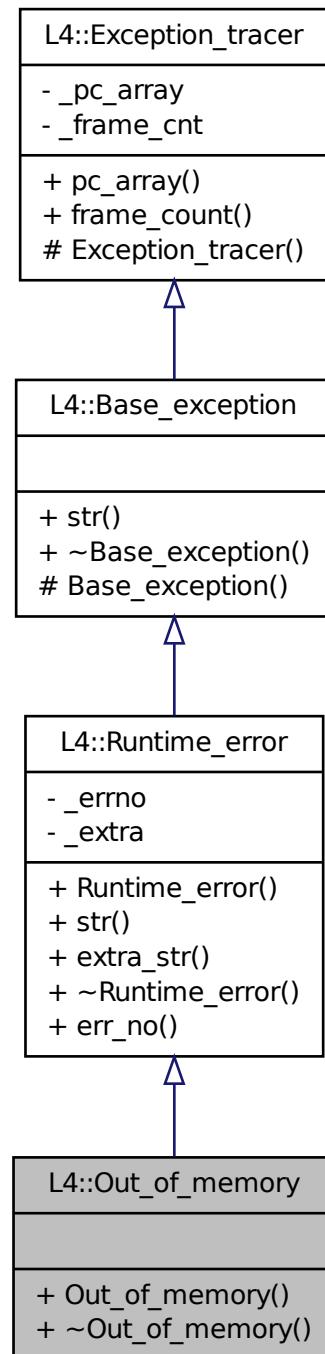
Exception signalling insufficient memory.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Out\_of\_memory:



Collaboration diagram for L4::Out\_of\_memory:



## Public Member Functions

- **Out\_of\_memory** (char const \*extra="") throw ()  
*Create an out-of-memory exception.*
- **~Out\_of\_memory** () throw ()  
*Destruction.*

### 11.160.1 Detailed Description

Exception signalling insufficient memory.

Definition at line 162 of file [exceptions](#).

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

- 14/cxx/exceptions

## 11.161 cxx::Pair< First, Second > Struct Template Reference

[Pair](#) of two values.

## Public Types

- **typedef First First\_type**  
*Type of first value.*
- **typedef Second Second\_type**  
*Type of second value.*

## Public Member Functions

- **Pair** (First const &**first**, Second const &**second**)  
*Create a pair from the two values.*
- **Pair** ()  
*Default construction.*

## Data Fields

- First **first**  
*First value.*
- Second **second**  
*Second value.*

### 11.161.1 Detailed Description

**template<typename First, typename Second> struct cxx::Pair< First, Second >**

**Pair** of two values. Standard container for a pair of values.

#### Parameters

*First* Type of the first value.

*Second* Type of the second value.

Definition at line 36 of file [pair](#).

### 11.161.2 Constructor & Destructor Documentation

**11.161.2.1 template<typename First, typename Second> cxx::Pair< First, Second >::Pair ( First const & *first*, Second const & *second* ) [inline]**

Create a pair from the two values.

#### Parameters

*first* The first value.

*second* The second value.

Definition at line 53 of file [pair](#).

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

- 14/cxx/pair

## 11.162 cxx::Pair\_first\_compare< Cmp, Typ > Class Template Reference

Comparison functor for [Pair](#).

### Public Member Functions

- [Pair\\_first\\_compare](#) (Cmp const &cmp=Cmp())
 

*Construction.*
- bool [operator\(\)](#) (Typ const &l, Typ const &r) const
 

*Do the comparison based on the first value.*

### 11.162.1 Detailed Description

**template<typename Cmp, typename Typ> class cxx::Pair\_first\_compare< Cmp, Typ >**

Comparison functor for [Pair](#).

### Parameters

*Cmp* Comparison functor for the first value of the pair.

*Typ* The pair type.

This functor can be used to compare [Pair](#) values with respect to the first value.

Definition at line [74](#) of file [pair](#).

## 11.162.2 Constructor & Destructor Documentation

### 11.162.2.1 `template<typename Cmp , typename Typ > cxx::Pair_first_compare< Cmp, Typ >::Pair_first_compare ( Cmp const & cmp = Cmp () ) [inline]`

Construction.

### Parameters

*cmp* The comparison functor used for the first value.

Definition at line [84](#) of file [pair](#).

## 11.162.3 Member Function Documentation

### 11.162.3.1 `template<typename Cmp , typename Typ > bool cxx::Pair_first_compare< Cmp, Typ >::operator() ( Typ const & l, Typ const & r ) const [inline]`

Do the comaprison based on the first value.

### Parameters

*l* The lefthand value.

*r* The righthand value.

Definition at line [91](#) of file [pair](#).

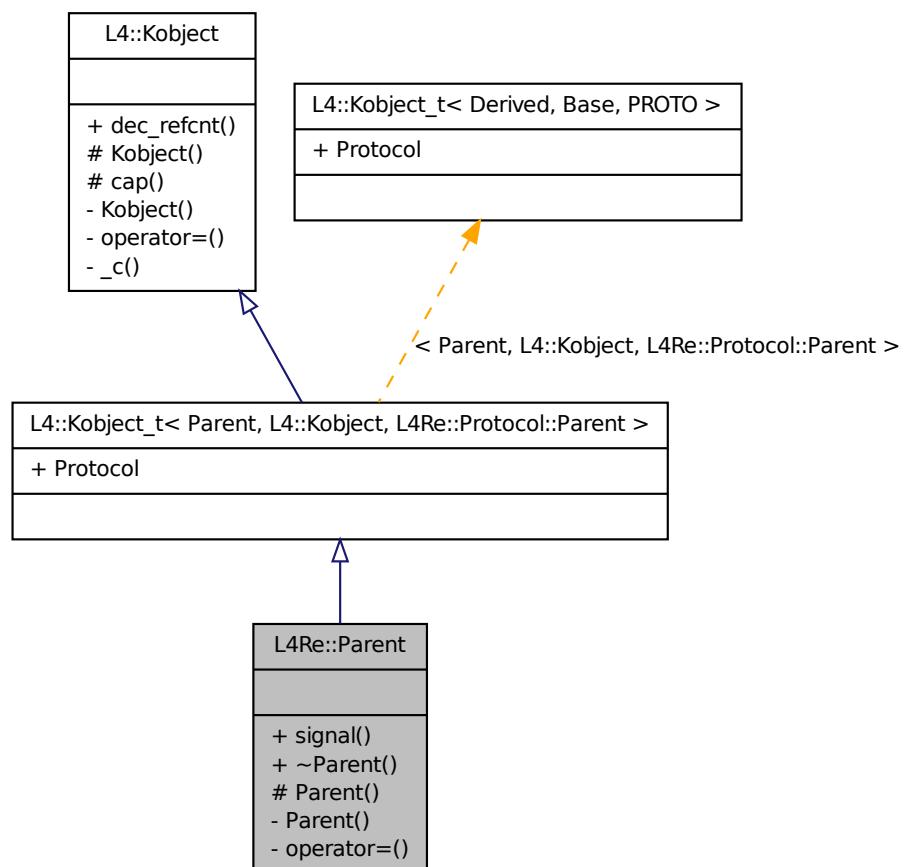
The documentation for this class was generated from the following file:

- [14/cxx/pair](#)

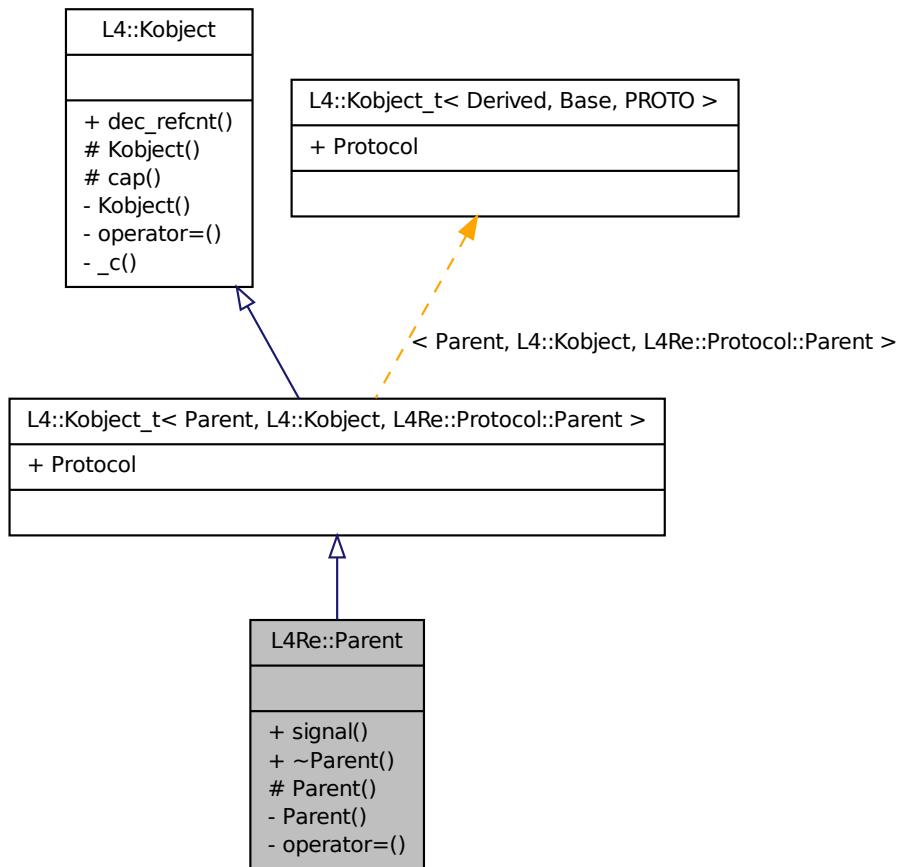
## 11.163 L4Re::Parent Class Reference

[Parent](#) interface.

Inheritance diagram for L4Re::Parent:



Collaboration diagram for L4Re::Parent:



## Public Member Functions

- long [signal](#) (unsigned long sig, unsigned long val) const throw ()

*Send a signal to the parent.*

### 11.163.1 Detailed Description

[Parent](#) interface.

#### See also

[Parent API](#) for more details about the purpose.

Definition at line [50](#) of file [parent](#).

### 11.163.2 Member Function Documentation

#### 11.163.2.1 long L4Re::Parent::signal ( `unsigned long sig, unsigned long val` ) const throw ()

Send a signal to the parent.

##### Parameters

*sig* Signal to send

*val* Value of the signal

##### Returns

0 on success, <0 on error

- [-L4\\_ENOREPLY](#)
- IPC errors

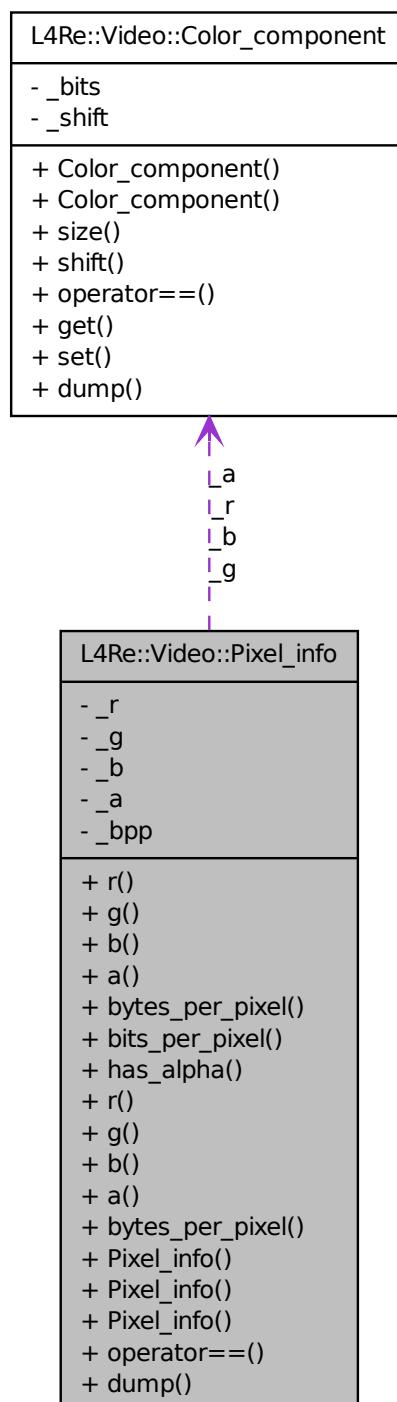
The documentation for this class was generated from the following file:

- [l4/re/parent](#)

### 11.164 L4Re::Video::Pixel\_info Class Reference

Pixel information.

Collaboration diagram for L4Re::Video::Pixel\_info:



## Public Member Functions

- `Color_component const & r () const`  
*Return the red color compoment of the pixel.*
- `Color_component const & g () const`  
*Return the green color compoment of the pixel.*
- `Color_component const & b () const`  
*Return the blue color compoment of the pixel.*
- `Color_component const & a () const`  
*Return the alpha color compoment of the pixel.*
- `unsigned char bytes_per_pixel () const`  
*Query size of pixel in bytes.*
- `unsigned char bits_per_pixel () const`  
*Number of bits of the pixel.*
- `bool has_alpha () const`  
*Return whether the pixel has an alpha channel.*
- `void r (Color_component const &c)`  
*Set the red color component of the pixel.*
- `void g (Color_component const &c)`  
*Set the green color component of the pixel.*
- `void b (Color_component const &c)`  
*Set the blue color component of the pixel.*
- `void a (Color_component const &c)`  
*Set the alpha color component of the pixel.*
- `void bytes_per_pixel (unsigned char bpp)`  
*Set the size of the pixel in bytes.*
- `Pixel_info ()`  
*Constructor.*
- `Pixel_info (unsigned char bpp, char r, char rs, char g, char gs, char b, char bs, char a=0, char as=0)`  
*Constructor.*
- `template<typename VBI > Pixel_info (VBI const *vbi)`  
*Convenience constructor.*
- `bool operator== (Pixel_info const &o) const`  
*Compare for complete equazality of the color sapce.*

- template<typename STREAM >  
STREAM & **dump** (STREAM &s) const

*Dump information on the pixel to a stream.*

## 11.164.1 Detailed Description

Pixel information. This class wraps the information on a pixel, such as the size and position of each color component in the pixel.

Definition at line 108 of file [colors](#).

## 11.164.2 Constructor & Destructor Documentation

### 11.164.2.1 L4Re::Video::Pixel\_info ( unsigned char *bpp*, char *r*, char *rs*, char *g*, char *gs*, char *b*, char *bs*, char *a* = 0, char *as* = 0 ) [inline]

Constructor.

#### Parameters

- bpp* Size of pixel in bytes.
- r* Red component size.
- rs* Red component shift.
- g* Green component size.
- gs* Green component shift.
- b* Blue component size.
- bs* Blue component shift.
- a* Alpha component size, defaults to 0.
- as* Alpha component shift, defaults to 0.

Definition at line 205 of file [colors](#).

### 11.164.2.2 template<typename VBI > L4Re::Video::Pixel\_info::Pixel\_info ( VBI const \* *vbi* ) [inline, explicit]

Convenience constructor.

#### Parameters

- vbi* Suitable information structure. Convenience constructor to create the pixel info from a VESA Framebuffer Info.

Definition at line 217 of file [colors](#).

### 11.164.3 Member Function Documentation

#### 11.164.3.1 Color\_component const& L4Re::Video::Pixel\_info::r ( ) const [inline]

Return the red color component of the pixel.

##### Returns

Red color component.

Definition at line 119 of file [colors](#).

#### 11.164.3.2 Color\_component const& L4Re::Video::Pixel\_info::g ( ) const [inline]

Return the green color component of the pixel.

##### Returns

Green color component.

Definition at line 125 of file [colors](#).

#### 11.164.3.3 Color\_component const& L4Re::Video::Pixel\_info::b ( ) const [inline]

Return the blue color component of the pixel.

##### Returns

Blue color component.

Definition at line 131 of file [colors](#).

#### 11.164.3.4 Color\_component const& L4Re::Video::Pixel\_info::a ( ) const [inline]

Return the alpha color component of the pixel.

##### Returns

Alpha color component.

Definition at line 137 of file [colors](#).

#### 11.164.3.5 unsigned char L4Re::Video::Pixel\_info::bytes\_per\_pixel ( ) const [inline]

Query size of pixel in bytes.

##### Returns

Size of pixel in bytes.

Definition at line 143 of file [colors](#).

**11.164.3.6 unsigned char L4Re::Video::Pixel\_info::bits\_per\_pixel( ) const [inline]**

Number of bits of the pixel.

**Returns**

Number of bits used by the pixel.

Definition at line 149 of file [colors](#).

**11.164.3.7 bool L4Re::Video::Pixel\_info::has\_alpha( ) const [inline]**

Return whether the pixel has an alpha channel.

**Returns**

True if the pixel has an alpha channel, false if not.

Definition at line 156 of file [colors](#).

**11.164.3.8 void L4Re::Video::Pixel\_info::r( Color\_component const & c ) [inline]**

Set the red color component of the pixel.

**Parameters**

*c* Red color component.

Definition at line 162 of file [colors](#).

**11.164.3.9 void L4Re::Video::Pixel\_info::g( Color\_component const & c ) [inline]**

Set the green color component of the pixel.

**Parameters**

*c* Green color component.

Definition at line 168 of file [colors](#).

**11.164.3.10 void L4Re::Video::Pixel\_info::b( Color\_component const & c ) [inline]**

Set the blue color component of the pixel.

**Parameters**

*c* Blue color component.

Definition at line 174 of file [colors](#).

**11.164.3.11 void L4Re::Video::Pixel\_info::a ( Color\_component const & *c* ) [inline]**

Set the alpha color component of the pixel.

**Parameters**

*c* Alpha color component.

Definition at line 180 of file [colors](#).

**11.164.3.12 void L4Re::Video::Pixel\_info::bytes\_per\_pixel ( unsigned char *bpp* ) [inline]**

Set the size of the pixel in bytes.

**Parameters**

*bpp* Size of pixel in bytes.

Definition at line 186 of file [colors](#).

**11.164.3.13 bool L4Re::Video::Pixel\_info::operator== ( Pixel\_info const & *o* ) const [inline]**

Compare for complete equality of the color space.

**Parameters**

*o* A [Pixel\\_info](#) to compare to.

**Returns**

true if the both [Pixel\\_info](#)'s are equal, false if not.

Definition at line 229 of file [colors](#).

**11.164.3.14 template<typename STREAM> STREAM& L4Re::Video::Pixel\_info::dump ( STREAM & *s* ) const [inline]**

Dump information on the pixel to a stream.

**Parameters**

*s* Stream

**Returns**

The stream

Definition at line 240 of file [colors](#).

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

- l4/re/video/colors

## 11.165 L4Re::Util::Ref\_cap< T > Struct Template Reference

Automatic capability that implements automatic free and unmap of the capability selector.

### 11.165.1 Detailed Description

**template<typename T> struct L4Re::Util::Ref\_cap< T >**

Automatic capability that implements automatic free and unmap of the capability selector.

#### Parameters

*T* the type of the object that is referred by the capability.

This kind of automatic capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero.

Usage:

```
L4Re::Util::Ref_cap<L4Re::Dataspace>::Ca global_ds_cap;

{
    L4Re::Util::Ref_cap<L4Re::Dataspace>::Cap
        ds_cap(L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>());
    // reference count for the allocated cap selector is now 1
*
    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

    global_ds_cap = ds_cap;
    // reference count is now 2
    ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
```

Definition at line 223 of file [cap\\_alloc](#).

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

- [l4/re/util/cap\\_alloc](#)

## 11.166 L4Re::Util::Ref\_del\_cap< T > Struct Template Reference

Automatic capability that implements automatic free and unmap+delete of the capability selector.

### 11.166.1 Detailed Description

**template<typename T> struct L4Re::Util::Ref\_del\_cap< T >**

Automatic capability that implements automatic free and unmap+delete of the capability selector.

#### Parameters

*T* the type of the object that is referred by the capability.

This kind of automatic capability implements a counted reference to a capability selector. The capability shall be unmapped and freed when the reference count in the allocator goes to zero. The main difference to [Ref\\_cap](#) is that the unmap is done with the deletion flag enabled and this leads to the deletion of the object if the current task holds appropriate deletion rights.

Usage:

```
L4Re::Util::Ref_del_cap<L4Re::Dataspace>::Ca global_ds_cap;

{
    L4Re::Util::Ref_del_cap<L4Re::Dataspace>::Cap
        ds_cap(L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>());
    // reference count for the allocated cap selector is now 1

    // use the dataspace cap
    L4Re::chksys(mem_alloc->alloc(4096, ds_cap.get()));

    global_ds_cap = ds_cap;
    // reference count is now 2
    ...
}
// reference count dropped to 1 (ds_cap is no longer existing).
...
global_ds_cap = L4_INVALID_CAP;
// reference count dropped to 0 (data space shall be deleted).
```

Definition at line [263](#) of file [cap\\_alloc](#).

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

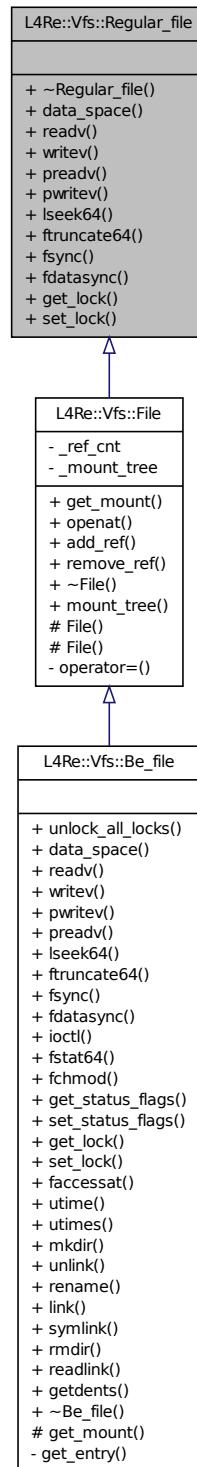
- l4/re/util/cap\_alloc

## 11.167 L4Re::Vfs::Regular\_file Class Reference

Interface for a POSIX file that provides regular file semantics.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Regular\_file:



## Public Member Functions

- virtual `L4::Cap< L4Re::Dataspace > data_space () const =0 throw ()`  
*Get an L4Re::Dataspace object for the file.*
- virtual `ssize_t readv (const struct iovec *, int iovcnt)=0 throw ()`  
*Read one or more blocks of data from the file.*
- virtual `ssize_t writev (const struct iovec *, int iovcnt)=0 throw ()`  
*Write one or more blocks of data to the file.*
- virtual `off64_t lseek64 (off64_t, int)=0 throw ()`  
*Change the file pointer.*
- virtual `int ftruncate64 (off64_t pos)=0 throw ()`  
*Truncate the file at the given position.*
- virtual `int fsync () const =0 throw ()`  
*Sync the data and meta data to persistent storage.*
- virtual `int fdatasync () const =0 throw ()`  
*Sync the data to persistent storage.*
- virtual `int get_lock (struct flock64 *lock)=0 throw ()`  
*Test if the given lock can be placed in the file.*
- virtual `int set_lock (struct flock64 *lock, bool wait)=0 throw ()`  
*Acquire or release the given lock on the file.*

### 11.167.1 Detailed Description

Interface for a POSIX file that provides regular file semantics. Real objects use always the combined `L4Re::Vfs::File` interface.

Definition at line 261 of file `vfs.h`.

### 11.167.2 Member Function Documentation

#### 11.167.2.1 virtual L4::Cap<L4Re::Dataspace> L4Re::Vfs::Regular\_file::data\_space ( ) const throw () [pure virtual]

Get an L4Re::Dataspace object for the file.

This is used as a backend for POSIX mmap and mmap2 functions.

##### Note

mmap is not possible if the functions returns an invalid capability.

##### Returns

A capability to an L4Re::Dataspace, that represents the files contents in an L4Re way.

**11.167.2.2 virtual ssize\_t L4Re::Vfs::Regular\_file::readv ( const struct iovec \*, int iovcnt ) throw () [pure virtual]**

Read one or more blocks of data from the file.

This function acts as backend for POSIX read and ready calls and reads data starting for the f\_pos pointer of that open file. The file pointer is advanced according to the number of read bytes.

#### Returns

The number of bytes read from the file, or <0 on error.

**11.167.2.3 virtual ssize\_t L4Re::Vfs::Regular\_file::writev ( const struct iovec \*, int iovcnt ) throw () [pure virtual]**

Write one or more blocks of data to the file.

This function acts as backend for POSIX write and writev calls. The data is written starting at the current file pointer and the file pointer must be advanced according to the number of written bytes.

#### Returns

The number of bytes written to the file, or <0 on error.

**11.167.2.4 virtual off64\_t L4Re::Vfs::Regular\_file::lseek64 ( off64\_t, int ) throw () [pure virtual]**

Change the file pointer.

This is the backend for POSIX seek, lseek and friends.

#### Returns

The new file position, or <0 on error.

**11.167.2.5 virtual int L4Re::Vfs::Regular\_file::ftruncate64 ( off64\_t pos ) throw () [pure virtual]**

Truncate the file at the given position.

This function is the backend for truncate and friends.

#### Parameters

*pos* The offset at which the file shall be truncated.

#### Returns

0 on success, or <0 on error.

**11.167.2.6 virtual int L4Re::Vfs::Regular\_file::fsync ( ) const throw () [pure virtual]**

Sync the data and meta data to persistent storage.

This is the backend for POSIX fsync.

**11.167.2.7 virtual int L4Re::Vfs::Regular\_file::fdatasync ( ) const throw () [pure virtual]**

Sync the data to persistent storage.

This is the backend for POSIX fdatasync.

**11.167.2.8 virtual int L4Re::Vfs::Regular\_file::get\_lock ( struct flock64 \* lock ) throw () [pure virtual]**

Test if the given lock can be placed in the file.

This function is used as backend for fcntl F\_GETLK commands.

**Parameters**

*lock* The lock that shall be placed on the file. The *l\_type* member will contain F\_UNLCK if the lock could be placed.

**Returns**

0 on success, <0 on error.

**11.167.2.9 virtual int L4Re::Vfs::Regular\_file::set\_lock ( struct flock64 \* lock, bool wait ) throw () [pure virtual]**

Acquire or release the given lock on the file.

This function is used as backend for fcntl F\_SETLK and F\_SETLKW commands.

**Parameters**

*lock* The lock that shall be placed on the file.

*wait* If true, then block if there is a conflicting lock on the file.

**Returns**

0 on success, <0 on error.

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

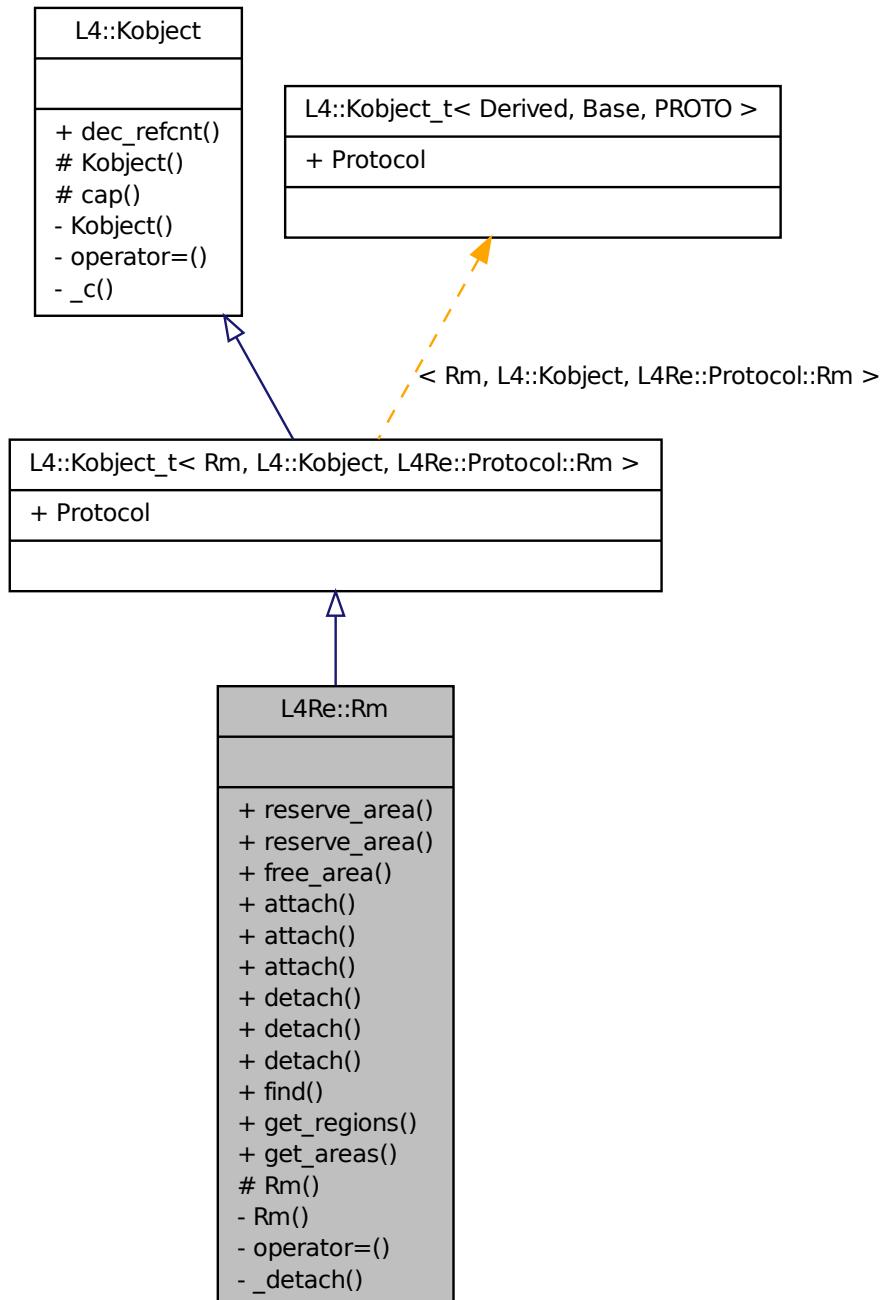
- l4/l4re\_vfs/vfs.h

**11.168 L4Re::Rm Class Reference**

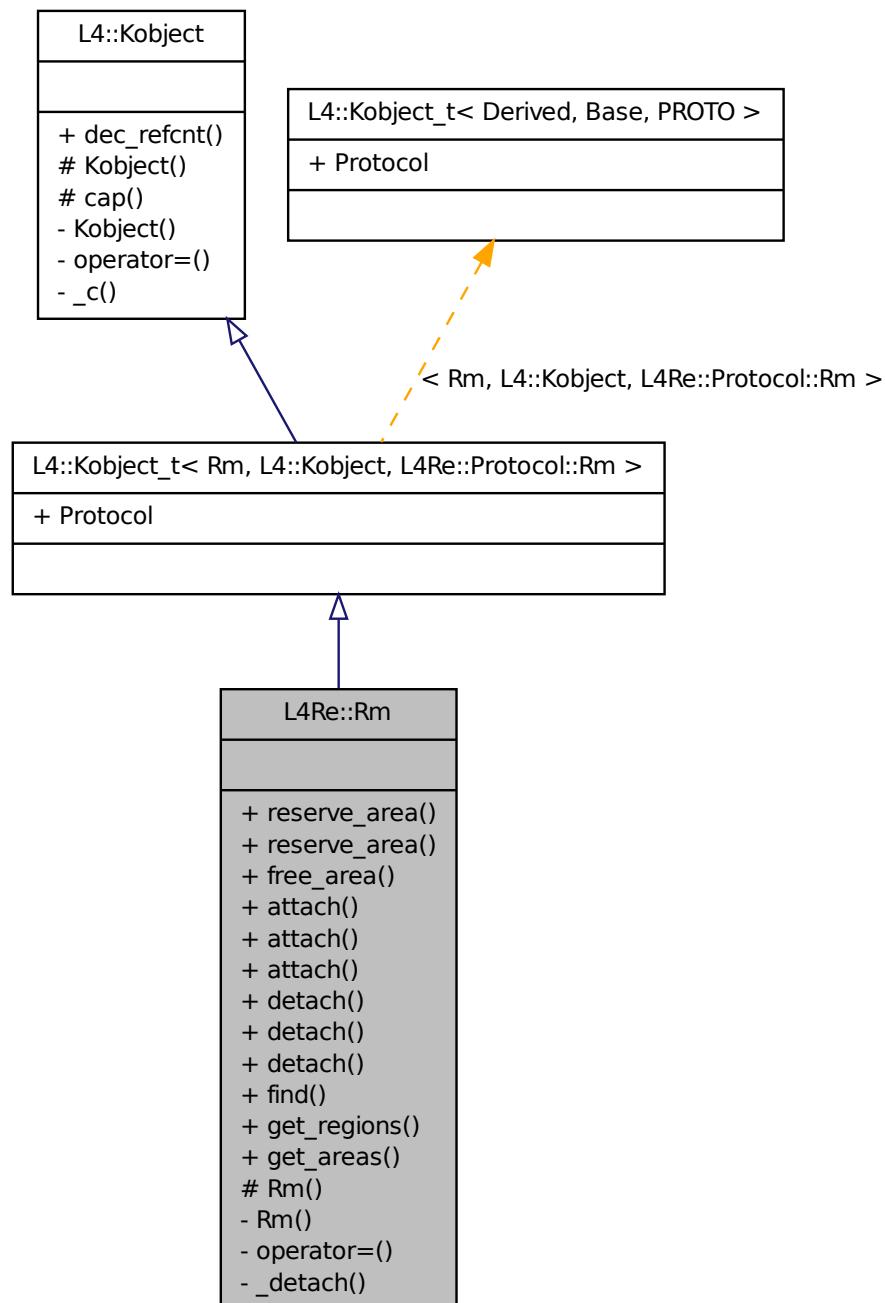
Region map.

```
#include <l4/re/rm>
```

Inheritance diagram for L4Re::Rm:



Collaboration diagram for L4Re::Rm:



## Public Types

- enum `Detach_result` { `Detached_ds` = 0, `Kept_ds` = 1, `Split_ds` = 2 , `Detach_again` = 4 }
 

*Result values for detach operation.*
- enum `Region_flags` {
   
`Read_only` = 0x01, `Detach_free` = 0x02, `Pager` = 0x04, `Reserved` = 0x08,
   
`Region_flags` = 0x0f }
 

*Flags for regions.*
- enum `Attach_flags` { `Search_addr` = 0x20, `In_area` = 0x40, `Eager_map` = 0x80, `Attach_flags` = 0xf0 }
 

*Flags for attach operation.*
- enum `Detach_flags` { `Detach_exact` = 1, `Detach_overlap` = 2 }
 

*Flags for detach operation.*

## Public Member Functions

- long `reserve_area` (`l4_addr_t` \*start, unsigned long size, unsigned flags=0, unsigned char align=L4\_-PAGESHIFT) const throw ()
 

*Reserve the given area in the region map.*
- template<typename T >
 long `reserve_area` (T \*\*start, unsigned long size, unsigned flags=0, unsigned char align=L4\_-PAGESHIFT) const throw ()
 

*Reserve the given area in the region map.*
- long `free_area` (`l4_addr_t` addr) const throw ()
 

*Free an area from the region map.*
- long `attach` (`l4_addr_t` \*start, unsigned long size, unsigned long flags, `L4::Cap< Dataspace >` mem, `l4_addr_t` offs=0, unsigned char align=L4\_PAGESHIFT) const throw ()
 

*Attach a data space to a region.*
- template<typename T >
 long `attach` (T \*\*start, unsigned long size, unsigned long flags, `L4::Cap< Dataspace >` mem, `l4_addr_t` offs=0, unsigned char align=L4\_PAGESHIFT) const throw ()
 

*Attach a dataspace to a region.*
- int `detach` (`l4_addr_t` addr, `L4::Cap< Dataspace >` \*mem, `L4::Cap< L4::Task >` const &task=This\_task) const throw ()
 

*Detach a region from the address space.*
- int `detach` (void \*addr, `L4::Cap< Dataspace >` \*mem, `L4::Cap< L4::Task >` const &task=This\_task) const throw ()
 

*Detach a region from the address space.*

- int `detach` (`l4_addr_t` start, unsigned long size, `L4::Cap< Dataspace >` \*mem, `L4::Cap< L4::Task >` const &task) const throw ()

*Detach all regions of the specified interval.*

- int `find` (`l4_addr_t` \*addr, unsigned long \*size, `l4_addr_t` \*offset, unsigned \*flags, `L4::Cap< Dataspace >` \*m) throw ()

*Find a region given an address and size.*

## 11.168.1 Detailed Description

Region map.

Definition at line [69](#) of file `rm`.

## 11.168.2 Member Enumeration Documentation

### 11.168.2.1 enum L4Re::Rm::Detach\_result

Result values for detach operation.

**Enumerator:**

*Detached\_ds* Detached data space.

*Kept\_ds* Kept data space.

*Split\_ds* Splitted data space, and done.

*Detach\_again* Detached data space, more to do.

Definition at line [76](#) of file `rm`.

### 11.168.2.2 enum L4Re::Rm::Region\_flags

Flags for regions.

**Enumerator:**

*Read\_only* Region is read-only.

*Detach\_free* Free the portion of the data space after detach.

*Pager* Region has a pager.

*Reserved* Region is reserved (blocked).

*Region\_flags* Mask of all region flags.

Definition at line [87](#) of file `rm`.

### 11.168.2.3 enum L4Re::Rm::Attach\_flags

Flags for attach operation.

**Enumerator:**

- Search\_addr* Search for a suitable address range.
- In\_area* Search only in area, or map into area.
- Eager\_map* Eagerly map the attached data space in.
- Attach\_flags* Mask of all attach flags.

Definition at line 99 of file [rm](#).

**11.168.2.4 enum L4Re::Rm::Detach\_flags**

Flags for detach operation.

**Enumerator:**

- Detach\_exact* Do an unmap of the exact region given.
- Detach\_overlap* Do an unmap of all overlapping regions.

Definition at line 109 of file [rm](#).

**11.168.3 Member Function Documentation****11.168.3.1 long L4Re::Rm::reserve\_area ( l4\_addr\_t \* start, unsigned long size, unsigned flags = 0, unsigned char align = L4\_PAGESHIFT ) const throw ()**

Reserve the given area in the region map.

**Parameters**

- start* The virtual start address of the area to reserve.
- size* The size of the area to reserve (in bytes).
- flags* Flags for the reserved area (see [Region\\_flags](#) and [Attach\\_flags](#)).
- align* Alignment of area if searched as bits (log2 value).

**Return values**

- start* Start of address.

**Returns**

0 on success, <0 on error

- [-L4\\_EADDRNOTAVAIL](#)
- IPC errors

This function reserves an area within the virtual address space implemented by the region map. There are two kinds of areas available:

- Reserved areas (*flags* = [Reserved](#)), where no data spaces can be attached
- Special purpose areas (*flags* = 0), where data spaces can be attached to the area via the [In\\_area](#) flag and a start address within the area itself.

### Note

When searching for a free place in the virtual address space (with *flags* = `Search_addr`), the space between *start* and the end of the virtual address space is searched.

Definition at line 30 of file [rm\\_impl.h](#).

**11.168.3.2 template<typename T > long L4Re::Rm::reserve\_area ( T \*\* start, unsigned long size, unsigned flags = 0, unsigned char align = `L4_PAGESHIFT` ) const throw () [inline]**

Reserve the given area in the region map.

### Parameters

*start* The virtual start address of the area to reserve.

*size* The size of the area to reserve (in bytes).

*flags* Flags for the reserved area (see [Region\\_flags](#) and [Attach\\_flags](#)).

*align* Alignment of area if searched as bits (log2 value).

### Return values

*start* Start of address.

### Returns

0 on success, <0 on error

- [-L4\\_EADDRNOTAVAIL](#)
- IPC errors

For more information, please refer to the analogous function

### See also

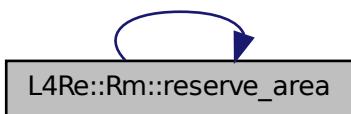
[L4Re::Rm::reserve\\_area](#).

Definition at line 232 of file [rm](#).

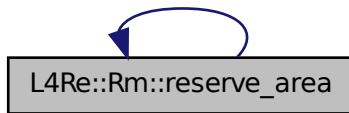
References [reserve\\_area\(\)](#).

Referenced by [reserve\\_area\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 11.168.3.3 long L4Re::Rm::free\_area ( l4\_addr\_t *addr* ) const throw ()

Free an area from the region map.

#### Parameters

*addr* An address within the area to free.

#### Returns

0 on success, <0 on error

- [-L4\\_ENOENT](#)
- IPC errors

#### Note

The data spaces that are attached to that area are not detached by this operation.

#### See also

[reserve\\_area\(\)](#) for more information about areas.

Definition at line 44 of file [rm\\_impl.h](#).

### 11.168.3.4 long L4Re::Rm::attach ( l4\_addr\_t \* *start*, unsigned long *size*, unsigned long *flags*, L4::Cap< Dataspace > *mem*, l4\_addr\_t *offs* = 0, unsigned char *align* = L4\_PAGESHIFT ) const throw ()

Attach a data space to a region.

#### Parameters

*start* Virtual start address

*size* Size of the data space to attach (in bytes)

*flags* Flags, see [Attach\\_flags](#) and [Region\\_flags](#)

*mem* Data space

*offs* Offset into the data space to use

**align** Alignment of the virtual region, log2-size, default: a page ([L4\\_PAGESHIFT](#)), Only meaningful if the [Search\\_addr](#) flag is used.

#### Return values

**start** Start of region if [Search\\_addr](#) was used.

#### Returns

0 on success, <0 on error

- [-L4\\_ENOENT](#)
- [-L4\\_EPERM](#)
- [-L4\\_EINVAL](#)
- [-L4\\_EADDRNOTAVAIL](#)
- IPC errors

Makes the whole or parts of a data space visible in the virtual memory of the corresponding task. The corresponding region in the virtual address space is backed with the contents of the dataspace.

#### Note

When searching for a free place in the virtual address space, the space between *start* and the end of the virtual address space is searched.

There is no region object created, instead the region is defined by a virtual address within this range (see [find](#)).

Definition at line 52 of file [rm\\_impl.h](#).

**11.168.3.5 template<typename T> long L4Re::Rm::attach ( T \*\* *start*, unsigned long *size*,  
unsigned long *flags*, L4::Cap< Dataspace > \* *mem*, l4\_addr\_t *offs* = 0, unsigned char  
*align* = [L4\\_PAGESHIFT](#) ) const throw () [inline]**

Attach a dataspace to a region.

#### See also

[attach](#)

Definition at line 292 of file [rm](#).

**11.168.3.6 int L4Re::Rm::detach ( l4\_addr\_t *addr*, L4::Cap< Dataspace > \* *mem*, L4::Cap<  
L4::Task > const & *task* = [This\\_task](#) ) const throw () [inline]**

Detach a region from the address space.

#### Parameters

**addr** Virtual address of region, any address within the region is valid.

#### Return values

**mem** [Dataspace](#) that is affected. Give 0 if not interested.

**Parameters**

*task* If given, task specifies the task where the pages are unmapped. Give 0 for none. Default is current task.

**Returns**

[Detach\\_result](#) on success, <0 on error

- -L4\_ENOENT
- IPC errors

Frees a region in the virtual address space given by *addr* (address type). The corresponding part of the address space is now available again.

Definition at line [443](#) of file [rm](#).

**11.168.3.7 int L4Re::Rm::detach ( void \* *addr*, L4::Cap< Dataspace > \* *mem*, L4::Cap< L4::Task > const & *task* = *This\_task* ) const throw () [inline]**

Detach a region from the address space.

**Parameters**

*addr* Virtual address of region, any address within the region is valid.

**Return values**

*mem* [Dataspace](#) that is affected. Give 0 if not interested.

**Parameters**

*task* If given, task specifies the task where the pages are unmapped. Give 0 for none. Default is current task.

**Returns**

[Detach\\_result](#) on success, <0 on error

- -L4\_ENOENT
- IPC errors

Frees a region in the virtual address space given by *addr* (void pointer type). The corresponding part of the address space is now available again.

Definition at line [448](#) of file [rm](#).

**11.168.3.8 int L4Re::Rm::detach ( l4\_addr\_t *start*, unsigned long *size*, L4::Cap< Dataspace > \* *mem*, L4::Cap< L4::Task > const & *task* ) const throw () [inline]**

Detach all regions of the specified interval.

**Parameters**

*start* Start of area to detach, must be within region.

*size* Size of area to detach (in bytes).

**Return values**

*mem* Dataspace that is affected. Give 0 if not interested.

**Parameters**

*task* Specifies the task where the pages are unmapped. Give 0 for none.

**Returns**

`Detach_result` on success, <0 on error

- -L4\_ENOENT
- IPC errors

Frees all regions within the interval given by start and size. If a region overlaps the start or the end of the interval this region is only detached partly. If the interval is within one region the original region is split up into two separate regions.

Definition at line 453 of file `rm`.

**11.168.3.9 int L4Re::Rm::find ( l4\_addr\_t \* addr, unsigned long \* size, l4\_addr\_t \* offset, unsigned \* flags, L4::Cap< Dataspace > \* m ) throw ()**

Find a region given an address and size.

**Parameters**

*addr* Address to look for

*size* Size of the area to look for (in bytes).

**Return values**

*addr* Start address of the found region.

*size* Size of the found region (in bytes).

*offset* Offset at the beginning of the region within the associated dataspace.

*flags* Region flags, see `Region_flags`.

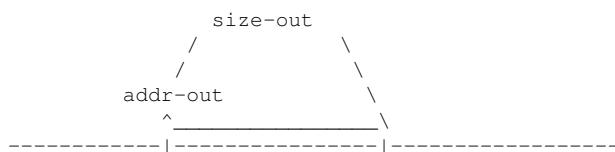
*m* Associated dataspace or paging service.

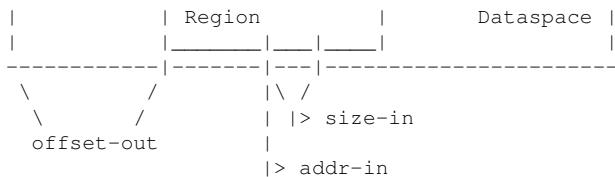
**Returns**

0 on success, <0 on error

- -L4\_EPERM: not allowed
- -L4\_ENOENT: not found
- IPC errors

This function returns the properties of the region that contains the area described by the *addr* and *size* parameter.



**Note**

The value of the size input parameter should be 1 to assure that a region can be determined unambiguously.

The documentation for this class was generated from the following files:

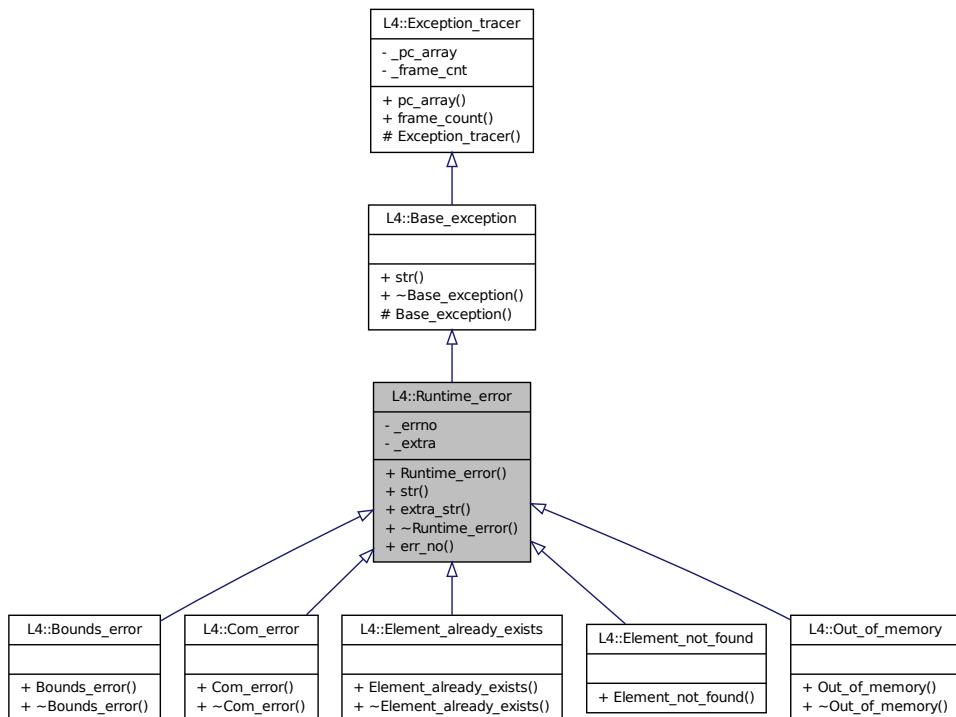
- l4/re/rm
- l4/re/impl/rm\_impl.h

## 11.169 L4::Runtime\_error Class Reference

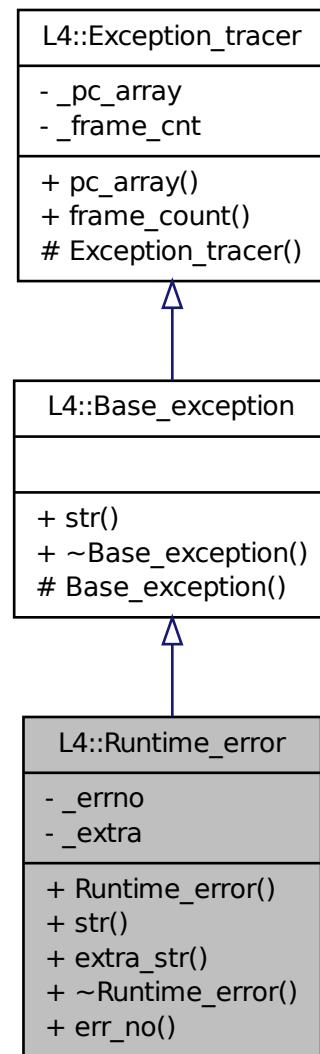
Exception for an abstract runtime error.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Runtime\_error:



Collaboration diagram for L4::Runtime\_error:



## Public Member Functions

- `char const * str () const throw ()`

*Should return a human readable string for the exception.*

### 11.169.1 Detailed Description

Exception for an abstract runtime error. This is the base class for a set of exceptions that cover all errors that have a C error value (see [l4\\_error\\_code\\_t](#)).

Definition at line [136](#) of file [exceptions](#).

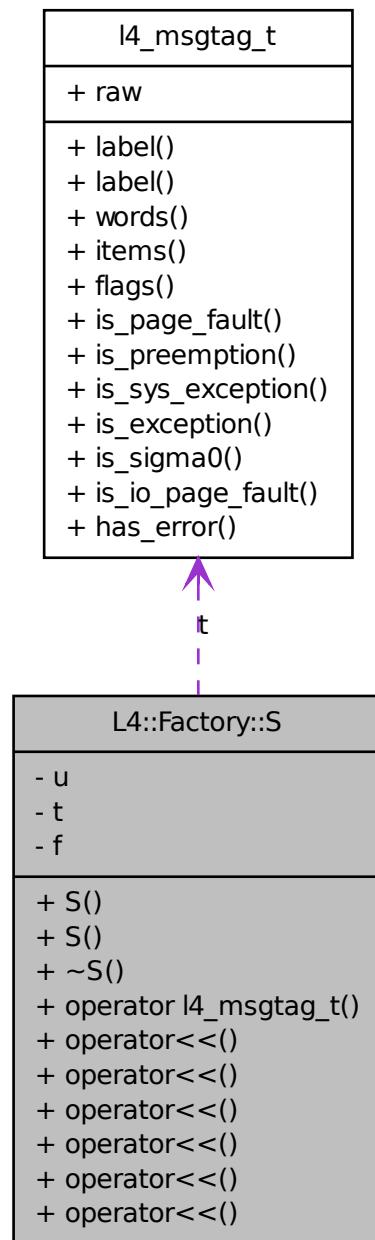
The documentation for this class was generated from the following file:

- l4/cxx/exceptions

## 11.170 L4::Factory::S Class Reference

Stream class for the [create\(\)](#) argument stream.

Collaboration diagram for L4::Factory::S:



## Public Member Functions

- `S (S const &co)`

*create a copy.*

- **S (l4\_cap\_idx\_t f, long obj, L4::Cap< L4::Kobject > target, l4\_utcb\_t \*utcb) throw ()**  
*create a stream for a specific `create()` call.*
- **~S ()**  
*Commit the operation in the destructor to have a cool syntax for `create()`.*
- **operator l4\_mshtag\_t ()**  
*Explicitely commits the operation and returns the result.*
- **S & operator<< (l4\_mword\_t i)**  
*Put a single l4\_mword\_t as next argument.*
- **S & operator<< (l4\_umword\_t i)**  
*Put a single l4\_umword\_t as next argument.*
- **S & operator<< (char const \*s)**  
*Add a zero-terminated string as next argument.*
- **S & operator<< (Lstr const &s)**  
*Add a pascal string as next argument.*
- **S & operator<< (Nil)**  
*Add an empty argument.*
- **S & operator<< (l4\_fpage\_t d)**  
*Add a flex page as next argument.*

## 11.170.1 Detailed Description

Stream class for the `create()` argument stream. This stream allows a variable number of arguments to be added to a `create()` call.

Definition at line 82 of file [factory](#).

## 11.170.2 Constructor & Destructor Documentation

### 11.170.2.1 L4::Factory::S::S ( l4\_cap\_idx\_t f, long obj, L4::Cap< L4::Kobject > target, l4\_utcb\_t \* utcb ) throw () [inline]

create a stream for a specific `create()` call.

#### Parameters

*f* is the capability for the factory object ([L4::Factory](#)).

*obj* is the protocol ID to describe the type of the object that shall be created.

*target* is the capability selector for the new object.

*utcb* is the UTCB that shall be used for the operation.

Definition at line 105 of file [factory](#).

### 11.170.3 Member Function Documentation

#### 11.170.3.1 L4::Factory::S::operator l4\_mshtag\_t( ) [inline]

Explicitely commits the operation and returns the result.

##### Returns

The result of the [create\(\)](#) operation.

Definition at line [124](#) of file [factory](#).

References [l4\\_mshtag\\_t::raw](#).

#### 11.170.3.2 S& L4::Factory::S::operator<<( l4\_mword\_t i ) [inline]

Put a single l4\_mword\_t as next argument.

##### Parameters

*i* is the value to add as next argument.

Definition at line [135](#) of file [factory](#).

#### 11.170.3.3 S& L4::Factory::S::operator<<( l4\_umword\_t i ) [inline]

Put a single l4\_umword\_t as next argument.

##### Parameters

*i* is the value to add as next argument.

Definition at line [145](#) of file [factory](#).

#### 11.170.3.4 S& L4::Factory::S::operator<<( char const \* s ) [inline]

Add a zero-terminated string as next argument.

##### Parameters

*s* is the string to add as next argument.

Definition at line [155](#) of file [factory](#).

#### 11.170.3.5 S& L4::Factory::S::operator<<( Lstr const & s ) [inline]

Add a pascal string as next argument.

##### Parameters

*s* is the string to add as next argument.

Definition at line [165](#) of file [factory](#).

References [L4::Factory::Lstr::len](#), and [L4::Factory::Lstr::s](#).

**11.170.3.6 S & L4::Factory::S::operator<<( l4\_fpage\_t d ) [inline]**

Add a flex page as next argument.

**Parameters**

*d* is the flex page to add (there will be no map operation).

Definition at line 184 of file [factory](#).

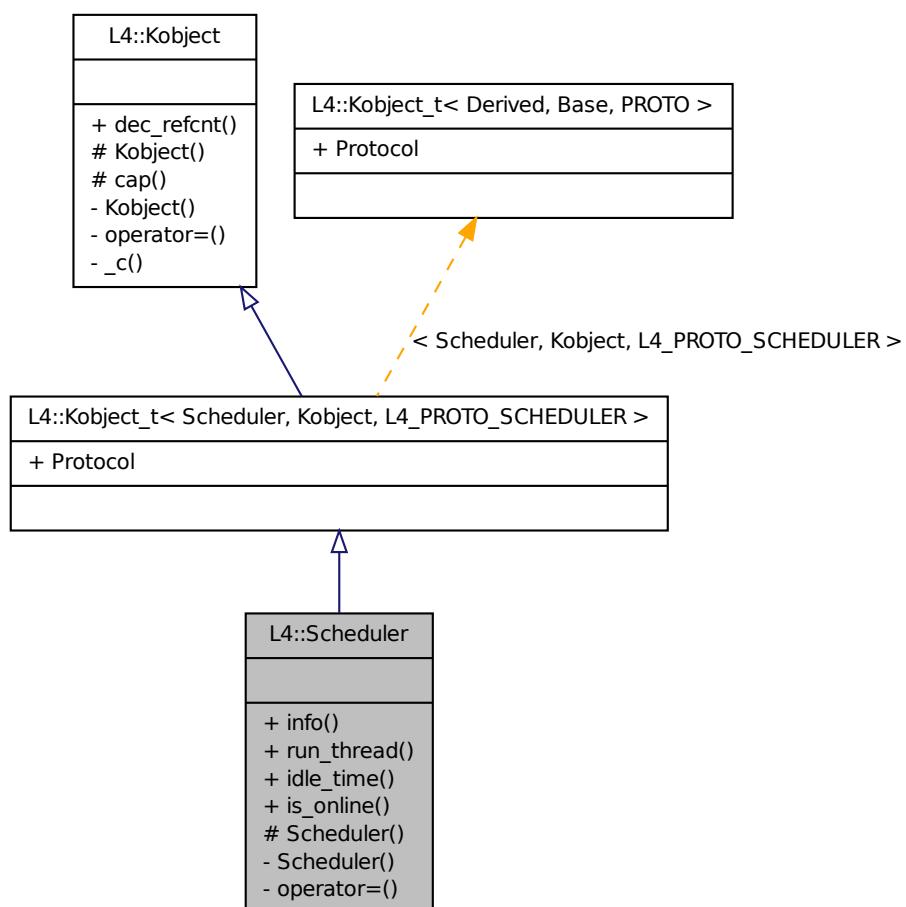
The documentation for this class was generated from the following file:

- [l4/sys/factory](#)

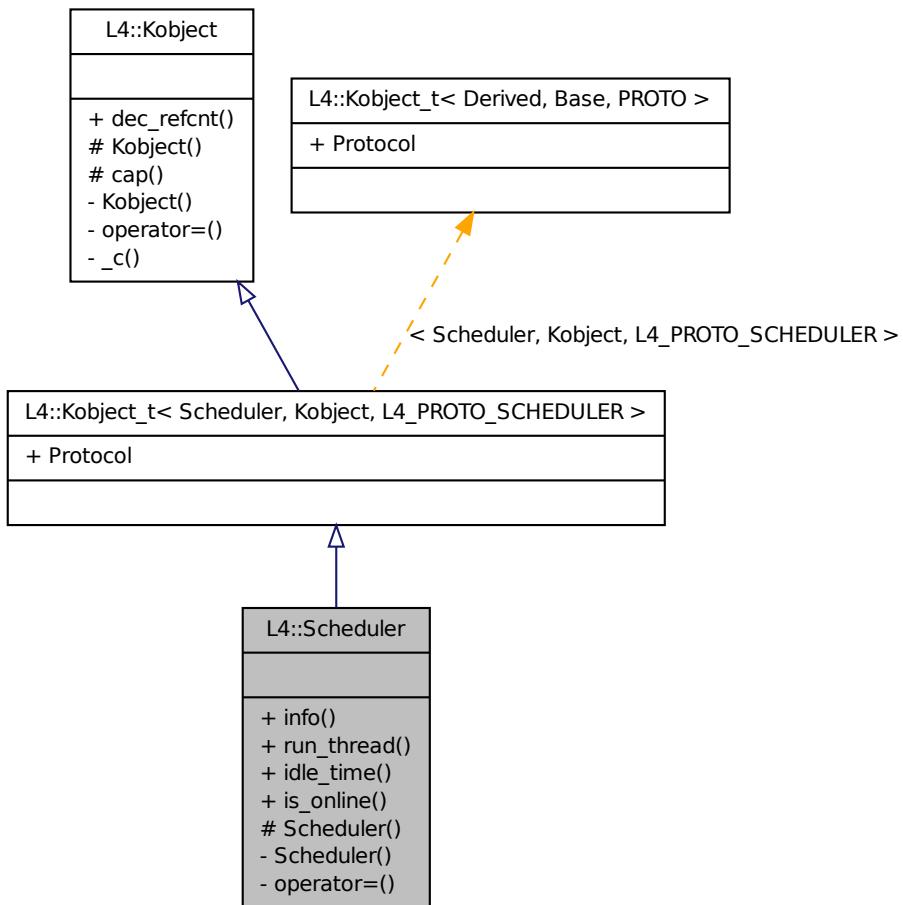
## 11.171 L4::Scheduler Class Reference

[Scheduler](#) object.

Inheritance diagram for L4::Scheduler:



Collaboration diagram for L4::Scheduler:



## Public Member Functions

- `l4_mshtag_t info (l4_umword_t *cpu_max, l4_sched_cpu_set_t *cpus, l4_utcb_t *utcb=l4_utcb()) const throw ()`
- `l4_mshtag_t run_thread (Cap< Thread > const &thread, l4_sched_param_t const &sp, l4_utcb_t *utcb=l4_utcb()) const throw ()`
- `l4_mshtag_t idle_time (l4_sched_cpu_set_t const &cpus, l4_utcb_t *utcb=l4_utcb()) const throw ()`
- `bool is_online (l4_umword_t cpu, l4_utcb_t *utcb=l4_utcb()) const throw ()`

### 11.171.1 Detailed Description

Scheduler object. #include <l4/sys/scheduler>

**See also**

[Scheduler](#) for an overview description.

Definition at line 38 of file [scheduler](#).

## 11.171.2 Member Function Documentation

### 11.171.2.1 `l4_mshtag_t L4::Scheduler::info ( l4_umword_t * cpu_max, l4_sched_cpu_set_t * cpus, l4_utcb_t * utcb = L4_utcb() ) const throw () [inline]`

Get scheduler information.

**Parameters**

*scheduler* Scheduler object.

**Return values**

*cpu\_max* maximum number of CPUs ever available.

**Parameters**

*cpus* *cpus.offset* is first CPU of interest. *cpus.granularity* (see [l4\\_sched\\_cpu\\_set\\_t](#)).

**Return values**

*cpus* *cpus.map* Bitmap of online CPUs.

**Returns**

0 on success, <0 error code otherwise.

**Note**

*scheduler* is the implicit *this* pointer.

Definition at line 35 of file [scheduler](#).

### 11.171.2.2 `l4_mshtag_t L4::Scheduler::run_thread ( Cap< Thread > const & thread, l4_sched_param_t const & sp, l4_utcb_t * utcb = L4_utcb() ) const throw () [inline]`

Run a thread on a Scheduler.

**Parameters**

*scheduler* Scheduler object.

*thread* Thread to run.

*sp* Scheduling parameters.

**Returns**

0 on success, <0 error code otherwise.

**Note**

*scheduler* is the implicit *this* pointer.

Definition at line 43 of file [scheduler](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 11.171.2.3 l4\_mshtag\_t L4::Scheduler::idle\_time ( l4\_sched\_cpu\_set\_t const & cpus, l4\_utcb\_t \* utcb = l4\_utcb() ) const throw() [inline]

Query idle time of a CPU, in Ås.

**Parameters**

*scheduler* Scheduler object.

*cpus* Set of CPUs to query.

The consumed time is returned as l4\_kernel\_clock\_t at UTCB message register 0.

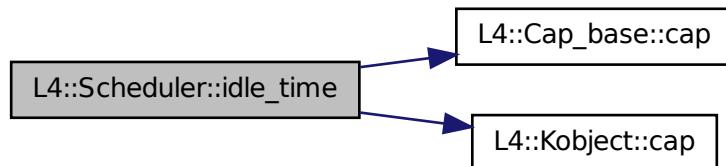
**Note**

*scheduler* is the implicit *this* pointer.

Definition at line 52 of file [scheduler](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



---

**11.171.2.4 bool L4::Scheduler::is\_online ( l4\_umword\_t *cpu*, l4\_utcb\_t \* *utcb* = *L4\_utcb()* )  
const throw () [inline]**

Query if a CPU is online.

#### Parameters

*scheduler* Scheduler object.

*cpu* CPU number.

#### Returns

true if online, false if not (or any other query error).

#### Note

*scheduler* is the implicit *this* pointer.

Definition at line 61 of file [scheduler](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



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

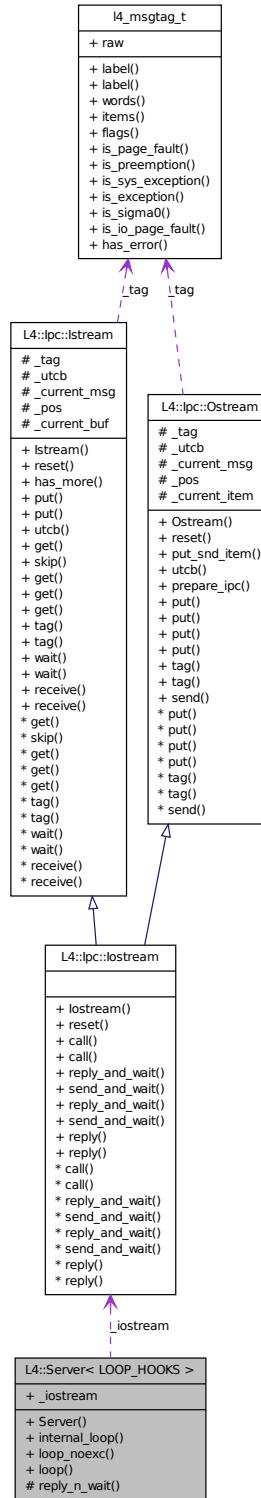
- [l4/sys/scheduler](#)

## 11.172 L4::Server< LOOP\_HOOKS > Class Template Reference

Basic server loop for handling client requests.

Inherited by [L4Re::Util::Registry\\_server< LOOP\\_HOOKS >](#).

Collaboration diagram for L4::Server< LOOP\_HOOKS >:



## Public Member Functions

- **Server (l4\_utcb\_t \*utcb)**  
*Create a basic server loop.*
- template<typename DISPATCH >  
**void internal\_loop (DISPATCH dispatch)**  
*The server loop.*

### 11.172.1 Detailed Description

**template<typename LOOP\_HOOKS = Ipc\_svr::Default\_loop\_hooks> class L4::Server< LOOP\_HOOKS >**

Basic server loop for handling client requests.

#### Parameters

*OBJ* the server inherits from OBJ and calls its dispatch() function.

*LOOP\_HOOKS* the server inherits from LOOP\_HOOKS and calls the hooks defined in LOOP\_HOOKS in the server loop. See [Ipc\\_svr::Default\\_loop\\_hooks](#), [Ipc\\_svr::Ignore\\_errors](#), [Ipc\\_svr::Default\\_timeout](#), [Ipc\\_svr::Compound\\_reply](#), and [Ipc\\_svr::Default\\_setup\\_wait](#).

This is basically a simple server loop that uses a single message buffer for receiving requests and sending replies. The dispatcher determines how incoming messages are handled.

Definition at line 186 of file [ipc\\_server](#).

### 11.172.2 Constructor & Destructor Documentation

**11.172.2.1 template<typename LOOP\_HOOKS = Ipc\_svr::Default\_loop\_hooks> L4::Server< LOOP\_HOOKS >::Server ( l4\_utcb\_t \* utcb ) [inline, explicit]**

Create a basic server loop.

#### Parameters

*msg\_buf* The message buffer for send and receive operations.

*d* The dispatcher to handle incoming messages.

Definition at line 195 of file [ipc\\_server](#).

### 11.172.3 Member Function Documentation

**11.172.3.1 template<typename L > template<typename DISPATCH > void L4::Server< L >::internal\_loop ( DISPATCH dispatch ) [inline]**

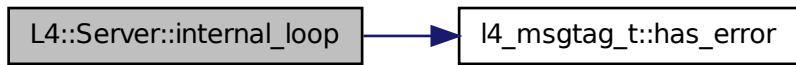
The server loop.

This function usually never returns, it waits for incoming messages calls the dispatcher, sends a reply and waits again.

Definition at line 247 of file [ipc\\_server](#).

References [l4\\_mshtag\\_t::has\\_error\(\)](#).

Here is the call graph for this function:



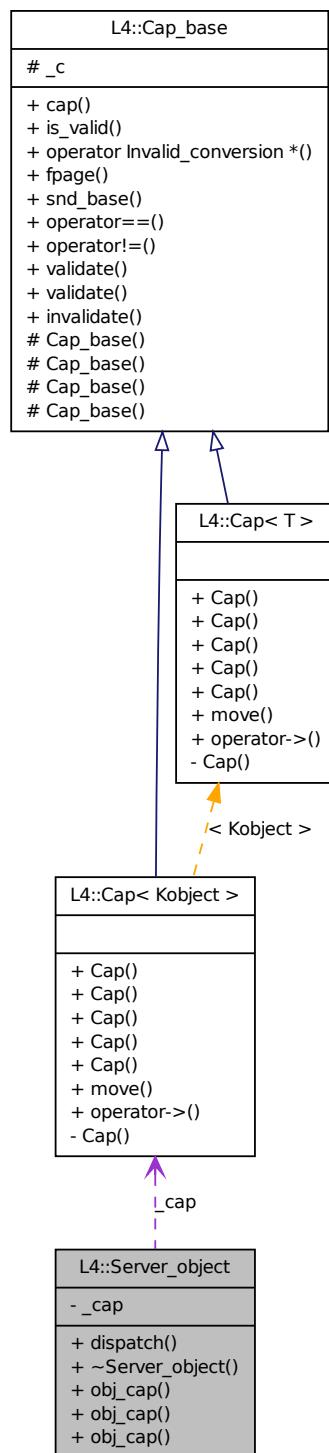
The documentation for this class was generated from the following file:

- [l4/cxx/ipc\\_server](#)

## 11.173 L4::Server\_object Class Reference

Abstract server object to be used with the L4::Registry\_dispatcher.

Collaboration diagram for L4::Server\_object:



## Public Member Functions

- virtual int **dispatch** (unsigned long obj, [Ipc::Iostream](#) &ios)=0

*The abstract handler for client requests to the object.*

### 11.173.1 Detailed Description

Abstract server object to be used with the L4::Registry\_dispatcher. This server object provides an abstract interface that is used by the L4::Registry\_dispatcher model. You can derive subclasses from this interface and implement application specific server objects.

Definition at line 279 of file [ipc\\_server](#).

### 11.173.2 Member Function Documentation

#### 11.173.2.1 virtual int L4::Server\_object::dispatch ( **unsigned long** *obj*, [Ipc::Iostream](#) & *ios* ) [pure virtual]

The abstract handler for client requests to the object.

##### Parameters

*obj* The object ID used for looking up the object.

*op* The opcode for the request.

*ios* The [Ipc::Iostream](#) for reading the request and writing the reply.

##### Returns

Reply, No\_reply, or Invalid\_opcode.

This function must be implemented by application specific server objects. The implementation must unmarshal data from the stream (*ios*) and create a reply by marshalling to the stream (*ios*). For details about the IPC stream see [IPC stream operators](#) .

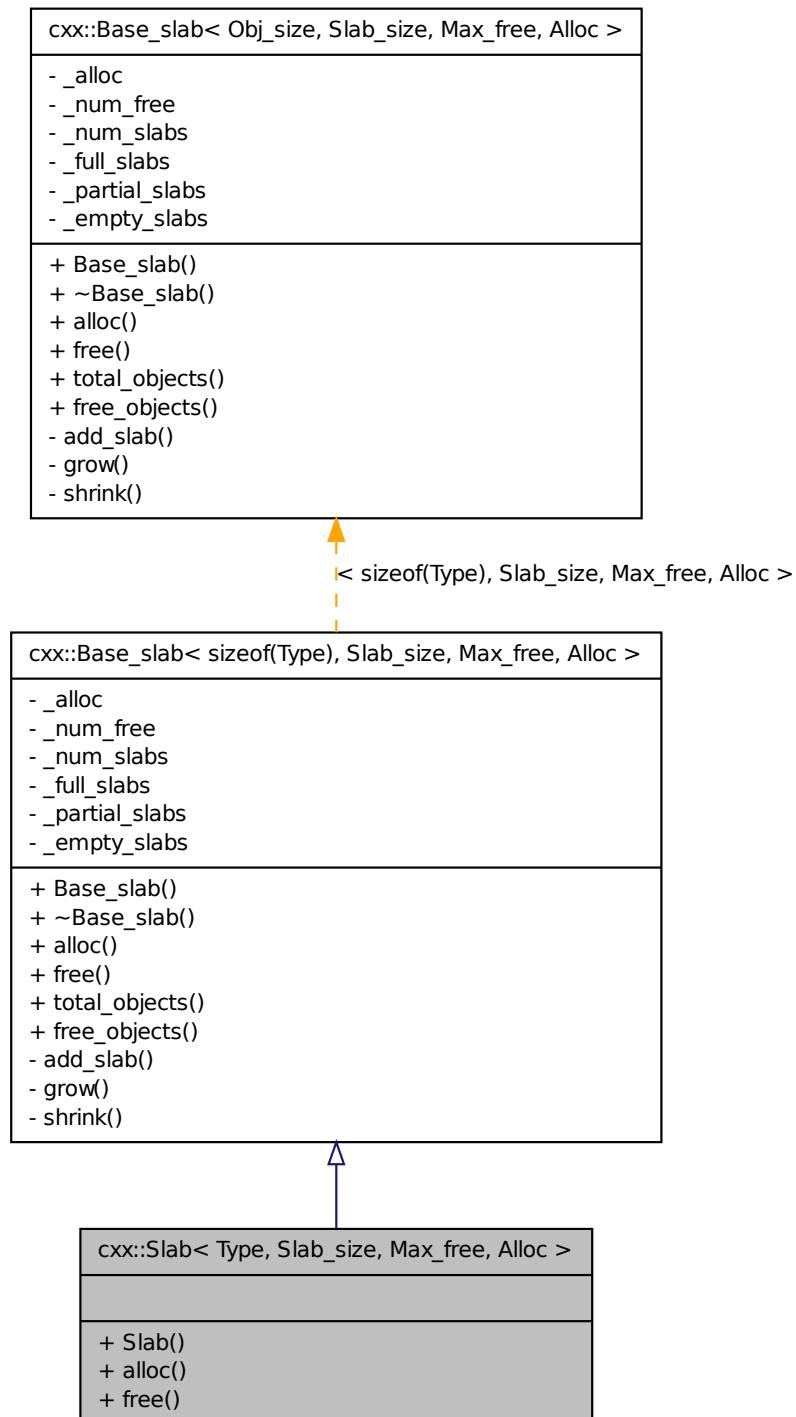
The documentation for this class was generated from the following file:

- 14/cxx/ipc\_server

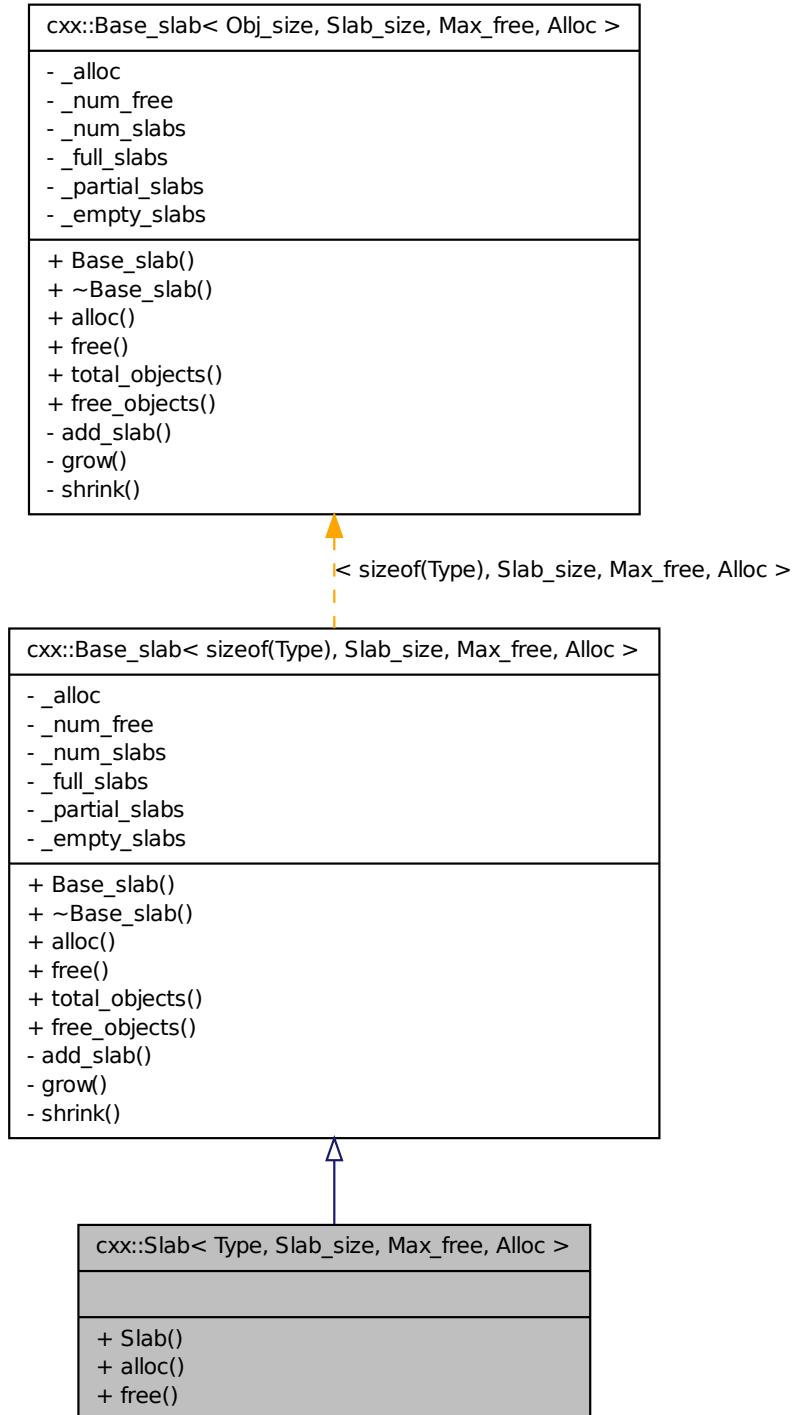
## 11.174 cxx::Slab< Type, Slab\_size, Max\_free, Alloc > Class Template Reference

[Slab](#) allocator for object of type *Type*.

Inheritance diagram for cxx::Slab< Type, Slab\_size, Max\_free, Alloc >:



Collaboration diagram for cxx::Slab< Type, Slab\_size, Max\_free, Alloc >:



## Public Member Functions

- `Type * alloc () throw ()`  
*Allocate an object of type Type.*
- `void free (Type *o) throw ()`  
*Free the object addressed by o.*

### 11.174.1 Detailed Description

```
template<typename Type, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc = New_allocator> class cxx::Slab< Type, Slab_size, Max_free, Alloc >
```

`Slab` allocator for object of type *Type*.

#### Parameters

- Type* the type of the objects to manage.  
*Slab\_size* size of a slab cache.  
*Max\_free* the maximum number of free slab caches.  
*Alloc* the allocator for the slab caches.

Definition at line 299 of file [slab\\_alloc](#).

### 11.174.2 Member Function Documentation

```
11.174.2.1 template<typename Type , int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc = New_allocator> Type* cxx::Slab< Type, Slab_size, Max_free, Alloc >::alloc ( ) throw () [inline]
```

Allocate an object of type *Type*.

#### Returns

A pointer to the object just allocated, or 0 on failure.

Reimplemented from `cxx::Base_slab< sizeof(Type), Slab_size, Max_free, Alloc >`.

Definition at line 316 of file [slab\\_alloc](#).

```
11.174.2.2 template<typename Type , int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc = New_allocator> void cxx::Slab< Type, Slab_size, Max_free, Alloc >::free ( Type * o ) throw () [inline]
```

Free the object addressed by *o*.

#### Parameters

- o* The pointer to the object to free.

### Precondition

The object must have been allocated with this allocator.

Definition at line 327 of file [slab\\_alloc](#).

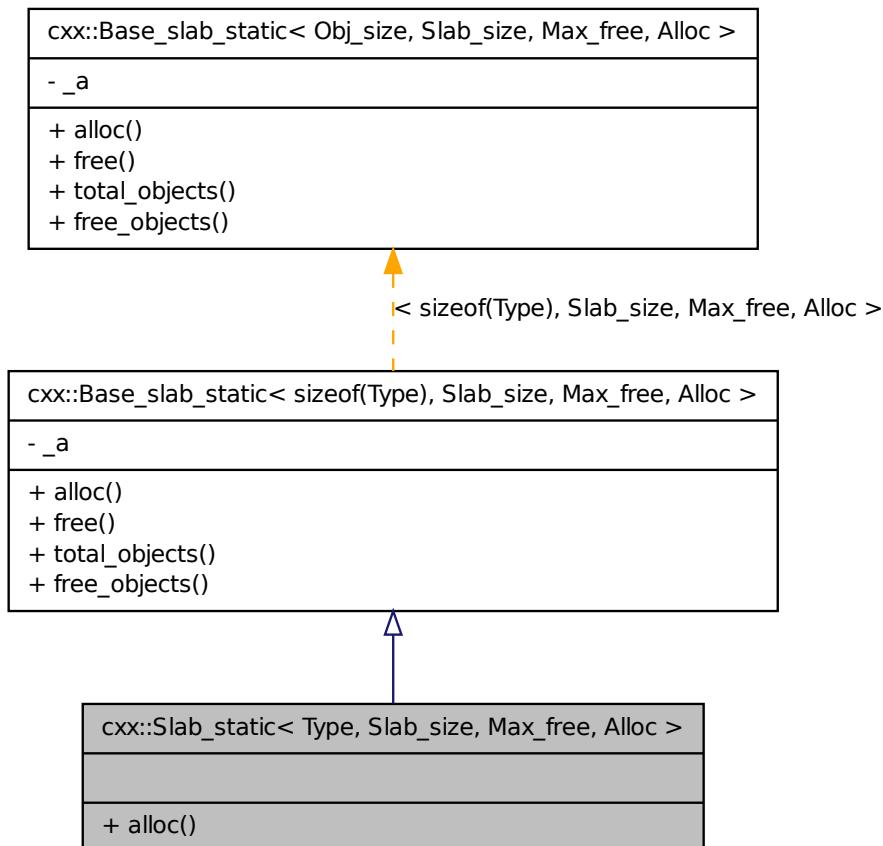
The documentation for this class was generated from the following file:

- 14/cxx/slab\_alloc

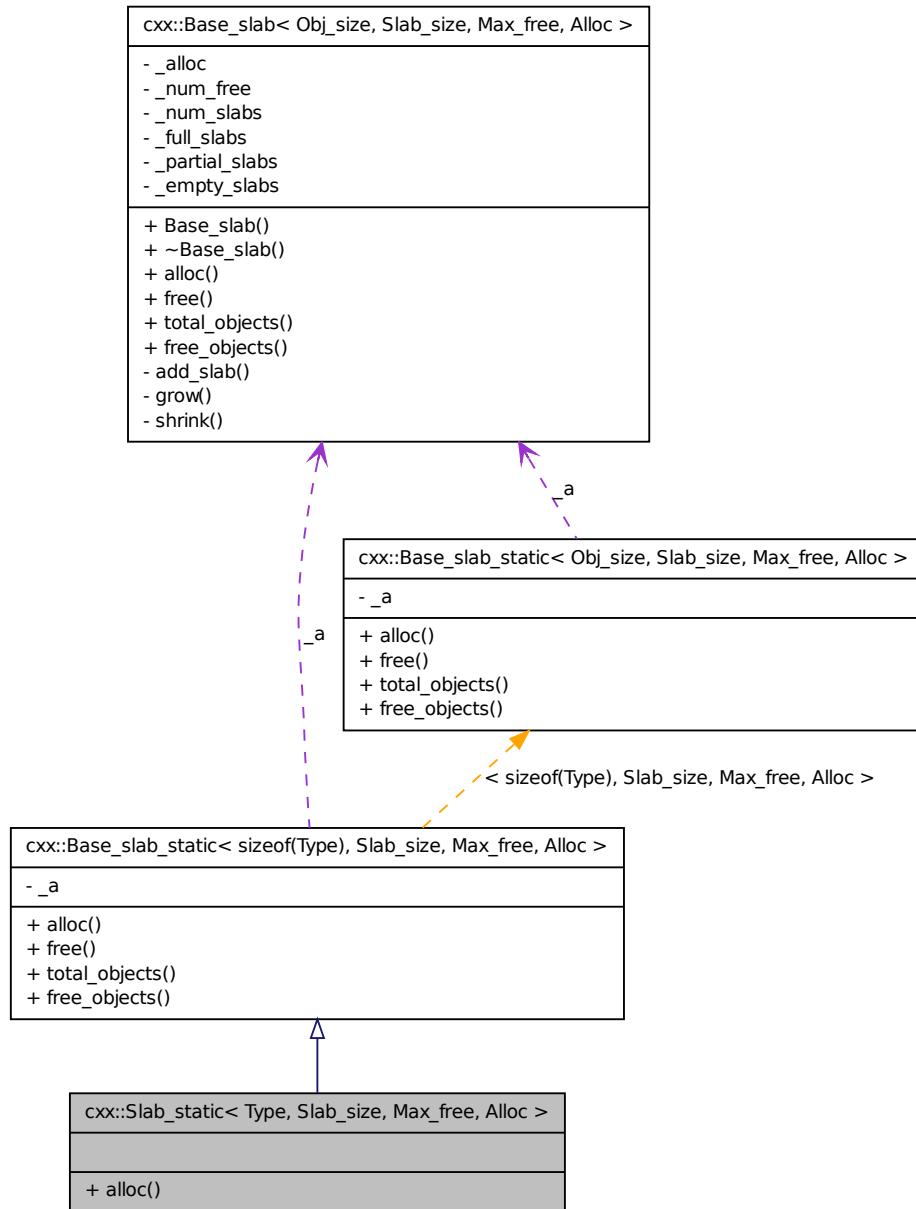
## 11.175 cxx::Slab\_static< Type, Slab\_size, Max\_free, Alloc > Class Template Reference

Merged slab allocator (allocators for objects of the same size are merged together).

Inheritance diagram for cxx::Slab\_static< Type, Slab\_size, Max\_free, Alloc >:



Collaboration diagram for cxx::Slab\_static< Type, Slab\_size, Max\_free, Alloc >:



## Public Member Functions

- `Type * alloc () throw ()`

*Allocate an object of type Type.*

### 11.175.1 Detailed Description

```
template<typename Type, int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc = New_allocator> class cxx::Slab_static< Type, Slab_size, Max_free, Alloc >
```

Merged slab allocator (allocators for objects of the same size are merged together).

#### Parameters

*Type* The type of the objects to manage.

*Slab\_size* The size of a slab cache.

*Max\_free* The maximum number of free slab caches.

*Alloc* The allocator for the slab caches.

This slab allocator class is useful for merging slab allocators with the same parameters (equal *sizeof(Type)*, *Slab\_size*, *Max\_free*, and *Alloc* parameters) together and share the overhead for the slab caches among all equal-sized objects.

Definition at line 417 of file [slab\\_alloc](#).

### 11.175.2 Member Function Documentation

```
11.175.2.1 template<typename Type , int Slab_size = L4_PAGESIZE, int Max_free = 2, template< typename A > class Alloc = New_allocator> Type* cxx::Slab_static< Type, Slab_size, Max_free, Alloc >::alloc( ) throw() [inline]
```

Allocate an object of type *Type*.

#### Returns

A pointer to the just allocated object, or 0 if failure.

Reimplemented from [cxx::Base\\_slab\\_static< sizeof\(Type\), Slab\\_size, Max\\_free, Alloc >](#).

Definition at line 427 of file [slab\\_alloc](#).

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

- l4/cxx/slab\_alloc

## 11.176 L4::Ipc::Small\_buf Class Reference

A receive item for receiving a single capability.

### 11.176.1 Detailed Description

A receive item for receiving a single capability. This class is the main abstraction for receiving capabilities via [Ipc::Istream](#). To receive a capability an instance of [Small\\_buf](#) that refers to an empty capability slot must be inserted into the [Ipc::Istream](#) before the receive operation.

Definition at line 256 of file [ipc\\_stream](#).

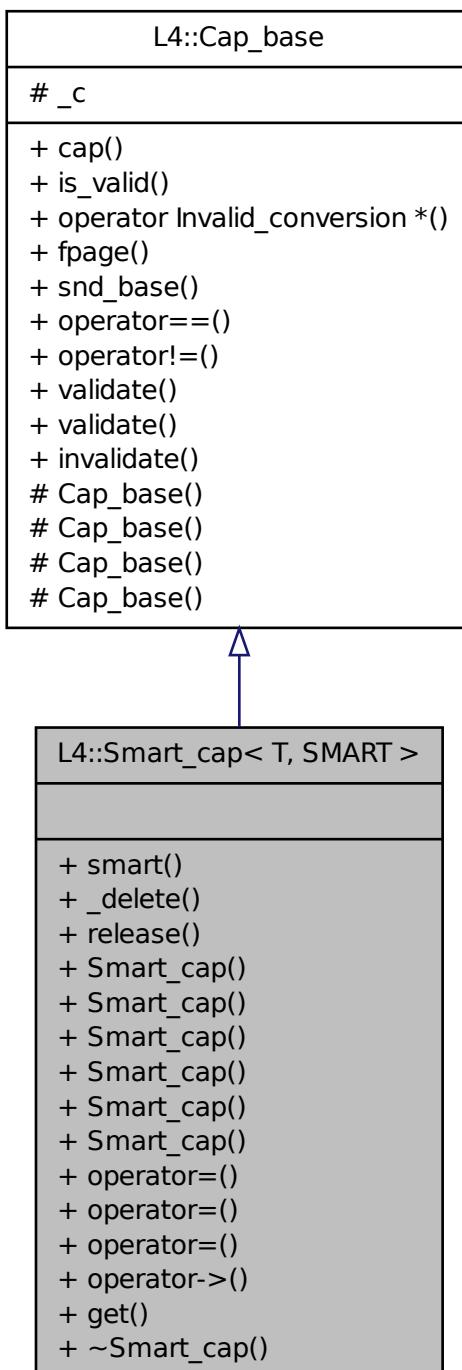
The documentation for this class was generated from the following file:

- l4/cxx/ipc\_stream

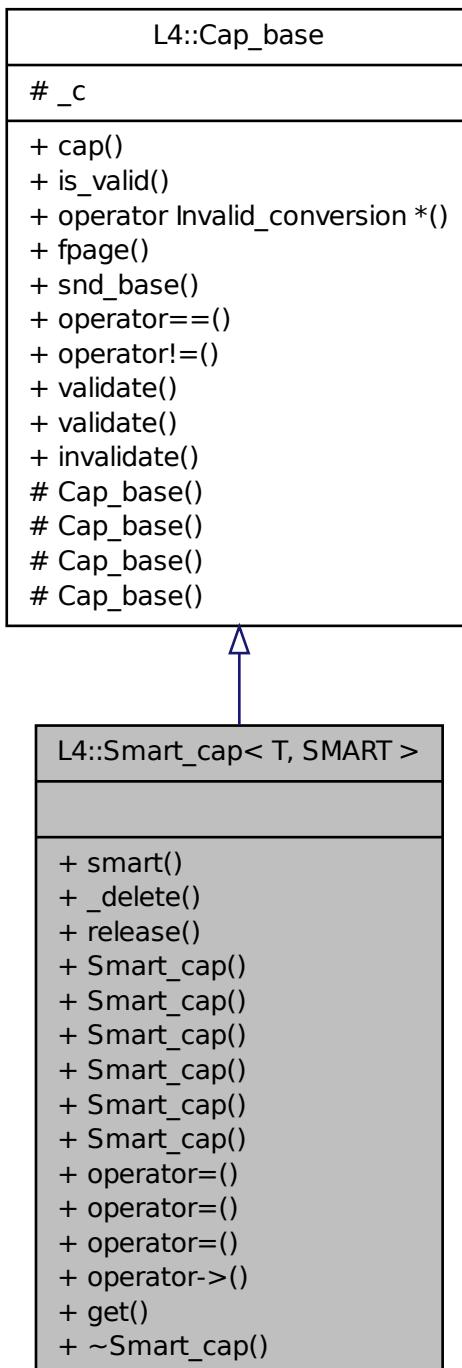
## 11.177 L4::Smart\_cap< T, SMART > Class Template Reference

Smart capability class.

Inheritance diagram for L4::Smart\_cap< T, SMART >:



Collaboration diagram for L4::Smart\_cap< T, SMART >:



## Public Member Functions

- template<typename O >  
**Smart\_cap** (**Cap**< O > const &p) throw ()

*Internal Constructor, use to generate a capability from a this pointer.*

- **Cap**< T > **operator->** () const throw ()

*Member access of a T.*

### 11.177.1 Detailed Description

**template<typename T, typename SMART> class L4::Smart\_cap< T, SMART >**

Smart capability class.

Definition at line 36 of file [smart\\_capability](#).

### 11.177.2 Constructor & Destructor Documentation

**11.177.2.1 template<typename T, typename SMART> template<typename O > L4::Smart\_cap< T, SMART >::Smart\_cap ( Cap< O > const & p ) throw () [inline]**

Internal Constructor, use to generate a capability from a *this* pointer.

#### Attention

This constructor is only useful to generate a capability from the *this* pointer of an objected that is an [L4::Kobject](#). Do *never* use this constructor for something else!

#### Parameters

**p** The *this* pointer of the [Kobject](#) or derived object

Definition at line 67 of file [smart\\_capability](#).

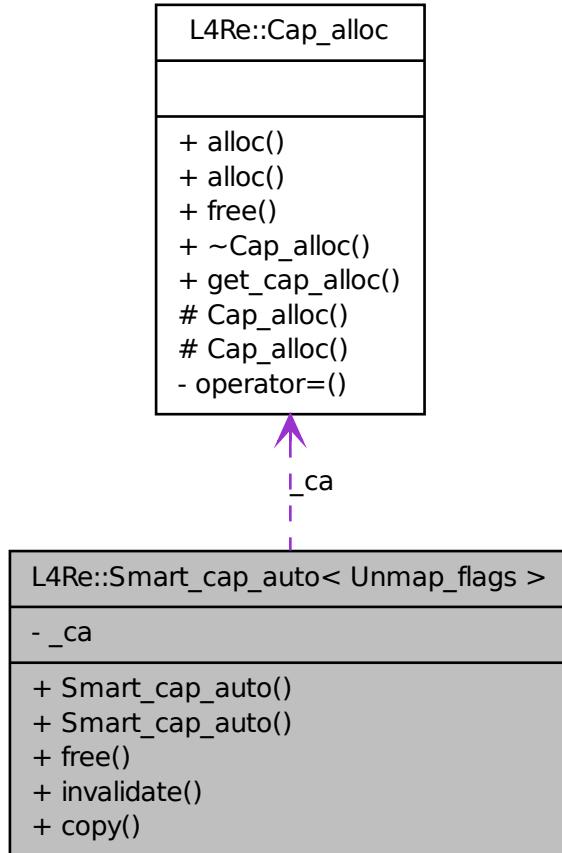
The documentation for this class was generated from the following file:

- l4/sys/smart\_capability

## 11.178 L4Re::Smart\_cap\_auto< Unmap\_flags > Class Template Reference

Helper for Auto\_cap and Auto\_del\_cap.

Collaboration diagram for L4Re::Smart\_cap\_auto< Unmap\_flags >:



### 11.178.1 Detailed Description

```
template<unsigned long Unmap_flags = L4_FP_ALL_SPACES> class L4Re::Smart_cap_auto< Unmap_flags >
```

Helper for Auto\_cap and Auto\_del\_cap.

Definition at line 109 of file [cap\\_alloc](#).

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

- [l4/re/cap\\_alloc](#)

## 11.179 L4Re::Util::Smart\_cap\_auto< Unmap\_flags > Class Template Reference

Helper for [Auto\\_cap](#) and [Auto\\_del\\_cap](#).

### Static Public Member Functions

- static void [free \(L4::Cap\\_base &c\)](#)  
*free operation for L4::Smart\_cap.*
- static void [invalidate \(L4::Cap\\_base &c\)](#)  
*invalidate operation for L4::Smart\_cap.*
- static [L4::Cap\\_base copy \(L4::Cap\\_base const &src\)](#)  
*copy operation for L4::Smart\_cap.*

### 11.179.1 Detailed Description

```
template<unsigned long Unmap_flags = L4_FP_ALL_SPACES> class L4Re::Util::Smart_cap_
auto< Unmap_flags >
```

Helper for [Auto\\_cap](#) and [Auto\\_del\\_cap](#).

Definition at line 54 of file [cap\\_alloc](#).

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

- l4/re/util/cap\_alloc

## 11.180 L4Re::Util::Smart\_count\_cap< Unmap\_flags > Class Template Reference

Helper for [Ref\\_cap](#) and [Ref\\_del\\_cap](#).

### Static Public Member Functions

- static void [free \(L4::Cap\\_base &c\)](#)  
*free operation for L4::Smart\_cap (decrement ref count and delete if 0).*
- static void [invalidate \(L4::Cap\\_base &c\)](#)  
*invalidate operation for L4::Smart\_cap.*
- static [L4::Cap\\_base copy \(L4::Cap\\_base const &src\)](#)  
*copy operation for L4::Smart\_cap (increment ref count).*

### 11.180.1 Detailed Description

```
template<unsigned long Unmap_flags = L4_FP_ALL_SPACES> class L4Re::Util::Smart_count_-  
cap< Unmap_flags >
```

Helper for [Ref\\_cap](#) and [Ref\\_del\\_cap](#).

Definition at line [94](#) of file [cap\\_alloc](#).

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

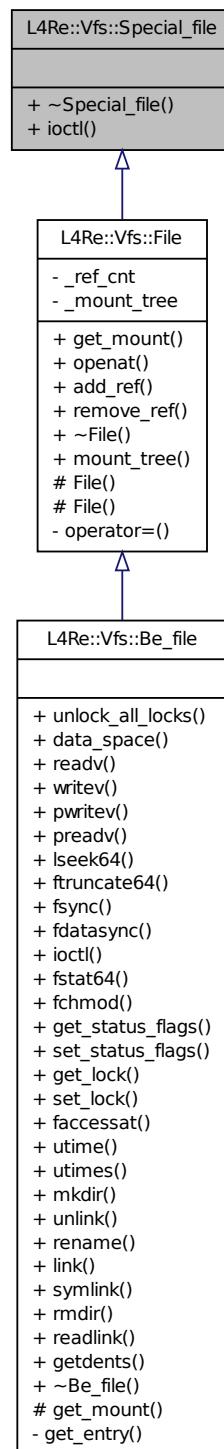
- [l4/re/util/cap\\_alloc](#)

## 11.181 L4Re::Vfs::Special\_file Class Reference

Interface for a POSIX file that provides special file semantics.

```
#include <vfs.h>
```

Inheritance diagram for L4Re::Vfs::Special\_file:



## Public Member Functions

- virtual int `ioctl` (unsigned long cmd, va\_list args)=0 throw ()

*The famous IO control.*

### 11.181.1 Detailed Description

Interface for a POSIX file that provides special file semantics. Real objects use always the combined `L4Re::Vfs::File` interface.

Definition at line 368 of file `vfs.h`.

### 11.181.2 Member Function Documentation

#### 11.181.2.1 virtual int L4Re::Vfs::Special\_file::ioctl ( unsigned long cmd, va\_list args ) throw () [pure virtual]

The famous IO control.

Backend for POSIX generic object invocation ioctl.

##### Parameters

*cmd* The ioctl command.

*args* The arguments for the ioctl, usually some kind of pointer.

##### Returns

$\geq 0$  on success, or  $< 0$  on error.

Implemented in `L4Re::Vfs::Be_file`.

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

- l4/l4re\_vfs/vfs.h

## 11.182 L4vcpu::State Class Reference

C++ implementation of state word in the vCPU area.

## Public Member Functions

- `State` (l4vcpu\_state\_t v)

*Initialize state.*

- void `add` (unsigned bits) throw ()

*Add flags.*

- void `clear` (unsigned bits) throw ()

*Clear flags.*

- void [set](#) (l4vcpu\_state\_t v) throw ()

*Set flags.*

### 11.182.1 Detailed Description

C++ implementation of state word in the vCPU area.

Definition at line [30](#) of file [vcpu](#).

### 11.182.2 Constructor & Destructor Documentation

#### 11.182.2.1 L4vcpu::State::State ( l4vcpu\_state\_t v ) [inline, explicit]

Initialize state.

##### Parameters

*v* Initial state.

Definition at line [33](#) of file [vcpu](#).

### 11.182.3 Member Function Documentation

#### 11.182.3.1 void L4vcpu::State::add ( unsigned bits ) throw () [inline]

Add flags.

##### Parameters

*bits* Bits to add to the word.

Definition at line [40](#) of file [vcpu](#).

#### 11.182.3.2 void L4vcpu::State::clear ( unsigned bits ) throw () [inline]

Clear flags.

##### Parameters

*bits* Bits to clear in the word.

Definition at line [47](#) of file [vcpu](#).

#### 11.182.3.3 void L4vcpu::State::set ( l4vcpu\_state\_t v ) throw () [inline]

Set flags.

### Parameters

- *v* Set the word to the value of v.

Definition at line [54](#) of file [vcpu](#).

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

- [l4/vcpu/vcpu](#)

## 11.183 L4Re::Dataspace::Stats Struct Reference

Information about the data space.

### Data Fields

- unsigned long **size**  
*size*
- unsigned long **flags**  
*flags*

### 11.183.1 Detailed Description

Information about the data space.

Definition at line [93](#) of file [dataspace](#).

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

- [l4/re/dataspace](#)

## 11.184 L4::String Class Reference

A null-terminated string container class.

```
#include <string.h>
```

### 11.184.1 Detailed Description

A null-terminated string container class.

Definition at line [35](#) of file [string.h](#).

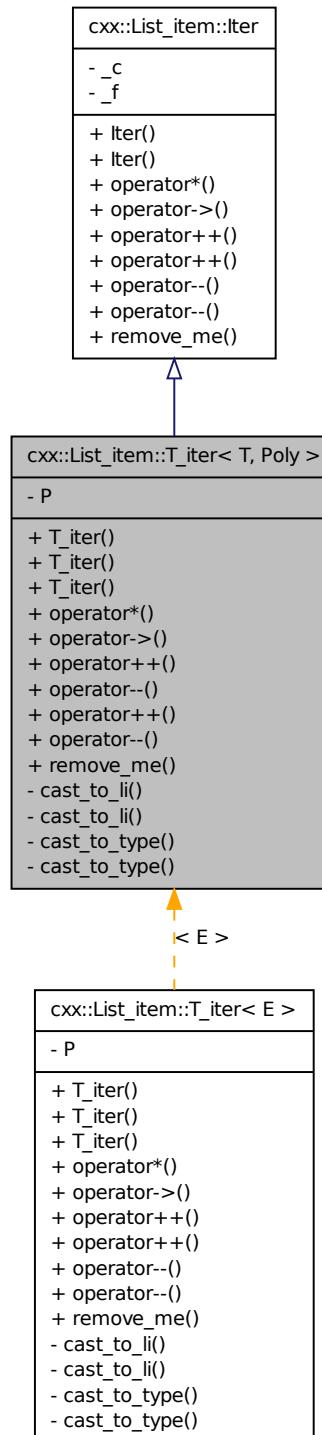
The documentation for this class was generated from the following file:

- [l4/cxx/string.h](#)

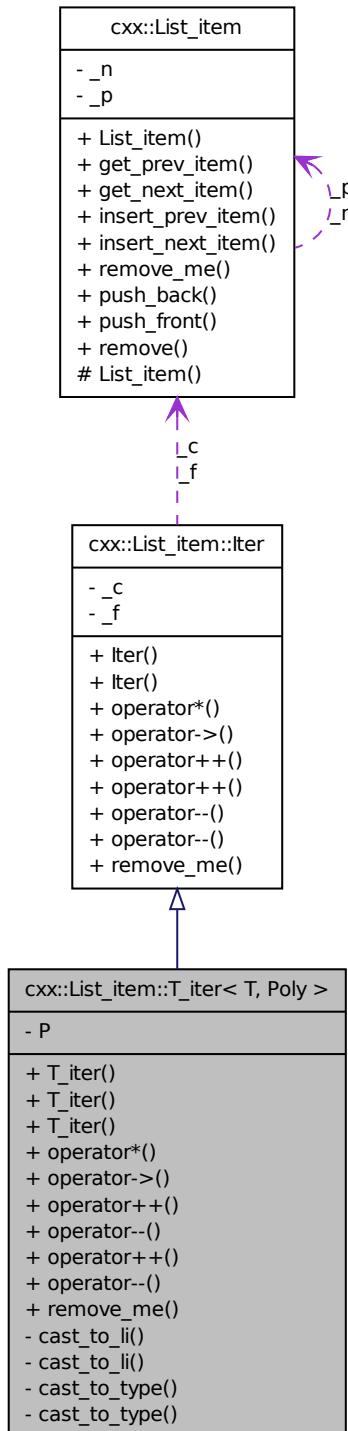
## 11.185 cxx::List\_item::T\_iter< T, Poly > Class Template Reference

Iterator for derived classes from ListItem.

Inheritance diagram for cxx::List\_item::T\_iter< T, Poly >:



Collaboration diagram for cxx::List\_item::T\_iter< T, Poly >:



## Public Member Functions

- `T * remove_me () throw ()`

*Remove item pointed to by iterator, and return pointer to element.*

### 11.185.1 Detailed Description

`template<typename T, bool Poly = false> class cxx::List_item::T_iter< T, Poly >`

Iterator for derived classes from ListItem. Allows direct access to derived classes by \* operator.

Example: class Foo : public ListItem { public: typedef T\_iter<Foo> Iter; ... };

Definition at line 119 of file [list](#).

### 11.185.2 Member Function Documentation

#### 11.185.2.1 `template<typename T, bool Poly> T * cxx::List_item::T_iter< T, Poly >::remove_me ( ) throw () [inline]`

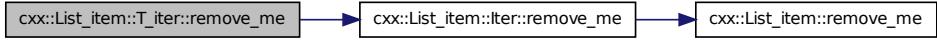
Remove item pointed to by iterator, and return pointer to element.

Reimplemented from [cxx::List\\_item::Iter](#).

Definition at line 290 of file [list](#).

References [cxx::List\\_item::Iter::remove\\_me\(\)](#).

Here is the call graph for this function:



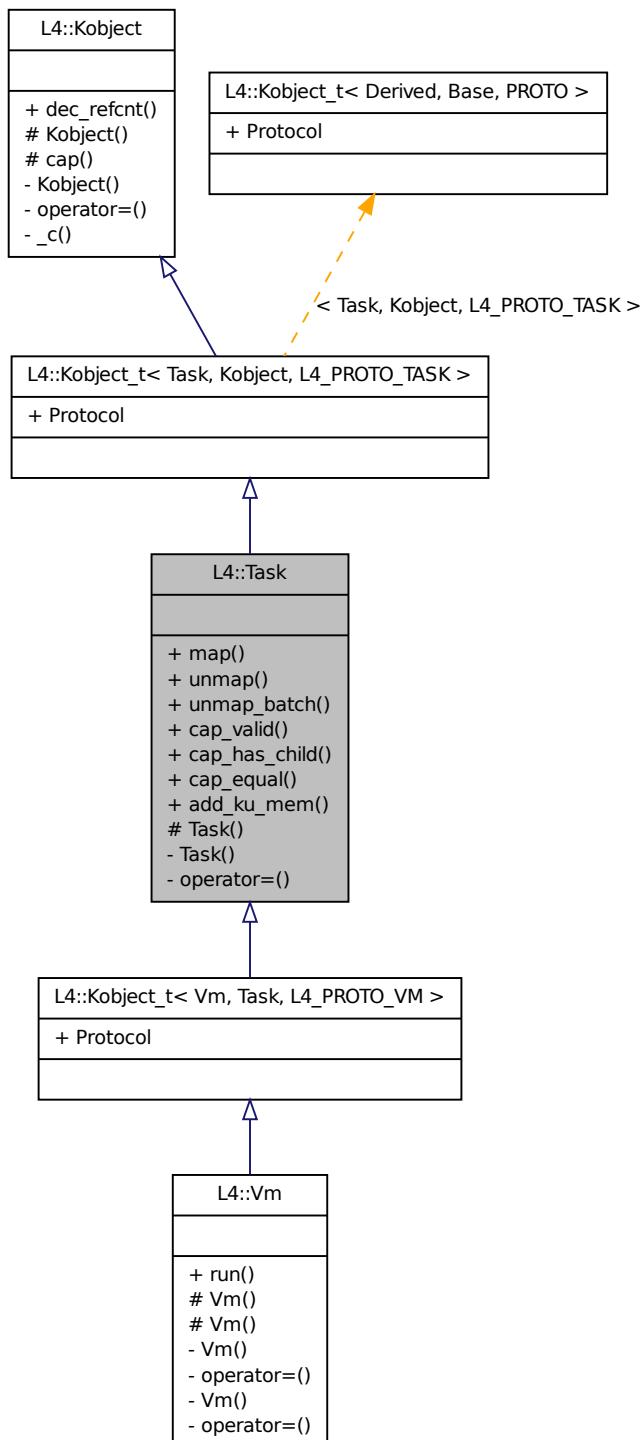
The documentation for this class was generated from the following file:

- [l4/cxx/list](#)

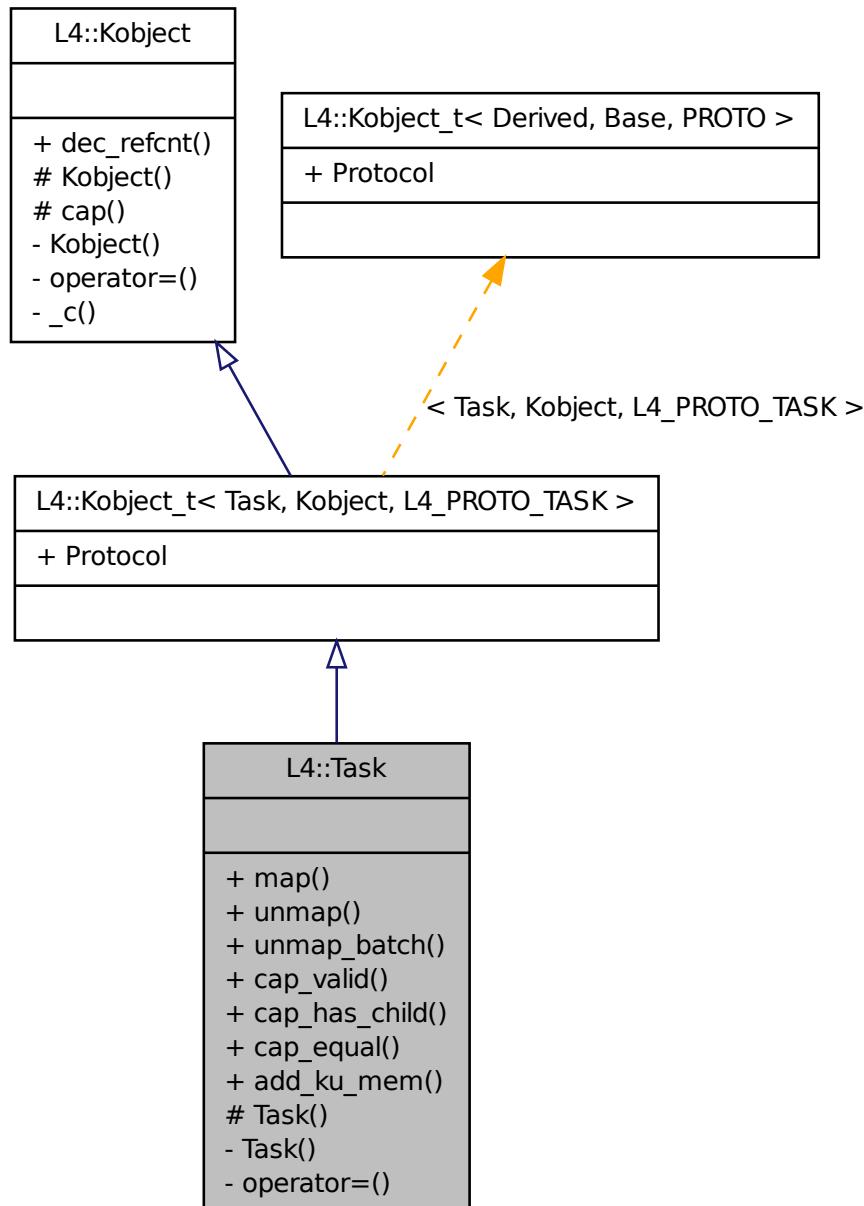
## 11.186 L4::Task Class Reference

An [L4 Task](#).

Inheritance diagram for L4::Task:



Collaboration diagram for L4::Task:



## Public Member Functions

- `l4_msgtag_t map (Cap< Task > const &src_task, l4_fpage_t const &snd_fpage, l4_addr_t snd_base, l4_utcb_t *utcb=l4_utcb()) throw ()`

- `l4_mshtag_t unmap (l4_fpage_t const &fpage, l4_umword_t map_mask, l4_utcb_t *utcb=l4_utcb()) throw ()`
- `l4_mshtag_t unmap_batch (l4_fpage_t const *fpages, unsigned num_fpages, l4_umword_t map_mask, l4_utcb_t *utcb=l4_utcb()) throw ()`
- `l4_mshtag_t cap_valid (Cap< void > const &cap, l4_utcb_t *utcb=l4_utcb()) throw ()`
- `l4_mshtag_t cap_has_child (Cap< void > const &cap, l4_utcb_t *utcb=l4_utcb()) throw ()`
- `l4_mshtag_t cap_equal (Cap< void > const &cap_a, Cap< void > const &cap_b, l4_utcb_t *utcb=l4_utcb()) throw ()`
- `l4_mshtag_t add_ku_mem (l4_fpage_t const &fpage, l4_utcb_t *utcb=l4_utcb()) throw ()`

## 11.186.1 Detailed Description

An [L4 Task](#). #include <l4/sys/task>

### See also

[Task](#) for an overview description.

Definition at line 41 of file [task](#).

## 11.186.2 Member Function Documentation

### 11.186.2.1 `l4_mshtag_t L4::Task::map ( Cap< Task > const & src_task, l4_fpage_t const & snd_fpage, l4_addr_t snd_base, l4_utcb_t * utcb = l4_utcb() ) throw () [inline]`

Map resources available in the source task to a destination task.

#### Parameters

`dst_task` Capability selector of destination task

`src_task` Capability selector of source task

`snd_fpage` Send flexpage that describes an area in the address space or object space of the source task

`snd_base` Send base that describes an offset in the receive window of the destination task.

#### Returns

Syscall return tag

This method allows for asynchronous rights delegation from one task to another. It can be used to share memory as well as to delegate access to objects.

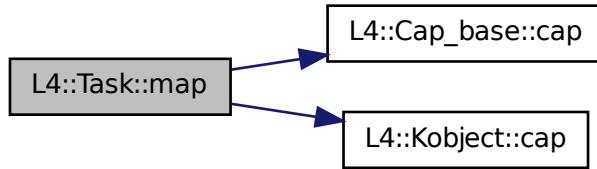
#### Note

`dst_task` is the implicit *this* pointer.

Definition at line 50 of file [task](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 11.186.2.2 `l4_mshtag_t L4::Task::unmap ( l4_fpage_t const & fpage, l4_umword_t map_mask, l4_utcb_t * utcb = 14_utcb () ) throw () [inline]`

Revoke rights from the task.

#### Parameters

*task* Capability selector of destination task

*fpage* Flexpage that describes an area in the address space or object space of the destination task

*map\_mask* Unmap mask, see [l4\\_unmap\\_flags\\_t](#)

#### Returns

Syscall return tag

This method allows to revoke rights from the destination task and from all the tasks that got the rights delegated from that task (i.e., this operation does a recursive rights revocation).

#### Note

Calling this function on the object space can cause a root capability of an object to be destructed, which destroys the object itself.

#### Note

*task* is the implicit *this* pointer.

Definition at line 59 of file [task](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



**11.186.2.3 l4\_mshtag\_t L4::Task::unmap\_batch ( l4\_fpage\_t const \* *fpages*, unsigned *num\_fpages*, l4\_umword\_t *map\_mask*, l4\_utcb\_t \* *utcb* = *l4\_utcb()* ) throw () [inline]**

Revoke rights from a task.

#### Parameters

*task* Capability selector of destination task

*fpages* An array of flexpages that describes an area in the address space or object space of the destination task each

*num\_fpages* The size of the fpages array in elements (number of fpages sent).

*map\_mask* Unmap mask, see [l4\\_unmap\\_flags\\_t](#)

#### Returns

Syscall return tag

This method allows to revoke rights from the destination task and from all the tasks that got the rights delegated from that task (i.e., this operation does a recursive rights revocation).

#### Precondition

The caller needs to take care that num\_fpages is not bigger than L4\_UTCB\_GENERIC\_DATA\_SIZE - 2.

#### Note

Calling this function on the object space can cause a root capability of an object to be destructed, which destroys the object itself.

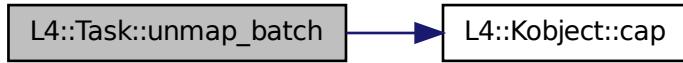
#### Note

*task* is the implicit *this* pointer.

Definition at line 68 of file [task](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



#### **11.186.2.4 l4\_mshtag\_t L4::Task::cap\_valid ( Cap< void > const & cap, l4\_utcb\_t \* utcb =   l4\_utcb() ) throw () [inline]**

Test whether a capability selector points to a valid capability.

#### **Parameters**

*task* Capability selector of the destination task to do the lookup in  
*cap* Capability selector to look up in the destination task

#### **Returns**

label contains 1 if valid, 0 if invalid

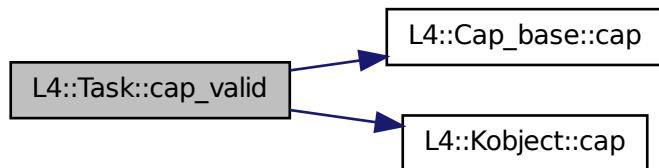
#### **Note**

*task* is the implicit *this* pointer.

Definition at line 78 of file [task](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



**11.186.2.5 l4\_mshtag\_t L4::Task::cap\_has\_child ( Cap< void > const & *cap*, l4\_utcb\_t \* *utcb* = *l4\_utcb()* ) throw () [inline]**

Test whether a capability has child mappings (in another task).

#### Parameters

*task* Capability selector of the destination task to do the lookup in  
*cap* Capability selector to look up in the destination task

#### Returns

label contains 1 if it has at least one child, 0 if not or invalid

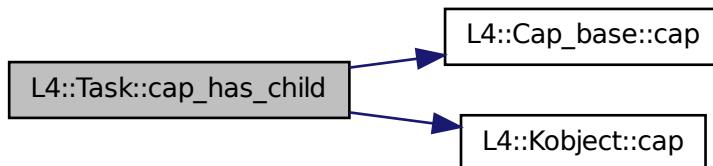
#### Note

*task* is the implicit *this* pointer.

Definition at line 86 of file [task](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



**11.186.2.6 l4\_mshtag\_t L4::Task::cap\_equal ( Cap< void > const & *cap\_a*, Cap< void > const & *cap\_b*, l4\_utcb\_t \* *utcb* = *l4\_utcb()* ) throw () [inline]**

Test if two capabilities point to the same object.

#### Parameters

*task* Capability selector of the destination task to do the lookup in  
*cap\_a* Capability selector to compare  
*cap\_b* Capability selector to compare

#### Returns

label contains 1 if equal, 0 if not equal

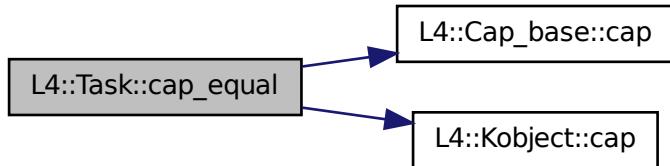
#### Note

*task* is the implicit *this* pointer.

Definition at line 94 of file [task](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



**11.186.2.7** `l4_mshtag_t L4::Task::add_ku_mem ( l4_fpage_t const & fpage, l4_utcb_t * utcb = 14_utcb() ) throw () [inline]`

Add kernel-user memory.

#### Parameters

*task* Capability selector of the task to add the memory to

*ku\_mem* Flexpage describing the virtual area the memory goes to.

#### Returns

Syscall return tag

#### Note

*task* is the implicit *this* pointer.

Definition at line 103 of file [task](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



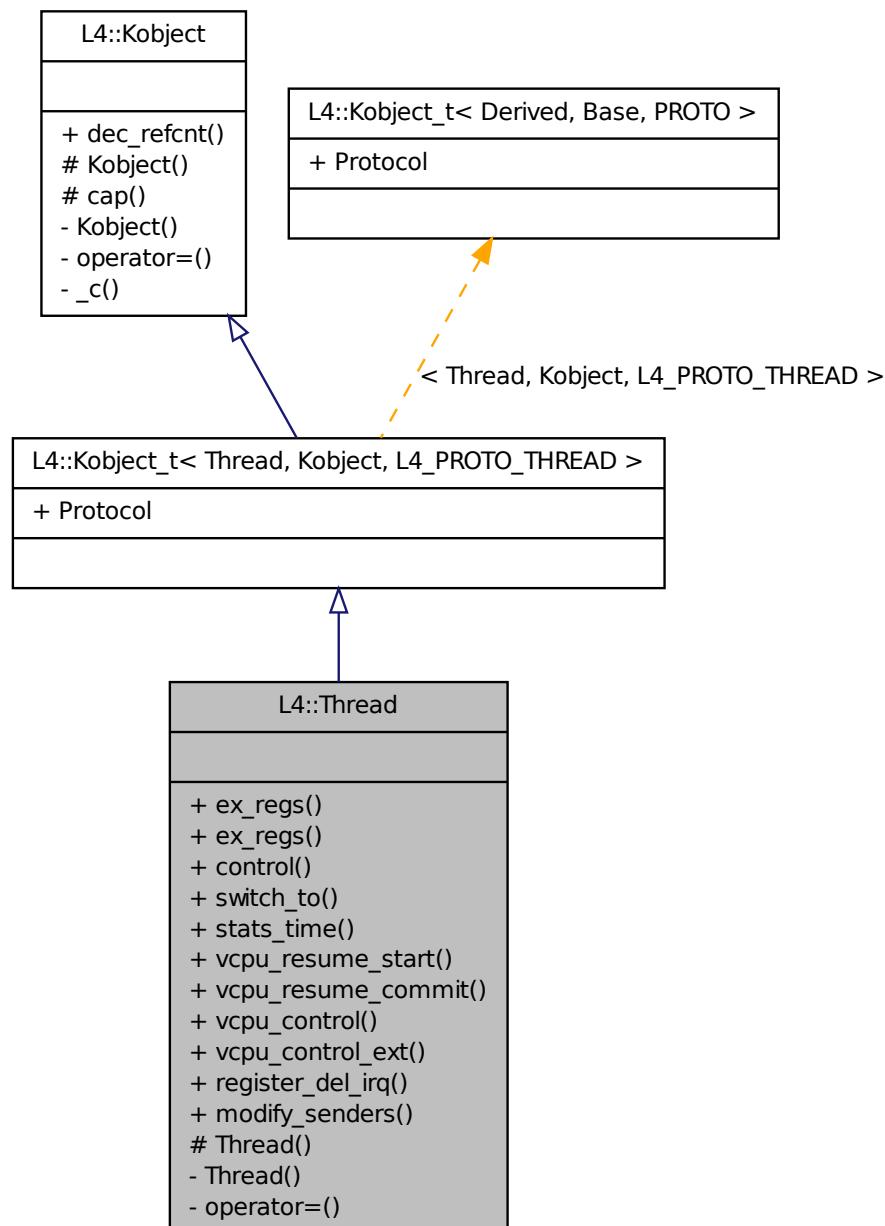
The documentation for this class was generated from the following file:

- l4/sys/task

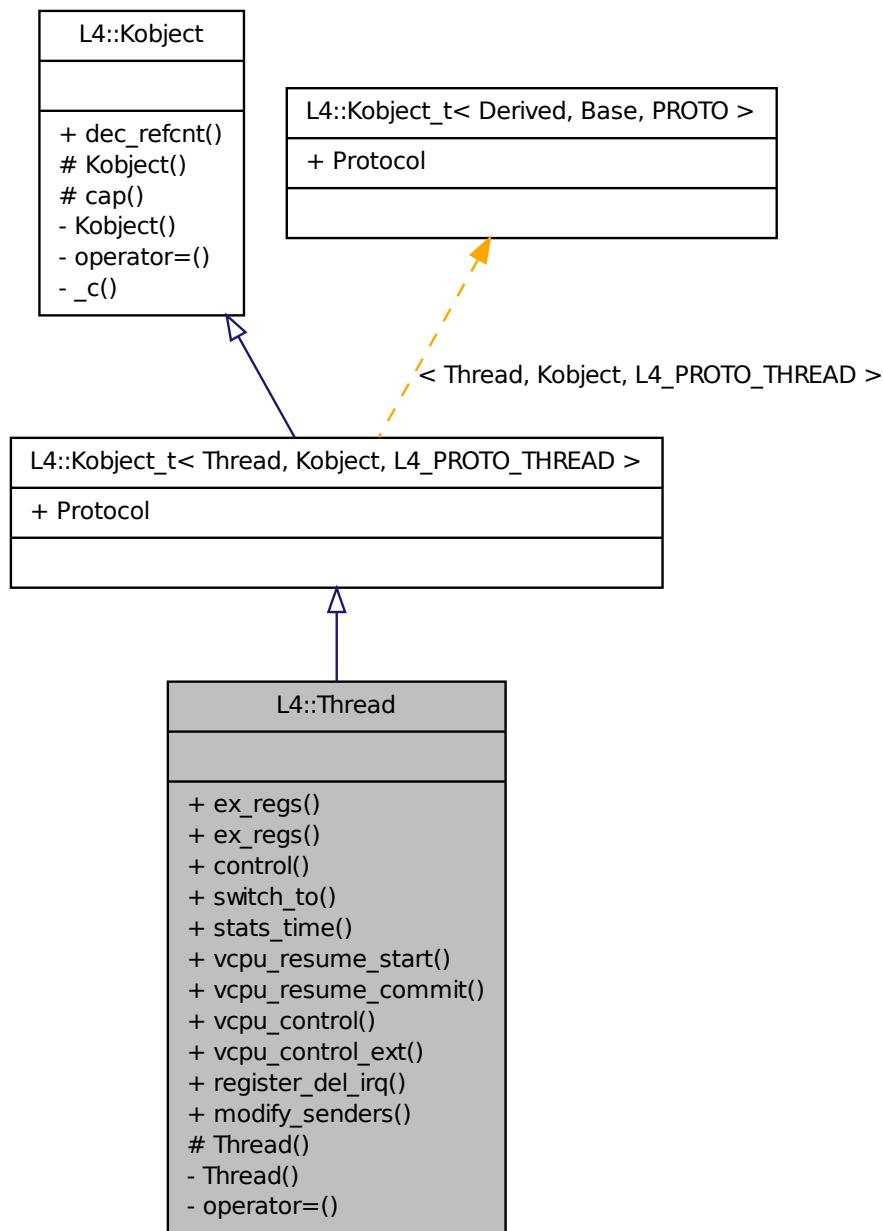
## **11.187 L4::Thread Class Reference**

[L4](#) kernel thread.

Inheritance diagram for L4::Thread:



Collaboration diagram for L4::Thread:



## Data Structures

- class [Attr](#)

*Thread* attributes used for `control_commit()`.

- class [Modify\\_senders](#)  
*Wrapper class for modifying senders.*

## Public Member Functions

- [l4\\_mshtag\\_t ex\\_regs \(l4\\_addr\\_t ip, l4\\_addr\\_t sp, l4\\_umword\\_t flags, l4\\_utcb\\_t \\*utcb=l4\\_utcb\(\)\) throw \(\)](#)  
*l4\_mshtag\_t ex\_regs (l4\_addr\_t \*ip, l4\_addr\_t \*sp, l4\_umword\_t \*flags, l4\_utcb\_t \*utcb=l4\_utcb()) throw ()*
- [l4\\_mshtag\\_t control \(Attr const &attr\) throw \(\)](#)  
*Commit the given thread-attributes object.*
- [l4\\_mshtag\\_t switch\\_to \(l4\\_utcb\\_t \\*utcb=l4\\_utcb\(\)\) throw \(\)](#)  
*Switch execution to this thread.*
- [l4\\_mshtag\\_t stats\\_time \(l4\\_utcb\\_t \\*utcb=l4\\_utcb\(\)\) throw \(\)](#)  
*Get consumed timed of thread in ns.*
- [l4\\_mshtag\\_t vcpu\\_resume\\_start \(l4\\_utcb\\_t \\*utcb=l4\\_utcb\(\)\) throw \(\)](#)  
*vCPU resume, start.*
- [l4\\_mshtag\\_t vcpu\\_resume\\_commit \(l4\\_mshtag\\_t tag, l4\\_utcb\\_t \\*utcb=l4\\_utcb\(\)\) throw \(\)](#)  
*vCPU resume, commit.*
- [l4\\_mshtag\\_t vcpu\\_control \(l4\\_addr\\_t vcpu\\_state, l4\\_utcb\\_t \\*utcb=l4\\_utcb\(\)\) throw \(\)](#)
- [l4\\_mshtag\\_t vcpu\\_control\\_ext \(l4\\_addr\\_t ext\\_vcpu\\_state, l4\\_utcb\\_t \\*utcb=l4\\_utcb\(\)\) throw \(\)](#)
- [l4\\_mshtag\\_t register\\_del\\_irq \(Cap< Irq > irq, l4\\_utcb\\_t \\*u=l4\\_utcb\(\)\) throw \(\)](#)  
*Register an IRQ that will trigger upon deletion events.*
- [l4\\_mshtag\\_t modify\\_senders \(Modify\\_senders const &todo\) throw \(\)](#)  
*Apply sender modifiction rules.*

### 11.187.1 Detailed Description

L4 kernel thread. #include <l4/sys/thread>

Definition at line 38 of file [thread](#).

### 11.187.2 Member Function Documentation

#### 11.187.2.1 l4\_mshtag\_t L4::Thread::ex\_regs ( l4\_addr\_t ip, l4\_addr\_t sp, l4\_umword\_t flags, l4\_utcb\_t \* utcb = l4\_utcb() ) throw () [inline]

Exchange basic thread registers.

##### Parameters

*thread* Thread to manipulate

*ip* New instruction pointer, use ~0UL to leave the instruction pointer unchanged

*sp* New stack pointer, use ~0UL to leave the stack pointer unchanged

*flags* Ex-reg flags, see [L4\\_thread\\_ex\\_regs\\_flags](#)

### Returns

System call return tag

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see *flags*).

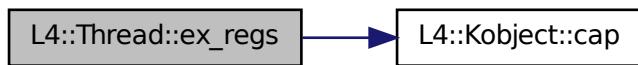
### Note

the *thread* argument is the implicit *this* pointer.

Definition at line 47 of file [thread](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



## 11.187.2.2 l4\_mshtag\_t L4::Thread::ex\_regs ( l4\_addr\_t \* ip, l4\_addr\_t \* sp, l4\_umword\_t \* flags, l4\_utcb\_t \* utcb = 14\_utcb() ) throw () [inline]

Exchange basic thread registers and return previous values.

### Parameters

*thread* Thread to manipulate

### Return values

*ip* New instruction pointer, use ~0UL to leave the instruction pointer unchanged, return previous instruction pointer

*sp* New stack pointer, use ~0UL to leave the stack pointer unchanged, returns previous stack pointer

*flags* Ex-reg flags, see [L4\\_thread\\_ex\\_regs\\_flags](#), return previous CPU flags of the thread.

### Returns

System call return tag

This method allows to manipulate and start a thread. The basic functionality is to set the instruction pointer and the stack pointer of a thread. Additionally, this method allows also to cancel ongoing IPC operations and to force the thread to raise an artificial exception (see *flags*).

Returned values are valid only if function returns successfully.

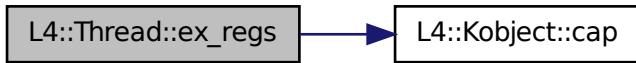
#### Note

the *thread* argument is the implicit *this* pointer.

Definition at line 56 of file [thread](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### **11.187.2.3 l4\_mshtag\_t L4::Thread::control ( Attr const & attr ) throw () [inline]**

Commit the given thread-attributes object.

#### Parameters

*attr* the attribute object to commit to the thread.

Definition at line 152 of file [thread](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### **11.187.2.4 l4\_mshtag\_t L4::Thread::switch\_to ( l4\_utcb\_t \* utcb = l4\_utcb() ) throw () [inline]**

Switch execution to this thread.

**Parameters**

*utcb* the UTCB of the current thread.

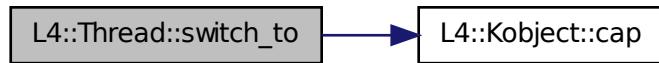
**Note**

The current time slice is inherited to this thread.

Definition at line 161 of file [thread](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 11.187.2.5 l4\_mshtag\_t L4::Thread::stats\_time ( l4\_utcb\_t \* utcb = [l4\\_utcb\(\)](#) ) throw () [inline]

Get consumed timed of thread in ns.

**Parameters**

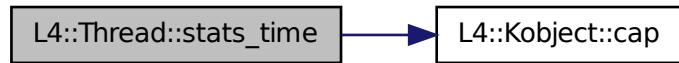
*utcb* the UTCB of the current thread.

The consumed time is return as l4\_kernel\_clock\_t at UTCB message register 0.

Definition at line 171 of file [thread](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 11.187.2.6 l4\_mshtag\_t L4::Thread::vcpu\_resume\_start ( l4\_utcb\_t \* utcb = [l4\\_utcb\(\)](#) ) throw () [inline]

vCPU resume, start.

**See also**

[l4\\_thread\\_vcpu\\_resume\\_start](#)

Definition at line 179 of file [thread](#).

**11.187.2.7 l4\_mshtag\_t L4::Thread::vcpu\_resume\_commit ( l4\_mshtag\_t tag, l4\_utcb\_t \* utcb =  
  l4\_utcb() )throw () [inline]**

vCPU resume, commit.

**See also**

[l4\\_thread\\_vcpu\\_resume\\_commit](#)

Definition at line 187 of file [thread](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



**11.187.2.8 l4\_mshtag\_t L4::Thread::vcpu\_control ( l4\_addr\_t vcpu\_state, l4\_utcb\_t \* utcb =  
  l4\_utcb() )throw () [inline]**

Enable or disable the vCPU feature for the thread.

**Parameters**

**thread** The thread for which the vCPU feature shall be enabled or disabled.

**vcpu\_state** The virtual address where the kernel shall store the vCPU state in case of vCPU exits. The address must be a valid kernel-user-memory address.

**Returns**

Systemcall result message tag.

This function enables the vCPU feature of the *thread* if *vcpu\_state* is set to a valid kernel-user-memory address, or disables the vCPU feature if *vcpu\_state* is 0.

Definition at line 194 of file [thread](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



**11.187.2.9** `l4_mshtag_t L4::Thread::vcpu_control_ext( l4_addr_t ext_vcpu_state, l4_utcb_t * utcb = l4_utcb() ) throw() [inline]`

Definition at line 201 of file [thread](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



**11.187.2.10** `l4_mshtag_t L4::Thread::register_del_irq( Cap< Irq > irq, l4_utcb_t * u = l4_utcb() ) throw() [inline]`

Register an IRQ that will trigger upon deletion events.

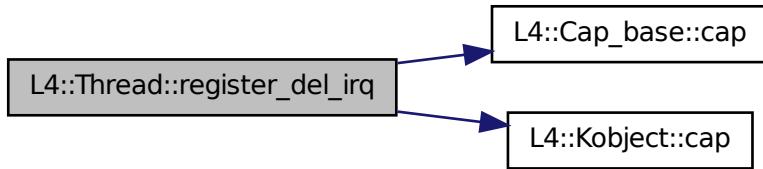
#### See also

[l4\\_thread\\_register\\_del\\_irq](#)

Definition at line 210 of file [thread](#).

References [L4::Cap\\_base::cap\(\)](#), and [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 11.187.2.11 l4\_mshtag\_t L4::Thread::modify\_senders ( *Modify\_senders const & todo* ) throw () [inline]

Apply sender modification rules.

#### Parameters

*todo* Prepared sender modification rules.

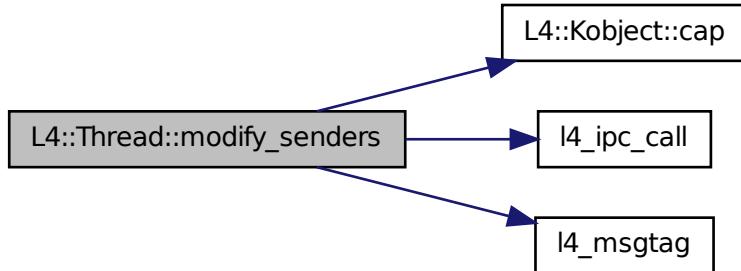
#### Returns

System call return tag.

Definition at line 270 of file [thread](#).

References [L4::Kobject::cap\(\)](#), [l4\\_ipc\\_call\(\)](#), [L4\\_IPC\\_NEVER](#), [l4\\_mshtag\(\)](#), and [L4\\_PROTO\\_THREAD](#).

Here is the call graph for this function:



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

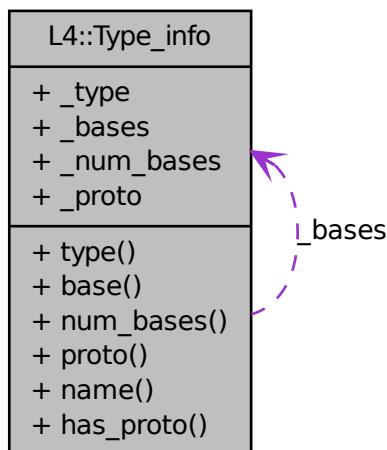
- [l4/sys/thread](#)

## 11.188 L4::Type\_info Struct Reference

Dynamic Type Information for L4Re Interfaces.

```
#include <__typeinfo.h>
```

Collaboration diagram for L4::Type\_info:



### 11.188.1 Detailed Description

Dynamic Type Information for L4Re Interfaces. This class represents the runtime-dynamic type information for L4Re interfaces, and is not intended to be used directly by applications.

#### Note

The interface of is subject to changes.

The main use for this info is to be used by the implementation of the L4::cap\_dynamic\_cast() function.

Definition at line 50 of file \_\_typeinfo.h.

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

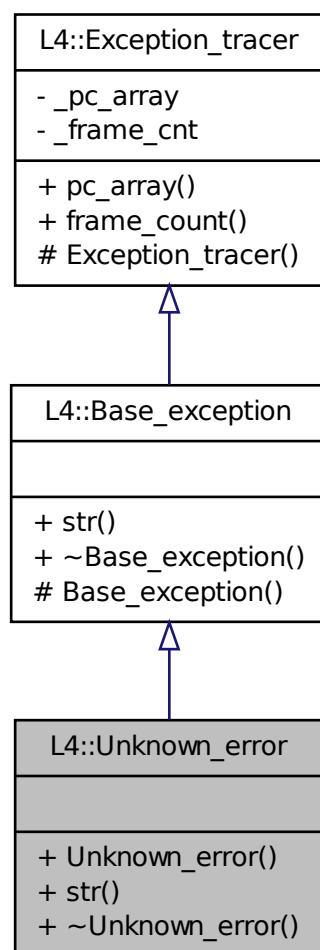
- l4/sys/\_\_typeinfo.h

## 11.189 L4::Unknown\_error Class Reference

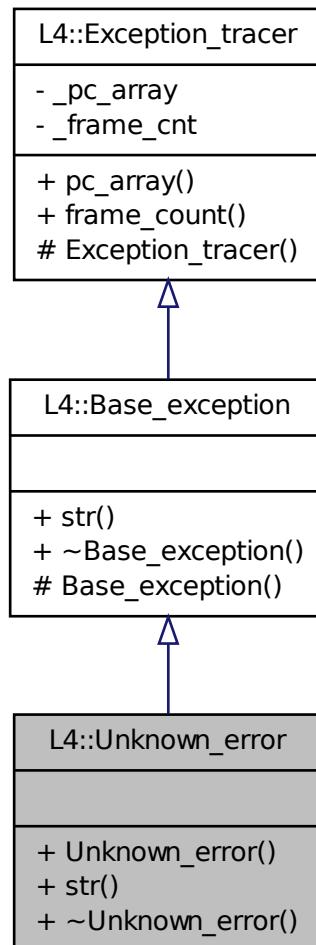
Exception for an unknown condition.

```
#include <l4/cxx/exceptions>
```

Inheritance diagram for L4::Unknown\_error:



Collaboration diagram for L4::Unknown\_error:



## Public Member Functions

- `char const * str () const throw ()`  
*Should return a human readable string for the exception.*

### 11.189.1 Detailed Description

Exception for an unknown condition. This error is usually used when a server returns an unknown return state to the client, this may indicate incompatible messages used by the client and the server.

Definition at line 193 of file [exceptions](#).

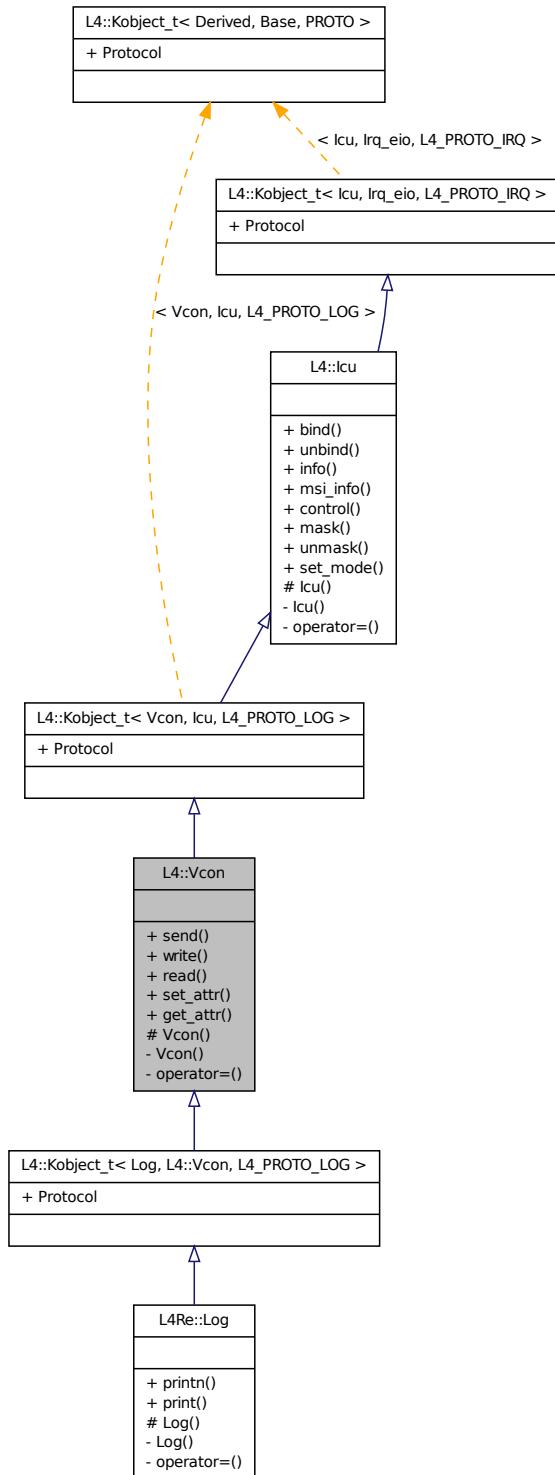
The documentation for this class was generated from the following file:

- 14/cxx/exceptions

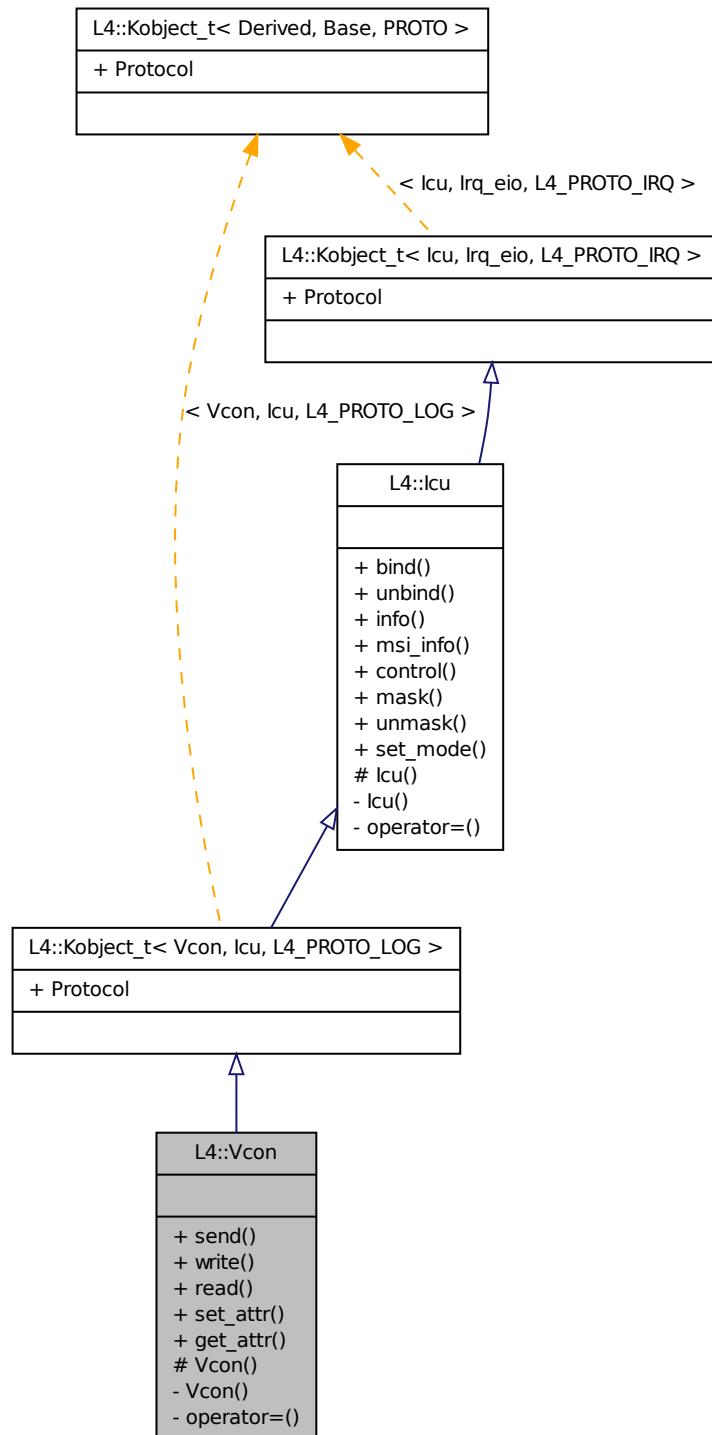
## 11.190 L4::Vcon Class Reference

C++ [L4 Vcon](#).

Inheritance diagram for L4::Vcon:



Collaboration diagram for L4::Vcon:



## Public Member Functions

- `l4_mshtag_t send (char const *buf, int size, l4_utcb_t *utcb=l4_utcb()) const throw ()`
- `long write (char const *buf, int size, l4_utcb_t *utcb=l4_utcb()) const throw ()`
- `int read (char *buf, int size, l4_utcb_t *utcb=l4_utcb()) const throw ()`
- `l4_mshtag_t set_attr (l4_vcon_attr_t const *attr, l4_utcb_t *utcb=l4_utcb()) const throw ()`
- `l4_mshtag_t get_attr (l4_vcon_attr_t *attr, l4_utcb_t *utcb=l4_utcb()) const throw ()`

### 11.190.1 Detailed Description

C++ [L4 Vcon](#). #include <l4/sys/vcon>

#### See also

[Virtual Console](#) for an overview and C bindings.

Definition at line 41 of file [vcon](#).

### 11.190.2 Member Function Documentation

#### 11.190.2.1 `l4_mshtag_t L4::Vcon::send ( char const * buf, int size, l4_utcb_t * utcb = l4_utcb() ) const throw () [inline]`

Send data to virtual console.

##### Parameters

`vcon` Vcon object.

`buf` Pointer to data buffer.

`size` Size of buffer in bytes.

##### Returns

Syscall return tag

##### Note

Size must not exceed L4\_VCON\_WRITE\_SIZE.

##### Note

`vcon` is the implicit *this* pointer.

Definition at line 52 of file [vcon](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



**11.190.2.2 long L4::Vcon::write ( char const \* buf, int size, l4\_utcb\_t \* utcb = `L4_utcb()` ) const throw () [inline]**

Write data to virtual console.

#### Parameters

*vcon* Vcon object.  
*buf* Pointer to data buffer.  
*size* Size of buffer in bytes.

#### Returns

Number of bytes written to the virtual console.

#### Note

*vcon* is the implicit *this* pointer.

Definition at line 60 of file [vcon](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



**11.190.2.3 int L4::Vcon::read ( char \* buf, int size, l4\_utcb\_t \* utcb = `L4_utcb()` ) const throw () [inline]**

Read data from virtual console.

**Parameters**

*vcon* Vcon object.  
*buf* Pointer to data buffer.  
*size* Size of buffer in bytes.

**Returns**

Negative error code on error, > size if more to read, size bytes are in the buffer, <= size bytes read

**Note**

*vcon* is the implicit *this* pointer.

Definition at line 68 of file [vcon](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



#### 11.190.2.4 l4\_mshtag\_t L4::Vcon::set\_attr ( l4\_vcon\_attr\_t const \* attr, l4\_utcb\_t \* utcb = l4\_utcb() ) const throw () [inline]

Set attributes of a Vcon.

**Parameters**

*vcon* Vcon object.  
*attr* Attribute structure.

**Returns**

Syscall return tag

**Note**

*vcon* is the implicit *this* pointer.

Definition at line 76 of file [vcon](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



### 11.190.2.5 l4\_mshtag\_t L4::Vcon::get\_attr ( l4\_vcon\_attr\_t \* attr, l4\_utcb\_t \* utcb = l4\_utcb() ) const throw () [inline]

Get attributes of a Vcon.

#### Parameters

*vcon* Vcon object.

#### Return values

*attr* Attribute structure.

#### Returns

Syscall return tag

#### Note

*vcon* is the implicit *this* pointer.

Definition at line 84 of file [vcon](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



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

- [l4/sys/vcon](#)

## 11.191 L4Re::Util::Vcon\_svr< SVR > Class Template Reference

Console server template class.

### Public Member Functions

- int `dispatch (l4_umword_t obj, L4::Ipc::Iostream &ios)`

*Server dispatch function.*

#### 11.191.1 Detailed Description

`template<typename SVR> class L4Re::Util::Vcon_svr< SVR >`

Console server template class. This template uses vcon\_write() and vcon\_read() to get and deliver data from the implementor.

vcon\_read() needs to return a value bigger than the given size to indicate that there's more data to read for the other end.

vcon\_write() gets the live data from the UTCB. Make sure to copy out the data before using the UTCB again.

The size parameter of both function is given in bytes.

Definition at line 43 of file [vcon\\_svr](#).

#### 11.191.2 Member Function Documentation

##### 11.191.2.1 `template<typename SVR > int L4Re::Util::Vcon_svr< SVR >::dispatch ( l4_umword_t obj, L4::Ipc::Iostream & ios )`

Server dispatch function.

#### Parameters

*obj* Server object ID to work on.

*ios* Input/Output stream.

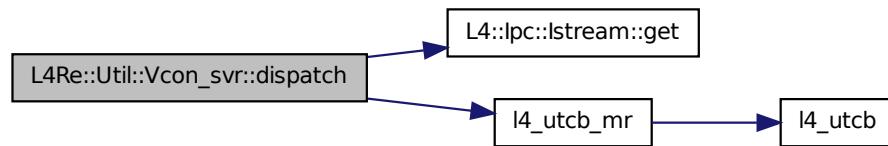
#### Returns

error code.

Definition at line 74 of file [vcon\\_svr](#).

References [L4::Ipc::Istream::get\(\)](#), [L4\\_EBADPROTO](#), [L4\\_EOK](#), [L4\\_UTCB\\_GENERIC\\_DATA\\_SIZE](#), [l4\\_utcb\\_mr\(\)](#), [L4\\_VCON\\_GET\\_ATTR\\_OP](#), [L4\\_VCON\\_SET\\_ATTR\\_OP](#), [L4\\_VCON\\_WRITE\\_OP](#), and [l4\\_msg\\_regs\\_t::mr](#).

Here is the call graph for this function:



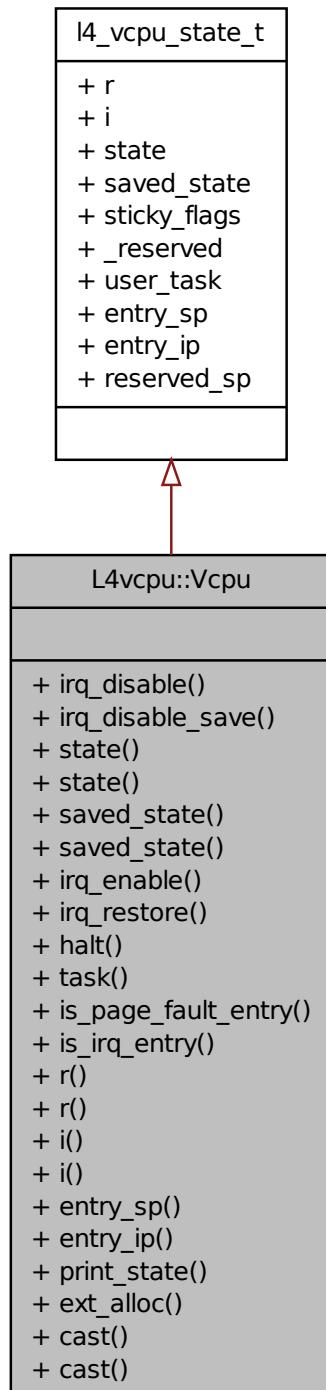
The documentation for this class was generated from the following file:

- [l4/re/util/vcon\\_svr](#)

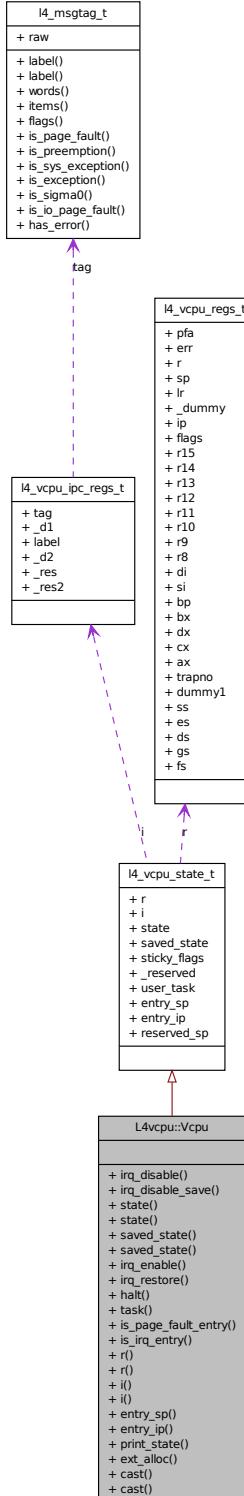
## 11.192 L4vcpu::Vcpu Class Reference

C++ implementation of the vCPU save state area.

Inheritance diagram for L4vcpu::Vcpu:



Collaboration diagram for L4vcpu::Vcpu:



## Public Types

- **typedef l4vcpu\_irq\_state\_t** **Irq\_state**  
*IRQ status type.*

## Public Member Functions

- **void irq\_disable () throw ()**  
*Disable the vCPU for event delivery.*
- **Irq\_state irq\_disable\_save () throw ()**  
*Disable the vCPU for event delivery and return previous state.*
- **State \* state () throw ()**  
*Get state word.*
- **State state () const throw ()**  
*Get state word.*
- **State \* saved\_state () throw ()**  
*Get saved\_state word.*
- **State saved\_state () const throw ()**  
*Get saved\_state word.*
- **void irq\_enable (l4\_utcb\_t \*utcb, l4vcpu\_event\_hdl\_t do\_event\_work\_cb, l4vcpu\_setup\_ipc\_t setup\_ipc) throw ()**  
*Enable the vCPU for event delivery.*
- **void irq\_restore (Irq\_state s, l4\_utcb\_t \*utcb, l4vcpu\_event\_hdl\_t do\_event\_work\_cb, l4vcpu\_setup\_ipc\_t setup\_ipc) throw ()**  
*Restore a previously saved IRQ/event state.*
- **void halt (l4\_utcb\_t \*utcb, l4vcpu\_event\_hdl\_t do\_event\_work\_cb, l4vcpu\_setup\_ipc\_t setup\_ipc) throw ()**  
*Halt/block the vCPU.*
- **void task (L4::Cap< L4::Task > const task=L4::Cap< L4::Task >::Invalid) throw ()**  
*Set the task of the vCPU.*
- **int is\_page\_fault\_entry ()**  
*Return whether the entry reason was a page fault.*
- **int is\_irq\_entry ()**  
*Return whether the entry reason was an IRQ/IPC message.*
- **l4\_vcpu\_regs\_t \* r () throw ()**  
*Return pointer to register state.*

- `l4_vcpu_regs_t const * r () const throw ()`  
*Return pointer to register state.*
- `l4_vcpu_ipc_REGS_t * i () throw ()`  
*Return pointer to IPC state.*
- `l4_vcpu_ipc_REGS_t const * i () const throw ()`  
*Return pointer to IPC state.*
- `void entry_sp (l4_umword_t sp)`  
*Set vCPU entry stack pointer.*
- `void entry_ip (l4_umword_t ip)`  
*Set vCPU entry instruction pointer.*
- `void print_state (const char *prefix="") throw ()`  
*Print the state of the vCPU.*

## Static Public Member Functions

- static int `ext_alloc (Vcpu **vcpu, l4_addr_t *ext_state, L4::Cap< L4::Task > task=L4Re::Env::env()->task(), L4::Cap< L4Re::Rm > rm=L4Re::Env::env()->rm()) throw ()`  
*Allocate state area for an extented vCPU.*
- static `Vcpu * cast (void *x) throw ()`  
*Cast a void ponter to a class pointer.*
- static `Vcpu * cast (l4_addr_t x) throw ()`  
*Cast an address to a class pointer.*

### 11.192.1 Detailed Description

C++ implementation of the vCPU save state area.

Definition at line 71 of file `vcpu`.

### 11.192.2 Member Function Documentation

#### 11.192.2.1 `Irq_state L4vcpu::Vcpu::irq_disable_save ( ) throw () [inline]`

Disable the vCPU for event delivery and return previous state.

##### Returns

IRQ state before disabling IRQs.

Definition at line 89 of file `vcpu`.

**11.192.2.2 State\* L4vcpu::Vcpu::state ( ) throw () [inline]**

Get state word.

**Returns**

Pointer to state word in the vCPU

Definition at line 96 of file [vcpu](#).

**11.192.2.3 State L4vcpu::Vcpu::state ( ) const throw () [inline]**

Get state word.

**Returns**

Pointer to state word in the vCPU

Definition at line 103 of file [vcpu](#).

**11.192.2.4 State\* L4vcpu::Vcpu::saved\_state ( ) throw () [inline]**

Get saved\_state word.

**Returns**

Pointer to saved\_state word in the vCPU

Definition at line 110 of file [vcpu](#).

**11.192.2.5 State L4vcpu::Vcpu::saved\_state ( ) const throw () [inline]**

Get saved\_state word.

**Returns**

Pointer to saved\_state word in the vCPU

Definition at line 117 of file [vcpu](#).

**11.192.2.6 void L4vcpu::Vcpu::irq\_enable ( l4\_utcb\_t \* utcb, l4vcpu\_event\_hdl\_t do\_event\_work\_cb, l4vcpu\_setup\_ipc\_t setup\_ipc ) throw () [inline]**

Enable the vCPU for event delivery.

**Parameters**

*utcb* The UTCB to use.

*do\_event\_work\_cb* Call-back function that is called in case an event (such as an interrupt) is pending.

*setup\_ipc* Call-back function that is called before an IPC operation is called.

Definition at line 129 of file [vcpu](#).

---

**11.192.2.7 void L4vcpu::Vcpu::irq\_restore ( *Irq\_state s*, *l4\_utcb\_t \* utcb*, *l4vcpu\_event\_hdl\_t do\_event\_work\_cb*, *l4vcpu\_setup\_ipc\_t setup\_ipc* ) throw () [inline]**

Restore a previously saved IRQ/event state.

#### Parameters

*s* IRQ state to be restored.

*utcb* The UTCB to use.

*do\_event\_work\_cb* Call-back function that is called in case an event (such as an interrupt) is pending.

*setup\_ipc* Call-back function that is called before an IPC operation is called.

Definition at line 143 of file [vcpu](#).

**11.192.2.8 void L4vcpu::Vcpu::halt ( *l4\_utcb\_t \* utcb*, *l4vcpu\_event\_hdl\_t do\_event\_work\_cb*, *l4vcpu\_setup\_ipc\_t setup\_ipc* ) throw () [inline]**

Halt/block the vCPU.

#### Parameters

*utcb* The UTCB to use.

*do\_event\_work\_cb* Call-back function that is called in case an event (such as an interrupt) is pending.

*setup\_ipc* Call-back function that is called before an IPC operation is called.

Definition at line 157 of file [vcpu](#).

**11.192.2.9 void L4vcpu::Vcpu::task ( *L4::Cap< L4::Task > const task = L4::Cap<L4::Task>::Invalid* ) throw () [inline]**

Set the task of the vCPU.

#### Parameters

*task* Task to set, defaults to invalid task.

Definition at line 165 of file [vcpu](#).

**11.192.2.10 int L4vcpu::Vcpu::is\_page\_fault\_entry ( ) [inline]**

Return whether the entry reason was a page fault.

return 0 if not, !=0 otherwise.

Definition at line 172 of file [vcpu](#).

**11.192.2.11 int L4vcpu::Vcpu::is\_irq\_entry ( ) [inline]**

Return whether the entry reason was an IRQ/IPC message.

return 0 if not, !=0 otherwise.

Definition at line 179 of file [vcpu](#).

**11.192.2.12 l4\_vcpu\_regs\_t\* L4vcpu::Vcpu::r( ) throw() [inline]**

Return pointer to register state.

**Returns**

Pointer to register state.

Definition at line 186 of file [vcpu](#).

**11.192.2.13 l4\_vcpu\_regs\_t const\* L4vcpu::Vcpu::r( ) const throw() [inline]**

Return pointer to register state.

**Returns**

Pointer to register state.

Definition at line 193 of file [vcpu](#).

**11.192.2.14 l4\_vcpu\_ipc\_regs\_t\* L4vcpu::Vcpu::i( ) throw() [inline]**

Return pointer to IPC state.

**Returns**

Pointer to IPC state.

Definition at line 200 of file [vcpu](#).

**11.192.2.15 l4\_vcpu\_ipc\_regs\_t const\* L4vcpu::Vcpu::i( ) const throw() [inline]**

Return pointer to IPC state.

**Returns**

Pointer to IPC state.

Definition at line 207 of file [vcpu](#).

**11.192.2.16 void L4vcpu::Vcpu::entry\_sp( l4\_umword\_t sp ) [inline]**

Set vCPU entry stack pointer.

**Parameters**

*sp* Stack pointer address to set.

**Note**

The value is only used when entering from a user-task.

Definition at line 216 of file [vcpu](#).

**11.192.2.17 void L4vcpu::Vcpu::entry\_ip( l4\_umword\_t ip ) [inline]**

Set vCPU entry instruction pointer.

**Parameters**

*ip* Instruction pointer address to set.

Definition at line 223 of file [vcpu](#).

**11.192.2.18 static int L4vcpu::Vcpu::ext\_alloc( Vcpu \*\* vcpu, l4\_addr\_t \* ext\_state, L4::Cap< L4::Task > task = L4Re::Env::env() ->task(), L4::Cap< L4Re::Rm > rm = L4Re::Env::env() ->rm() ) throw() [static]**

Allocate state area for an extented vCPU.

**Return values**

*vcpu* Allocated vcpu-state area.

*ext\_state* Allocated extended vcpu-state area.

**Parameters**

*task* Task to use for allocation, defaults to own task.

*rm* Region manager to use for allocation defaults to standard region manager.

**Returns**

0 for success, error code otherwise

**11.192.2.19 static Vcpu\* L4vcpu::Vcpu::cast( void \* x ) throw() [inline, static]**

Cast a void ponter to a class pointer.

**Parameters**

*x* Pointer.

**Returns**

Pointer to [Vcpu](#) class.

Definition at line 249 of file [vcpu](#).

**11.192.2.20 static Vcpu\* L4vcpu::Vcpu::cast( l4\_addr\_t x ) throw() [inline, static]**

Cast an address to a class pointer.

**Parameters**

*x* Pointer.

**Returns**

Pointer to [Vcpu](#) class.

Definition at line [259](#) of file [vcpu](#).

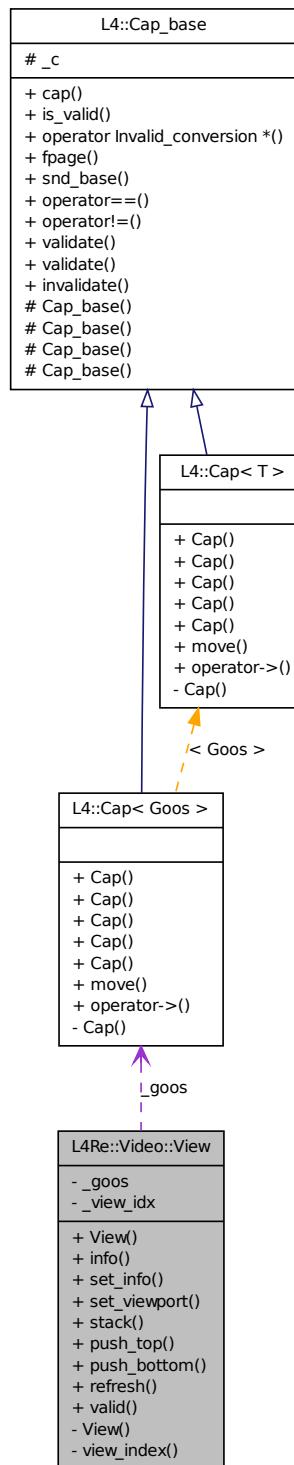
The documentation for this class was generated from the following file:

- l4/vcpu/vcpu

**11.193 L4Re::Video::View Class Reference**

[View](#).

Collaboration diagram for L4Re::Video::View:



## Data Structures

- struct [Info](#)

*Information structure of a view.*

## Public Types

- enum [Flags](#) {  
    [F\\_none](#) = 0x00, [F\\_set\\_buffer](#) = 0x01, [F\\_set\\_buffer\\_offset](#) = 0x02, [F\\_set\\_bytes\\_per\\_line](#) = 0x04,  
    [F\\_set\\_pixel](#) = 0x08, [F\\_set\\_position](#) = 0x10, [F\\_dyn\\_allocated](#) = 0x20, [F\\_set\\_background](#) = 0x40,  
    [F\\_set\\_flags](#) = 0x80, [F\\_fully\\_dynamic](#) }  
*Flags on a view.*
- enum [V\\_flags](#) { [F\\_above](#) = 0x1000, [F\\_flags\\_mask](#) = 0xff000 }  
*Property flags of a view.*

## Public Member Functions

- int [info](#) ([Info](#) \*info) const throw ()  
*Return the view information of the view.*
- int [set\\_info](#) ([Info](#) const &info) const throw ()  
*Set the information structure for this view.*
- int [set\\_viewport](#) (int scr\_x, int scr\_y, int w, int h, unsigned long buf\_offset) const throw ()  
*Set the position of the view in the goos.*
- int [stack](#) ([View](#) const &pivot, bool behind=true) const throw ()  
*Move this view in the view stack.*
- int [push\\_top](#) () const throw ()  
*Make this view the top-most view.*
- int [push\\_bottom](#) () const throw ()  
*Push this view the back.*
- int [refresh](#) (int x, int y, int w, int h) const throw ()  
*Refresh/Redraw the view.*
- bool [valid](#) () const  
*Return whether this view is valid.*

## Friends

- class [Goos](#)  
*ID for goos objects.*

### 11.193.1 Detailed Description

[View.](#)

Definition at line 34 of file [view](#).

### 11.193.2 Member Enumeration Documentation

#### 11.193.2.1 enum L4Re::Video::View::Flags

Flags on a view.

**Enumerator:**

*F\_none* everything for this view is static (the VESA-FB case)

*F\_set\_buffer* buffer object for this view can be changed

*F\_set\_buffer\_offset* buffer offset can be set

*F\_set\_bytes\_per\_line* bytes per line can be set

*F\_set\_pixel* pixel type can be set

*F\_set\_position* position on screen can be set

*F\_dyn\_allocated* [View](#) is dynamically allocated.

*F\_set\_background* Set view as background for session.

*F\_set\_flags* Set view flags (.

**See also**

[V\\_flags](#))

*F\_fully\_dynamic* Flags for a fully dynamic view.

Definition at line 53 of file [view](#).

#### 11.193.2.2 enum L4Re::Video::View::V\_flags

Property flags of a view.

Such flags can be set or deleted with the [F\\_set\\_flags](#) operation using the [set\\_info\(\)](#) method.

**Enumerator:**

*F\_above* Flag the view as stay on top.

*F\_flags\_mask* Mask containing all possible property flags.

Definition at line 76 of file [view](#).

### 11.193.3 Member Function Documentation

#### 11.193.3.1 int L4Re::Video::View::info ( Info \* *info* ) const throw ()

Return the view information of the view.

**Return values**

*info* Information structure pointer.

**Returns**

0 on success, error otherwise

**11.193.3.2 int L4Re::Video::View::set\_info ( Info const & *info* ) const throw ()**

Set the information structure for this view.

**Parameters**

*info* Information structure.

**Returns**

0 on success, error otherwise

The function will also set the view port according to the values given in the information structure.

**11.193.3.3 int L4Re::Video::View::set\_viewport ( int *scr\_x*, int *scr\_y*, int *w*, int *h*, unsigned long *buf\_offset* ) const throw ()**

Set the position of the view in the goos.

**Parameters**

*scr\_x* X position

*scr\_y* Y position

*w* Width

*h* Height

*buf\_offset* Offset in the buffer in bytes

**Returns**

0 on success, error otherwise

**11.193.3.4 int L4Re::Video::View::stack ( View const & *pivot*, bool *behind = true* ) const throw ()**

Move this view in the view stack.

**Parameters**

*pivot* [View](#) to move relative to

*behind* When true move the view behind the pivot view, if false move the view before the pivot view.

**Returns**

0 on success, error otherwise

### 11.193.3.5 int L4Re::Video::View::refresh ( int *x*, int *y*, int *w*, int *h* ) const throw ()

Refresh/Redraw the view.

#### Parameters

*x* X position.

*y* Y position.

*w* Width.

*h* Height.

#### Returns

0 on success, error otherwise

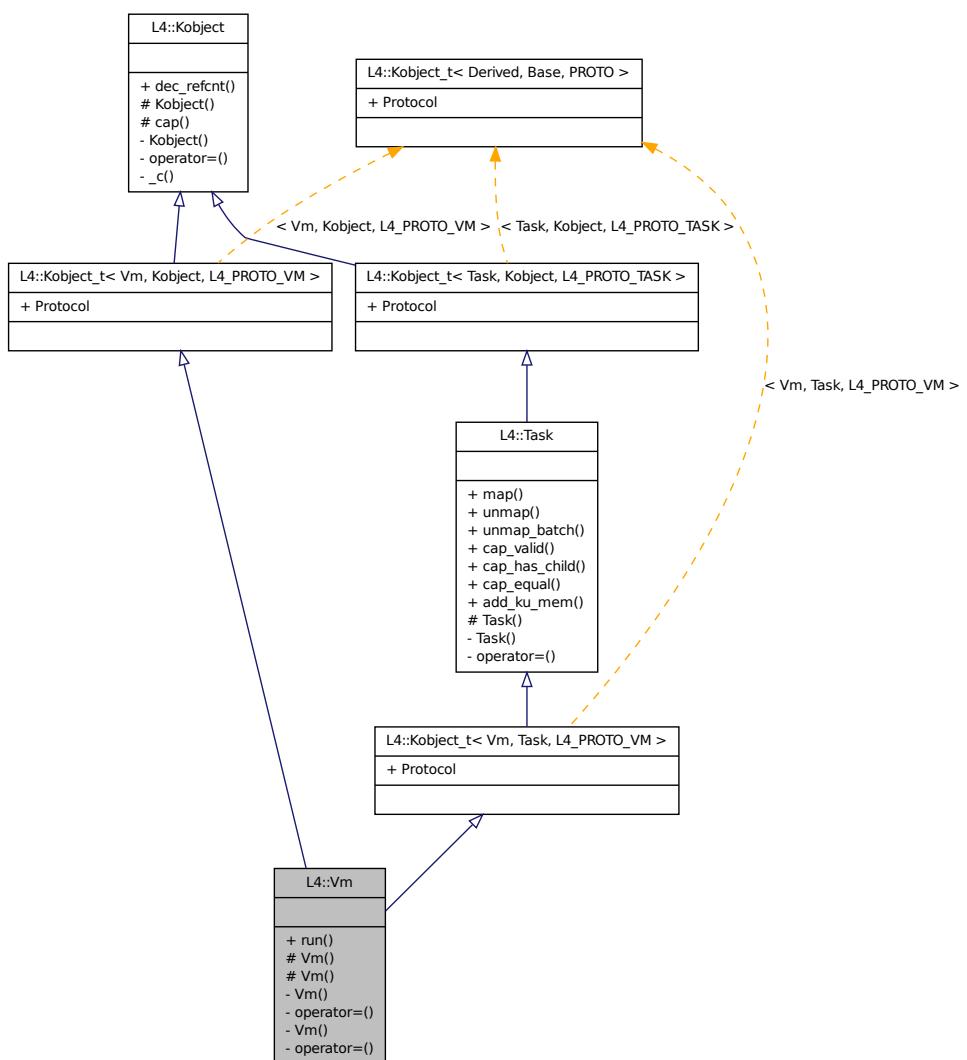
The documentation for this class was generated from the following file:

- l4/re/video/view

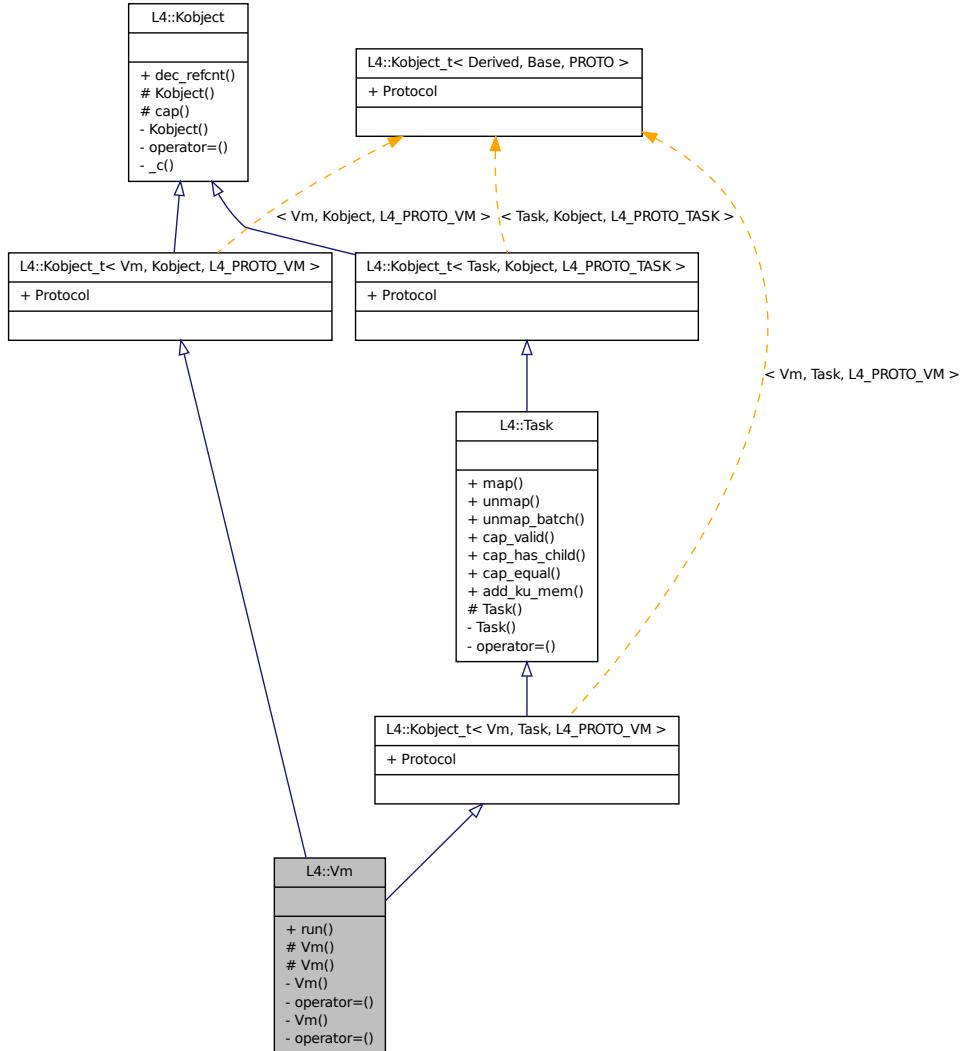
## 11.194 L4::Vm Class Reference

Virtual machine.

Inheritance diagram for L4::Vm:



Collaboration diagram for L4::Vm:



## Public Member Functions

- `l4_mshtag_t run (l4_fpage_t const &fpage, l4_umword_t *label, l4_utcb_t *utcb=l4_utcb()) throw ()`

### 11.194.1 Detailed Description

Virtual machine. TZ Virtual machine.

Definition at line 36 of file [\\_\\_vm](#).

## 11.194.2 Member Function Documentation

**11.194.2.1 l4\_mshtag\_t L4::Vm::run ( l4\_fpage\_t const & fpage, l4\_umword\_t \* label, l4\_utcb\_t \* utcb = *l4\_utcb()* ) throw () [inline]**

Run a VM.

### Parameters

*vm* Capability selector for VM

### Note

*dst\_task* is the implicit *this* pointer.

Definition at line 51 of file [vm](#).

References [L4::Kobject::cap\(\)](#).

Here is the call graph for this function:



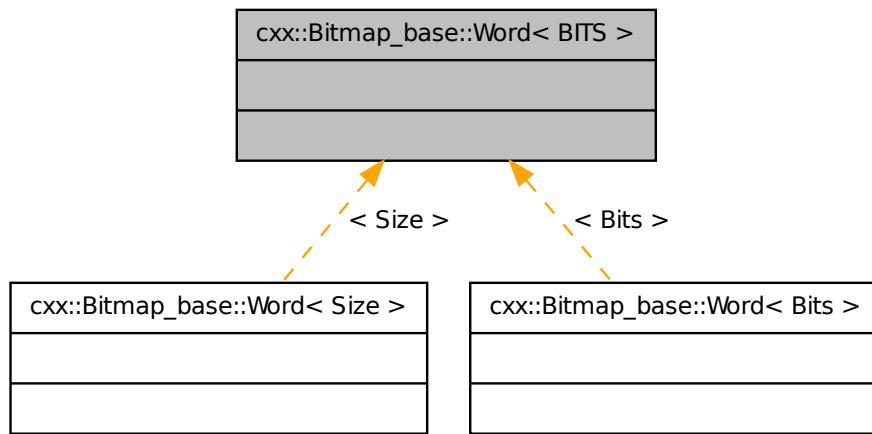
The documentation for this class was generated from the following files:

- [l4/sys/\\_\\_vm](#)
- [arm/l4/sys/vm](#)

## 11.195 cxx::Bitmap\_base::Word< BITS > Class Template Reference

Helper abstraction for a word contained in the bitmap.

Inheritance diagram for cxx::Bitmap\_base::Word< BITS >:



### 11.195.1 Detailed Description

**template<long BITS> class cxx::Bitmap\_base::Word< BITS >**

Helper abstraction for a word contained in the bitmap.

Definition at line 50 of file [bitmap](#).

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

- [l4/cxx\(bitmap\)](#)

# Chapter 12

## Example Documentation

### 12.1 examples/clntsrv/client.cc

Client/Server example using C++ infrastructure -- Client implementation.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *                  Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universitt Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/err.h>
#include <l4/sys/types.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>
#include <l4/cxx/ ipc_stream>

#include <stdio.h>

#include "shared.h"

int
func_neg_call(L4::Cap<void> const &server, l4_uint32_t *result,
              l4_uint32_t val)
{
    L4::Ipc::Iostream s(l4_utcb());
    s << l4_umword_t(Opcode::func_neg) << val;
    l4_mshtag_t res = s.call(server.cap());
    if (l4_ipc_error(res, l4_utcb()))
        return 1; // failure
    s >> *result;
    return 0; // ok
}

int
func_sub_call(L4::Cap<void> const &server, l4_uint32_t *result,
              l4_uint32_t val1, l4_uint32_t val2)
{
    L4::Ipc::Iostream s(l4_utcb());
    s << l4_umword_t(Opcode::func_sub) << val1 << val2;
    l4_mshtag_t res = s.call(server.cap(), Protocol::Calc);
    if (l4_ipc_error(res, l4_utcb()))
        return 1; // failure
}
```

```

    s >> *result;
    return 0; // ok
}

int
main()
{
    L4::Cap<void> server = L4Re::Env::env()->get_cap<void>("calc_server");
    if (!server.is_valid())
    {
        printf("Could not get server capability!\n");
        return 1;
    }

    l4_uint32_t val1 = 8;
    l4_uint32_t val2 = 5;

    printf("Asking for %d - %d\n", val1, val2);

    if (func_sub_call(server, &val1, val1, val2))
    {
        printf("Error talking to server\n");
        return 1;
    }
    printf("Result of subtract call: %d\n", val1);
    printf("Asking for -%d\n", val1);
    if (func_neg_call(server, &val1, val1))
    {
        printf("Error talking to server\n");
        return 1;
    }
    printf("Result of negate call: %d\n", val1);

    return 0;
}

```

## 12.2 examples/clntsrv/clntsrv.cfg

Sample configuration file for the client/server example.

```

-- vim:set ft=lua:

-- Include L4 functionality
require("L4");

-- Some shortcut for less typing
local ld = L4.default_loader;

-- Channel for the two programs to talk to each other.
local calc_server = ld:new_channel();

-- The server program, getting the channel in server mode.
ld:start({ caps = { calc_server = calc_server:svr() },
           log = { "server", "blue" } },
           "rom/ex_clntsrv-server");

-- The client program, getting the 'calc_server' channel to be able to talk
-- to the server. The client will be started with a green log output.
ld:start({ caps = { calc_server = calc_server },
           log = { "client", "green" } },
           "rom/ex_clntsrv-client");

```

## 12.3 examples/clntsrv/server.cc

Client/Server example using C++ infrastructure -- Server implementation.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universitt Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <stdio.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/object_registry>
#include <l4/cxx/ipc_server>

#include "shared.h"

static L4Re::Util::Registry_server<> server;

class Calculation_server : public L4::Server_object
{
public:
    int dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios);
};

int
Calculation_server::dispatch(l4_umword_t, L4::Ipc::Iostream &ios)
{
    l4_mshtag_t t;
    ios >> t;

    // We're only talking the calculation protocol
    if (t.label() != Protocol::Calc)
        return -L4_EBADPROTO;

    L4::Opcode opcode;
    ios >> opcode;

    switch (opcode)
    {
        case Opcode::func_neg:
            l4_uint32_t val;
            ios >> val;
            val = -val;
            ios << val;
            return L4_EOK;
        case Opcode::func_sub:
            l4_uint32_t val1, val2;
            ios >> val1 >> val2;
            val1 -= val2;
            ios << val1;
            return L4_EOK;
        default:
            return -L4_ENOSYS;
    }
}

int
main()
{
    static Calculation_server calc;
```

```

// Register calculation server
if (!server.registry()->register_obj(&calc, "calc_server").is_valid())
{
    printf("Could not register my service, readonly namespace?\n");
    return 1;
}

printf("Welcome to the calculation server!\n"
       "I can do substractions and negations.\n");

// Wait for client requests
server.loop();

return 0;
}

```

## 12.4 examples/libs/l4re/c++/mem\_alloc/ma+rm.cc

Coarse grained memory allocation, in C++.

```

/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *      economic rights: Technische Universitt Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

#include <l4/re/mem_alloc>
#include <l4/re/rm>
#include <l4/re/env>
#include <l4/re/dataspace>
#include <l4/re/util/cap_alloc>
#include <sys/err.h>
#include <cstdio>
#include <cstring>

static int allocate_mem(unsigned long size_in_bytes, unsigned long flags,
                      void **virt_addr)
{
    int r;
    L4::Cap<L4Re::Dataspace> d;

    /* Allocate a free capability index for our data space */
    d = L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>();
    if (!d.is_valid())
        return -L4_ENOMEM;

    size_in_bytes = l4_trunc_page(size_in_bytes);

    /* Allocate memory via a dataspace */
    if ((r = L4Re::Env::env()->mem_alloc()->alloc(size_in_bytes, d, flags)))
        return r;

    /* Make the dataspace visible in our address space */
    *virt_addr = 0;
    if ((r = L4Re::Env::env()->rm()->attach(virt_addr, size_in_bytes,
                                                L4Re::Rm::Search_addr, d, 0,
                                                flags & L4Re::Mem_alloc::Super_pages
                                                ? L4_SUPERPAGESHIFT : L4_PAGESHIFT)))
        return r;
}

```

```

/* Done, virtual address is in virt_addr */
return 0;
}

static int free_mem(void *virt_addr)
{
    int r;
    L4::Cap<L4Re::Dataspace> ds;

    /* Detach memory from our address space */
    if ((r = L4Re::Env::env()>rm()>detach(virt_addr, &ds)))
        return r;

    /* Free memory at our memory allocator, this is optional */
    if ((r = L4Re::Env::env()>mem_alloc()>free(ds)))
        return r;

    /* Release and return capability slot to allocator */
    L4Re::Util::cap_alloc.free(ds, L4Re::Env::env()>task().cap());

    /* All went ok */
    return 0;
}

int main(void)
{
    void *virt;

    /* Allocate memory: 16k Bytes (usually) */
    if (allocate_mem(4 * L4_PAGESIZE, 0, &virt))
        return 1;

    printf("Allocated memory.\n");

    /* Do something with the memory */
    memset(virt, 0x12, 4 * L4_PAGESIZE);

    printf("Touched memory.\n");

    /* Free memory */
    if (free_mem(virt))
        return 2;

    printf("Freed and done. Bye.\n");

    return 0;
}

```

## 12.5 examples/libs/l4re/c++/shared\_ds/ds\_clnt.cc

Sharing memory between applications, client side.

```

/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universitt Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

#include <l4/re/util/cap_alloc> // L4::Cap

```

```

#include <l4/re/dataspace>           // L4Re::Dataspace
#include <l4/re/rm>                 // L4::Rm
#include <l4/re/env>                // L4::Env
#include <l4/sys/cache.h>

#include <cstring>
#include <cstdio>
#include <unistd.h>

#include "interface.h"

int main()
{
    /*
     * Try to get server interface cap.
     */

    L4::Cap<My_interface> svr = L4Re::Env::env()->get_cap<My_interface>("shm");
    if (!svr.is_valid())
    {
        printf("Could not get the server capability\n");
        return 1;
    }

    /*
     * Alloc data space cap slot
     */
    L4::Cap<L4Re::Dataspace> ds = L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>();
    if (!ds.is_valid())
    {
        printf("Could not get capability slot!\n");
        return 1;
    }

    /*
     * Alloc server notifier IRQ cap slot
     */
    L4::Cap<L4::Irq> irq = L4Re::Util::cap_alloc.alloc<L4::Irq>();
    if (!irq.is_valid())
    {
        printf("Could not get capability slot!\n");
        return 1;
    }

    /*
     * Request shared data-space cap.
     */
    if (svr->get_shared_buffer(ds, irq))
    {
        printf("Could not get shared memory dataspace!\n");
        return 1;
    }

    /*
     * Attach to arbitrary region
     */
    char *addr = 0;
    int err = L4Re::Env::env()->rm()->attach(&addr, ds->size(),
                                                L4Re::Rm::Search_addr, ds);
    if (err < 0)
    {
        printf("Error attaching data space: %s\n", l4sys_errtostr(err));
        return 1;
    }

    printf("Content: %s\n", addr);
}

```

```

// wait a bit for the demo effect
sleep(3);

/*
 * Fill in new stuff
 */
memset(addr, 0, ds->size());
char const * const msg = "Hello from client, too!";
printf("Setting new content in shared memory\n");
snprintf(addr, strlen(msg)+1, msg);
l4_cache_clean_data((unsigned long)addr, (unsigned long)addr + strlen(msg) + 1)
;

// notify the server
irq->trigger();

/*
 * Detach region containing addr, result should be Detached_ds (other results
 * only apply if we split regions etc.).
 */
err = L4Re::Env::env()->rm()->detach(addr, 0);
if (err)
    printf("Failed to detach region\n");

L4Re::Util::cap_alloc.free(ds, L4Re::This_task);

return 0;
}

```

## 12.6 examples/libs/l4re/c++/shared\_ds/ds\_srv.cc

Sharing memory between applications, server/creator side.

```

/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universitt Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

#include <l4/re/env>
#include <l4/re/namespace>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/object_registry>
#include <l4/re/dataspace>
#include <l4/cxx/ipc_server>

#include <l4/sys/typeinfo_svr>

#include <cstring>
#include <cstdio>
#include <unistd.h>

#include "interface.h"

class My_server_obj : public L4::Server_object
{
private:
    L4::Cap<L4Re::Dataspace> _shm;
    L4::Cap<L4::Irq> _irq;

```

```

public:
    explicit My_server_obj(L4::Cap<L4Re::Dataspace> shm, L4::Cap<L4::Irq> irq)
    : _shm(shm), _irq(irq)
    {}

    int dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios);
};

int My_server_obj::dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios)
{
    // we don't care about the original object reference, however
    // we could read out the access rights from the lowest 2 bits
    (void) obj;

    l4_mshtag_t t;
    ios >> t; // extract the tag

    switch (t.label())
    {
        case L4::Meta::Protocol:
            // handle the meta protocol requests, implementing the
            // runtime dynamic type system for L4 objects.
            return L4::Util::handle_meta_request<My_interface>(ios);
        case 0:
            // since we have just one operation we have no opcode dispatch,
            // and just return the data-space and the notifier IRQ capabilities
            ios << _shm << _irq;
            return 0;
        default:
            // every other protocol is not supported.
            return -L4_EBADPROTO;
    }
}

class Shm_observer : public L4::Server_object
{
private:
    char *_shm;

public:
    explicit Shm_observer(char *shm)
    : _shm(shm)
    {}

    int dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios);
};

int Shm_observer::dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios)
{
    // we don't care about the original object reference, however
    // we could read out the access rights from the lowest 2 bits
    (void) obj;

    l4_mshtag_t t;
    ios >> t; // extract the tag

    switch (t.label())
    {
        case L4::Irq::Protocol:
            // Got an IRQ so just print the new contents of the
            // shared memory.
            printf("Content: %s\n", _shm);
            return 0;
        default:
            // every other protocol is not supported.
    }
}

```

```

        return -L4_EBADPROTO;
    }

static L4Re::Util::Registry_server<> server;

enum
{
    DS_SIZE = 4 << 12,
};

static char *get_ds(L4::Cap<L4Re::Dataspace> *_ds)
{
    *_ds = L4Re::Util::cap_alloc.alloc<L4Re::Dataspace>();
    if (!(*_ds).is_valid())
    {
        printf("Dataspace allocation failed.\n");
        return 0;
    }

    int err = L4Re::Env::env()->mem_alloc()->alloc(DS_SIZE, *_ds, 0);
    if (err < 0)
    {
        printf("mem_alloc->alloc() failed.\n");
        L4Re::Util::cap_alloc.free(*_ds);
        return 0;
    }

    /*
     * Attach DS to local address space
     */
    char *_addr = 0;
    err = L4Re::Env::env()->rm()->attach(&_addr, (*_ds)->size(),
                                             L4Re::Rm::Search_addr,
                                             *_ds);
    if (err < 0)
    {
        printf("Error attaching data space: %s\n", l4sys_errtostr(err));
        L4Re::Util::cap_alloc.free(*_ds);
        return 0;
    }

    /*
     * Success! Write something to DS.
     */
    printf("Attached DS\n");
    static char const * const msg = "[DS] Hello from server!";
    snprintf(_addr, strlen(msg) + 1, msg);

    return _addr;
}

int main()
{
    L4::Cap<L4Re::Dataspace> ds;
    char *addr;

    if (!(addr = get_ds(&ds)))
        return 2;

    // first the IRQ handler, because we need it in the My_server_obj object
    Shm_observer observer(addr);

    // Registering the observer as an IRQ handler, this allocates an
    // IRQ object using the factory of our server.
}

```

```

L4::Cap<L4::Irq> irq = server.registry()->register_irq_obj(&observer);

// now the initial server object shared with the client via our parent.
// it provides the data-space and the IRQ capabilities to a client.
My_server_obj server_obj(ds, irq);

// Registering the server object to the capability 'shm' in our the L4Re::Env.
// This capability must be provided by the parent. (see the shared_ds.lua)
server.registry()->register_obj(&server_obj, "shm");

// Run our server loop.
server.loop();
return 0;
}

```

## 12.7 examples/libs/l4re/c++/shared\_ds/shared\_ds.lua

Sharing memory between applications, configuration file.

```

-- Include L4 functionality
require("L4");

-- Create a channel from the client to the server
local channel = L4.default_loader:new_channel();

-- Start the server, giving the channel with full server rights.
-- The server will have a yellow log output.
L4.default_loader:start(
{
  caps = { shm = channel:svr() },
  log  = { "server", "yellow" }
},
"rom/ex_l4re_ds_srv"
);

-- Start the client, giving it the channel with read only rights. The
-- log output will be green.
L4.default_loader:start(
{
  caps = { shm = channel },
  log  = { "client", "green" },
  l4re_dbg = L4.Dbg.Warn
},
"rom/ex_l4re_ds_cint"
);

```

## 12.8 examples/libs/l4re/c/ma+rm.c

Coarse grained memory allocation, in C.

```

/*
 * (c) 2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *      economic rights: Technische Universitt Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the

```

```

 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

#include <l4/re/c/mem_alloc.h>
#include <l4/re/c/rm.h>
#include <l4/re/c/util/cap_alloc.h>
#include <l4/sys/err.h>
#include <stdio.h>
#include <string.h>

static int allocate_mem(unsigned long size_in_bytes, unsigned long flags,
                       void **virt_addr)
{
    int r;
    l4re_ds_t ds;

    /* Allocate a free capability index for our data space */
    ds = l4re_util_cap_alloc();
    if (l4_is_invalid_cap(ds))
        return -L4_ENOMEM;

    size_in_bytes = l4_trunc_page(size_in_bytes);

    /* Allocate memory via a dataspace */
    if ((r = l4re_ma_alloc(size_in_bytes, ds, flags)))
        return r;

    /* Make the dataspace visible in our address space */
    *virt_addr = 0;
    if ((r = l4re_rm_attach(virt_addr, size_in_bytes,
                           L4RE_RM_SEARCH_ADDR, ds, 0,
                           flags & L4RE_MA_SUPER_PAGES
                           ? L4_SUPERPAGESHIFT : L4_PAGESHIFT)))
        return r;

    /* Done, virtual address is in virt_addr */
    return 0;
}

static int free_mem(void *virt_addr)
{
    int r;
    l4re_ds_t ds;

    /* Detach memory from our address space */
    if ((r = l4re_rm_detach_ds(virt_addr, &ds)))
        return r;

    /* Free memory at our memory allocator */
    if ((r = l4re_ma_free(ds)))
        return r;

    l4re_util_cap_free(ds);

    /* All went ok */
    return 0;
}

int main(void)
{
    void *virt;

    /* Allocate memory: 16k Bytes (usually) */
    if (allocate_mem(4 * L4_PAGESIZE, 0, &virt))
        return 1;
}

```

```

printf("Allocated memory.\n");

/* Do something with the memory */
memset(virt, 0x12, 4 * L4_PAGESIZE);

printf("Touched memory.\n");

/* Free memory */
if (free_mem(virt))
    return 2;

printf("Freed and done. Bye.\n");

return 0;
}

```

## 12.9 examples/libs/l4re/streammap/client.cc

Client/Server example showing how to map a page to another task -- Client implementation. Note that there's also a shared memory library that supplies this functionality in more convenient way.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universitt Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/err.h>
#include <l4/sys/types.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>
#include <l4/cxx/ ipc_stream>

#include <stdio.h>

#include "shared.h"

int
func_smap_call(L4::Cap<void> const &server)
{
    L4::Ipc::Iostream s(l4_utcb());
    l4_addr_t addr = 0;
    int err;

    if ((err = L4Re::Env::env() ->rm() ->reserve_area(&addr, L4_PAGESIZE,
                                                       L4Re::Rm::Search_addr)))
    {
        printf("The reservation of one page within our virtual memory failed with %d\n", err);
        return 1;
    }

    s << l4_umword_t(Opcode::Do_map)
      << (l4_addr_t)addr;
    s << L4::Ipc::Rcv_fpage::mem((l4_addr_t)addr, L4_PAGESHIFT, 0);
    l4_mshtag_t res = s.call(server.cap(), Protocol::Map_example);
    if (l4_ipc_error(res, l4_utcb()))
        return 1; // failure

    printf("String sent by server: %s\n", (char *)addr);
}

```

```

    return 0; // ok
}

int
main()
{
    L4::Cap<void> server = L4Re::Env::env()->get_cap<void>("smap");
    if (!server.is_valid())
    {
        printf("Could not get capability slot!\n");
        return 1;
    }

    printf("Asking for page from server\n");

    if (func_smap_call(server))
    {
        printf("Error talking to server\n");
        return 1;
    }
    printf("It worked!\n");

    L4Re::Util::cap_alloc.free(server, L4Re::This_task);

    return 0;
}

```

## 12.10 examples/libs/l4re/streammap/server.cc

Client/Server example showing how to map a page to another task -- Server implementation. Note that there's also a shared memory library that supplies this functionality in more convenient way.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *                  Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universitt Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <stdio.h>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>
#include <l4/re/util/object_registry>
#include <l4/cxx/ipc_server>

#include "shared.h"

static char page_to_map[L4_PAGESIZE] __attribute__((aligned(L4_PAGESIZE)));

static L4Re::Util::Registry_server<> server;

class Smap_server : public L4::Server_object
{
public:
    int dispatch(l4_umword_t obj, L4::Ipc::Iostream &ios);
};

int
Smap_server::dispatch(l4_umword_t, L4::Ipc::Iostream &ios)

```

```

{
    l4_mshtag_t t;
    ios >> t;

    // We're only talking the Map_example protocol
    if (t.label() != Protocol::Map_example)
        return -L4_EBADPROTO;

    L4::Opcode opcode;
    ios >> opcode;

    switch (opcode)
    {
        case Opcode::Do_map:
            l4_addr_t snd_base;
            ios >> snd_base;
            // put something into the page to read it out at the other side
            sprintf(page_to_map, sizeof(page_to_map), "Hello from the server!");
            printf("Sending to client\n");
            // send page
            ios << L4::Ipc::Snd_fpage::mem((l4_addr_t)page_to_map, L4_PAGESHIFT,
                                            L4_FPAGE_RO, snd_base);
            return L4_EOK;
        default:
            return -L4_ENOSYS;
    }
}

int
main()
{
    static Smap_server smap;

    // Register server
    if (!server.registry()->register_obj(&smap, "smap").is_valid())
    {
        printf("Could not register my service, read-only namespace?\n");
        return 1;
    }

    printf("Welcome to the memory map example server!\n");

    // Wait for client requests
    server.loop();

    return 0;
}

```

## 12.11 examples/libs/l4re/streammap/streammap.cfg

Sample configuration file for the client/server map example.

```

-- vim:set ft=lua:

-- Include L4 functionality
require("L4");

-- Channel for the communication between the server and the client.
local smap_channel = L4.default_loader:new_channel();

-- The server program, using the 'smap' channel in server
-- mode. The log prefix will be 'server', colored yellow.
L4.default_loader:start({ caps = { smap = smap_channel:svr() },
                         log = { "server", "yellow" } },

```

```

        "rom/ex_smap-server");

-- The client program.
-- It is given the 'smap' channel to be able to talk to the server.
-- The log prefix will be 'client', colored green.
L4.default_loader:start({ caps = { smap = smap_channel },
                           log = { "client", "green" } },
                        "rom/ex_smap-client");

```

## 12.12 examples/libs/libirq/async\_isr.c

libirq usage example using asynchronous ISR handler functionality.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *      economic rights: Technische Universitt Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * This example shall show how to use the libirq.
 */

#include <l4/irq/irq.h>
#include <l4/util/util.h>

#include <stdio.h>

enum { IRQ_NO = 17 };

static void isr_handler(void *data)
{
    (void)data;
    printf("Got IRQ %d\n", IRQ_NO);
}

int main(void)
{
    const int seconds = 5;
    l4irq_t *irqdesc;

    if (!(irqdesc = l4irq_request(IRQ_NO, isr_handler, 0, 0xff, 0)))
    {
        printf("Requesting IRQ %d failed\n", IRQ_NO);
        return 1;
    }

    printf("Attached to key IRQ %d\nPress keys now, will terminate in %d seconds\n"
          , IRQ_NO, seconds);

    l4_sleep(seconds * 1000);

    if (l4irq_release(irqdesc))
    {
        printf("Failed to release IRQ\n");
        return 1;
    }

    printf("Bye\n");
    return 0;
}

```

```
}
```

## 12.13 examples/libs/libirq/loop.c

libirq usage example using a self-created thread.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *      economic rights: Technische Universitt Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

#include <l4/irq/irq.h>
#include <l4/util/util.h>
#include <stdio.h>
#include <pthread.h>

enum { IRQ_NO = 17 };

static void isr_handler(void)
{
    printf("Got IRQ %d\n", IRQ_NO);
}

static void *isr_thread(void *data)
{
    l4irq_t *irq;
    (void)data;

    if (!(irq = l4irq_attach(IRQ_NO)))
        return NULL;

    while (1)
    {
        if (l4irq_wait(irq))
            continue;
        isr_handler();
    }

    return NULL;
}

int main(void)
{
    pthread_t thread;

    if (pthread_create(&thread, NULL, isr_thread, NULL))
        return 1;

    l4_sleep_forever();
    return 0;
}
```

## 12.14 examples/libs/shmc/prodcons.c

Simple shared memory example.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>
 *      economic rights: Technische Universitt Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

/*
 * This example uses shared memory between two threads, one producer, one
 * consumer.
 */

#include <l4/shmc/shmc.h>

#include <l4/util/util.h>

#include <stdio.h>
#include <string.h>
#include <pthread-l4.h>

#include <l4/sys/thread.h>

// a small helper
#define CHK(func) if (func) { printf("failure: %d\n", __LINE__); return (void *)-1; }

static const char some_data[] = "Hi consumer!";

static void *thread_producer(void *d)
{
    (void)d;
    l4shmc_chunk_t p_one;
    l4shmc_signal_t s_one, s_done;
    l4shmc_area_t shmarea;

    // attach this thread to the shm object
    CHK(l4shmc_attach("testshm", &shmarea));

    // add a chunk
    CHK(l4shmc_add_chunk(&shmarea, "one", 1024, &p_one));

    // add a signal
    CHK(l4shmc_add_signal(&shmarea, "prod", &s_one));

    CHK(l4shmc_attach_signal_to(&shmarea, "done",
                                pthread_getl4cap(pthread_self()), 10000, &s_done));

    // connect chunk and signal
    CHK(l4shmc_connect_chunk_signal(&p_one, &s_one));

    printf("PRODUCER: ready\n");

    while (1)
    {
        while (l4shmc_chunk_try_to_take(&p_one))
            printf("Uh, should not happen!\n"); //l4_thread_yield();

        memcpy(l4shmc_chunk_ptr(&p_one), some_data, sizeof(some_data));
        CHK(l4shmc_chunk_ready_sig(&p_one, sizeof(some_data)));
        printf("PRODUCER: Sent data\n");
        CHK(l4shmc_wait_signal(&s_done));
    }
}

```

```

    }

    l4_sleep_forever();
    return NULL;
}

static void *thread_consume(void *d)
{
    (void)d;
    l4shmc_area_t shmarea;
    l4shmc_chunk_t p_one;
    l4shmc_signal_t s_one, s_done;

    // attach to shared memory area
    CHK(l4shmc_attach("testshm", &shmarea));

    // get chunk 'one'
    CHK(l4shmc_get_chunk(&shmarea, "one", &p_one));

    // add a signal
    CHK(l4shmc_add_signal(&shmarea, "done", &s_done));

    // attach signal to this thread
    CHK(l4shmc_attach_signal_to(&shmarea, "prod",
                                pthread_getl4cap(pthread_self()), 10000, &s_one));

    // connect chunk and signal
    CHK(l4shmc_connect_chunk_signal(&p_one, &s_one));

    while (1)
    {
        CHK(l4shmc_wait_chunk(&p_one));

        printf("CONSUMER: Received from chunk one: %s\n",
               (char *)l4shmc_chunk_ptr(&p_one));
        memset(l4shmc_chunk_ptr(&p_one), 0, l4shmc_chunk_size(&p_one));

        CHK(l4shmc_chunk_consumed(&p_one));
        CHK(l4shmc_trigger(&s_done));
    }

    return NULL;
}

int main(void)
{
    pthread_t one, two;

    // create new shared memory area, 8K in size
    if (l4shmc_create("testshm", 8192))
        return 1;

    // create two threads, one for producer, one for consumer
    pthread_create(&one, 0, thread_producer, 0);
    pthread_create(&two, 0, thread_consume, 0);

    // now sleep, the two threads are doing the work
    l4_sleep_forever();

    return 0;
}

```

## 12.15 examples/sys/aliens/main.c

This example shows how system call tracing can be done.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *                  Alexander Warg <warg@os.inf.tu-dresden.de>,
 *                  BjÄurn DÄubel <doebel@os.inf.tu-dresden.de>
 *      economic rights: Technische UniversitÄt Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * Example to show syscall tracing.
 */
//#define MEASURE

#include <l4/sys/ipc.h>
#include <l4/sys/thread.h>
#include <l4/sys/factory.h>
#include <l4/sys/utcb.h>
#include <l4/sys/kdebug.h>
#include <l4/util/util.h>
#include <l4/util/rdtsc.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>
#include <l4/sys/debugger.h>

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

static char alien_thread_stack[8 << 10];
static l4_cap_idx_t alien;

static void alien_thread(void)
{
    volatile l4_mshtag_t x;
    while (1) {
        x = l4_ipc_call(0x1234 << L4_CAP_SHIFT, l4_utcb(), l4_mshtag(0, 0, 0, 0),
                        L4_IPC_NEVER);
    #ifdef MEASURE
        l4_sleep(0);
    #else
        l4_sleep(1000);
        outstring("An int3 -- you should see this\n");
        outnstring("345", 3);
    #endif
    }
}

int main(void)
{
    l4_mshtag_t tag;
    #ifdef MEASURE
        l4_cpu_time_t s, e;
    #endif
    l4_utcb_t *u = l4_utcb();
    l4_exc_regs_t exc;
    l4_umword_t mr0, mr1;

    printf("Alien feature testing\n");

```

```

l4_debugger_set_object_name(l4re_env()->main_thread, "alientest");

/* Start alien thread */
if (l4_is_invalid_cap(alien = l4re_util_cap_alloc()))
    return 1;

l4_touch_rw(alien_thread_stack, sizeof(alien_thread_stack));

tag = l4_factory_create_thread(l4re_env()>factory, alien);
if (l4_mshtag_has_error(tag))
    return 1;

l4_debugger_set_object_name(alien, "alienth");

l4_thread_control_start();
l4_thread_control_pager(l4re_env()>main_thread);
l4_thread_control_exc_handler(l4re_env()>main_thread);
l4_thread_control_bind((l4_utcb_t *)l4re_env()>first_free_utcb, L4RE_THIS_TASK
    _CAP);
l4_thread_control_alien(1);
tag = l4_thread_control_commit(alien);
if (l4_mshtag_has_error(tag))
    return 2;

tag = l4_thread_ex_regs(alien,
    (l4_umword_t)alien_thread,
    (l4_umword_t)alien_thread_stack + sizeof(alien_thread_s
    tack),
    0);
if (l4_mshtag_has_error(tag))
    return 3;

#ifndef MEASURE
    l4_calibrate_tsc(l4re_kip());
#endif

/* Pager/Exception loop */
if (l4_mshtag_has_error(tag = l4_ipc_receive(alien, u, L4_IPC_NEVER)))
{
    printf("l4_ipc_receive failed");
    return 1;
}

memcpy(&exc, l4_utcb_exc(), sizeof(exc));
mr0 = l4_utcb_mr()>mr[0];
mr1 = l4_utcb_mr()>mr[1];

for (i++)
{
#ifdef MEASURE
    s = l4_rdtsc();
#endif

    if (l4_mshtag_is_exception(tag))
    {
#ifndef MEASURE
        printf("PC=%08lx SP=%08lx Err=%08lx Trap=%lx, %s syscall, SC-Nr: %lx\n"
        ,
            l4_utcb_exc_pc(&exc), exc.sp, exc.err,
            exc.trapno, (exc.err & 4) ? " after" : "before",
            exc.err >> 3);
#endif
    }
    tag = l4_mshtag((exc.err & 4) ? 0 : L4_PROTO_ALLOW_SYSCALL,
                    L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);
}
else

```

```

printf("Umm, non-handled request (like PF): %lx %lx\n", mr0, mr1);

memcpy(&exc, &exc, sizeof(exc));

/* Reply and wait */
if (l4_mshtag_has_error(tag = l4_ipc_call(alien, u, tag, L4_IPC_NEVER)))
{
    printf("l4_ipc_call failed\n");
    return 1;
}
memcpy(&exc, l4_utcb_exc(), sizeof(exc));
mr0 = l4_utcb_mr() ->mr[0];
mr1 = l4_utcb_mr() ->mr[1];

#endif MEASURE
e = l4_rdtsc();
printf("time %lld\n", l4_tsc_to_ns(e - s));
#endif
}

return 0;
}

```

## 12.16 examples/sys/ipc/ipc.cfg

Sample configuration file for the IPC example.

```

# vim:se ft=lua:

require("L4");

L4.default_loader:start({}, "rom/ex_ipcl");

```

## 12.17 examples/sys/ipc/ipc\_example.c

This example shows how two threads can exchange data using the L4 IPC mechanism. One thread is sending an integer to the other thread which is returning the square of the integer. Both values are printed.

```

/*
 * (c) 2008-2009 Author(s)
 *      economic rights: Technische Universitt Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/ipc.h>

#include <pthread-l4.h>
#include <unistd.h>
#include <stdio.h>

static pthread_t t2;

/* Thread1 is the initiator thread, i.e. it initiates the IPC calls. In
 * other words, it takes the client role. It uses L4 IPC mechanisms to send
 * an integer value to thread2 and received a calculation result back. */
static void *thread1_fn(void *arg)
{
    l4_mshtag_t tag;

```

```

int ipc_error;
unsigned long value = 1;
(void)arg;

while (1)
{
    printf("Sending: %ld\n", value);

    /* Store the value which we want to have squared in the first message
     * register of our UTCB. */
    l4_utcb_mr()>mr[0] = value;

    /* To an L4 IPC call, i.e. send a message to thread2 and wait for a
     * reply from thread2. The '1' in the msgtag denotes that we want to
     * transfer one word of our message registers (i.e. MRO). No timeout. */
    tag = l4_ipc_call(pthread_getl4cap(t2), l4_utcb(),
                      l4_mshtag(0, 1, 0, 0), L4_IPC_NEVER);
    /* Check for IPC error, if yes, print out the IPC error code, if not,
     * print the received result. */
    ipc_error = l4_ipc_error(tag, l4_utcb());
    if (ipc_error)
        fprintf(stderr, "thread1: IPC error: %x\n", ipc_error);
    else
        printf("Received: %ld\n", l4_utcb_mr()>mr[0]);

    /* Wait some time and increment our value. */
    sleep(1);
    value++;
}
return NULL;
}

/* Thread2 is in the server role, i.e. it waits for requests from others and
 * sends back the calculation results. */
static void *thread2_fn(void *arg)
{
    l4_mshtag_t tag;
    l4_umword_t label;
    int ipc_error;
    (void)arg;

    /* Wait for requests from any thread. No timeout, i.e. wait forever. */
    tag = l4_ipc_wait(l4_utcb(), &label, L4_IPC_NEVER);
    while (1)
    {
        /* Check if we had any IPC failure, if yes, print the error code
         * and just wait again. */
        ipc_error = l4_ipc_error(tag, l4_utcb());
        if (ipc_error)
        {
            fprintf(stderr, "thread2: IPC error: %x\n", ipc_error);
            tag = l4_ipc_wait(l4_utcb(), &label, L4_IPC_NEVER);
            continue;
        }

        /* So, the IPC was ok, now take the value out of message register 0
         * of the UTCB and store the square of it back to it. */
        l4_utcb_mr()>mr[0] = l4_utcb_mr()>mr[0] * l4_utcb_mr()>mr[0];

        /* Send the reply and wait again for new messages.
         * The '1' in the msgtag indicated that we want to transfer 1 word in
         * the message registers (i.e. MRO) */
        tag = l4_ipc_reply_and_wait(l4_utcb(), l4_mshtag(0, 1, 0, 0),
                                    &label, L4_IPC_NEVER);
    }
    return NULL;
}

```

```

int main(void)
{
    // We will have two threads, one is already running the main function, the
    // other (thread2) will be created using pthread_create.

    if (pthread_create(&t2, NULL, thread2_fn, NULL))
    {
        fprintf(stderr, "Thread creation failed\n");
        return 1;
    }

    // Just run thread1 in the main thread
    thread1_fn(NULL);
    return 0;
}

```

## 12.18 examples/sys/isr/main.c

Example of an interrupt service routine.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>,
 *      BjÃrn DÃbel <doebel@os.inf.tu-dresden.de>
 *      economic rights: Technische UniversitÃ¤t Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * This example shall show how to connect to an interrupt, receive interrupt
 * events and detach again. As the interrupt source we'll use the virtual
 * key interrupt. The interrupt number of the virtual key interrupt can be
 * found in the kernel info page.
 */

#include <l4/re/c/util/cap_alloc.h>
#include <l4/re/c/namespace.h>
#include <l4/sys/utcb.h>
#include <l4/sys/irq.h>
#include <l4/sys/factory.h>
#include <l4/sys/icu.h>

#include <stdio.h>

int main(void)
{
    int irqno = 1;
    l4_cap_idx_t irqcap, icucap;
    l4_mshtag_t tag;
    int err;
    icucap = l4re_get_env_cap("icu");

    /* Get a free capability slot for the ICU capability */
    if (l4_is_invalid_cap(icucap))
    {
        printf("Did not find an ICU\n");
        return 1;
    }

    /* Get another free capability slot for the corresponding IRQ object*/
    if (l4_is_invalid_cap(irqcap = l4re_util_cap_alloc()))

```

```

        return 1;
/* Create IRQ object */
if (l4_error(tag = l4_factory_create_irq(l4re_global_env->factory, irqcap)))
{
    printf("Could not create IRQ object: %lx\n", l4_error(tag));
    return 1;
}

/*
 * Bind the recently allocated IRQ object to the IRQ number irqno
 * as provided by the ICU.
*/
if (l4_error(l4_icu_bind(icucap, irqno, irqcap)))
{
    printf("Binding IRQ%d to the ICU failed\n", irqno);
    return 1;
}

/* Attach ourselves to the IRQ */
tag = l4_irq_attach(irqcap, 0xDEAD, l4re_env()->main_thread);
if ((err = l4_error(tag)))
{
    printf("Error attaching to IRQ %d: %d\n", irqno, err);
    return 1;
}

printf("Attached to key IRQ %d\nPress keys now, Shift-Q to exit\n", irqno);

/* IRQ receive loop */
while (1)
{
    unsigned long label = 0;
    /* Wait for the interrupt to happen */
    tag = l4_irq_receive(irqcap, L4_IPC_NEVER);
    if ((err = l4_ipc_error(tag, l4_utcb())))
        printf("Error on IRQ receive: %d\n", err);
    else
    {
        /* Process the interrupt -- may do a 'break' */
        printf("Got IRQ with label 0x%lx\n", label);
    }
}

/* We're done, detach from the interrupt. */
tag = l4_irq_detach(irqcap);
if ((err = l4_error(tag)))
    printf("Error detach from IRQ: %d\n", err);

return 0;
}

```

## 12.19 examples/sys/migrate/thread\_migrate.cc

Thread migration example.

```

/*
 * (c) 2008-2009 Author(s)
 *      economic rights: Technische Universitt Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */

```

```

#include <l4/sys/scheduler>
#include <l4/re/env>
#include <l4/re/util/cap_alloc>

#include <pthread-14.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>

enum { NR_THREADS = 12 };
static L4::Cap<L4::Thread> threads[NR_THREADS];
static l4_umword_t           cpu_map, cpu_nrs;

/* Function for the threads. The content is not really relevant, so lets
 * just sleep around a bit. */
static void *thread_fn(void *)
{
    while (1)
        sleep(1);

    return 0;
}

/* Check how many CPUs we have available.
 */
static int check_cpus(void)
{
    l4_sched_cpu_set_t cs = l4_sched_cpu_set(0, 0);

    if (l4_error(L4Re::Env::env()>scheduler()->info(&cpu_nrs, &cs)) < 0)
        return 1;

    cpu_map = cs.map;

    printf("Found %d CPUs.\n", cpu_nrs);
    if (cpu_nrs >= L4_MWORD_BITS)
    {
        printf("Will only handle %d CPUs.\n", cpu_nrs);
        cpu_nrs = L4_MWORD_BITS;
    }
    else if (cpu_nrs == 1)
        printf("Only found 1 CPU.\n");

    return cpu_nrs < 2;
}

/* Create a couple of threads and store their capabilities in an array */
static int create_threads(void)
{
    unsigned i;

    for (i = 0; i < NR_THREADS; ++i)
    {
        pthread_t t;

        if (pthread_create(&t, NULL, thread_fn, NULL))
            return 1;

        threads[i] = L4::Cap<L4::Thread>(pthread_getl4cap(t));
    }
    printf("Created %d threads.\n", NR_THREADS);
    return 0;
}

/* Helper function to get the next CPU */
static unsigned get_next_cpu(unsigned c)
{

```

```

unsigned x = c;
for (;;)
{
    x = (x + 1) % cpu_nrs;
    if (L4Re::Env::env()>scheduler()->is_online(x))
        return x;
    if (x == c)
        return c;
}
}

/* Function that shuffles the threads on the available CPUs */
static void shuffle(void)
{
    unsigned start = 0;
    while (1)
    {
        unsigned t;
        unsigned c = start;
        for (t = 0; t < NR_THREADS; ++t)
        {
            l4_sched_param_t sp = l4_sched_param(20);
            c = get_next_cpu(c);
            sp.affinity = l4_sched_cpu_set(c, 0);
            if (l4_error(L4Re::Env::env()>scheduler()->run_thread(threads[t], sp)))
                printf("Error migrating thread%02d to CPU%02d\n", t, c);
            printf("Migrated Thread%02d -> CPU%02d\n", t, c);
        }
        start++;
        if (start == cpu_nrs)
            start = 0;
        sleep(1);
    }
}

int main(void)
{
    if (check_cpus())
        return 1;

    if (create_threads())
        return 1;

    shuffle();

    return 0;
}

```

## 12.20 examples/sys/migrate/thread\_migrate.cfg

Sample configuration file for the thread migration example.

```
-- vim:set ft=lua:

-- The log prefix will be 'migrate', colored green.
L4.default_loader:start({ log = { "migrate", "green" } },
                        "rom/ex_thread_migrate");
```

## 12.21 examples/sys/singlestep/main.c

This example shows how a thread can be single stepped on the x86 architecture.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *                  Alexander Warg <warg@os.inf.tu-dresden.de>,
 *                  BjÄurn DÄbel <doebel@os.inf.tu-dresden.de>
 *      economic rights: Technische UniversitÄt Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * Single stepping example for the x86-32 architecture.
 */
#include <l4/sys/ipc.h>
#include <l4/sys/factory.h>
#include <l4/sys/thread.h>
#include <l4/sys/utcb.h>
#include <l4/sys/kdebug.h>

#include <l4/util/util.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

static char thread_stack[8 << 10];

static void thread_func(void)
{
    while (1)
    {
        unsigned long d = 0;

        /* Enable single stepping */
        asm volatile("pushf; pop %0; or $256,%0; push %0; popf\n"
                   : "=r" (d) : "r" (d));

        /* Some instructions */
        asm volatile("nop");
        asm volatile("nop");
        asm volatile("nop");
        asm volatile("mov $12345678, %%edx" : : : "edx"); // a non-existant cap
        asm volatile("int $0x30\n");
        asm volatile("nop");
        asm volatile("nop");
        asm volatile("nop");

        /* Disabled single stepping */
        asm volatile("pushf; pop %0; and $~256,%0; push %0; popf\n"
                   : "=r" (d) : "r" (d));

        /* You won't see those */
        asm volatile("nop");
        asm volatile("nop");
        asm volatile("nop");
    }
}

int main(void)
{
```

```

l4_mshtag_t tag;
int ipc_stat = 0;
l4_cap_idx_t th = l4re_util_cap_alloc();
l4_exc_regs_t exc;
l4_umword_t mr0, mrl;
l4_utcb_t *u = l4_utcb();

printf("Singlestep testing\n");

if (l4_is_invalid_cap(th))
    return 1;

l4_touch_rw(thread_stack, sizeof(thread_stack));
l4_touch_ro(thread_func, 1);

tag = l4_factory_create_thread(l4re_env()>factory, th);
if (l4_mshtag_has_error(tag))
    return 1;

l4_thread_control_start();
l4_thread_control_pager(l4re_env()>main_thread);
l4_thread_control_exc_handler(l4re_env()>main_thread);
l4_thread_control_bind((l4_utcb_t *)l4re_env()>first_free_utcb,
                      L4RE_THIS_TASK_CAP);
l4_thread_control_alien(1);
tag = l4_thread_control_commit(th);
if (l4_mshtag_has_error(tag))
    return 2;

tag = l4_thread_ex_regs(th, (l4_umword_t)thread_func,
                        (l4_umword_t)thread_stack + sizeof(thread_stack),
                        0);
if (l4_mshtag_has_error(tag))
    return 3;

/* Pager/Exception loop */
if (l4_mshtag_has_error(tag = l4_ipc_receive(th, u, L4_IPC_NEVER)))
{
    printf("l4_ipc_receive failed");
    return 4;
}
memcpy(&exc, l4_utcb_exc(), sizeof(exc));
mr0 = l4_utcb_mr()>mr[0];
mrl = l4_utcb_mr()>mr[1];

for (;;)
{
    if (l4_mshtag_is_exception(tag))
    {
        printf("PC = %08lx Trap = %08lx Err = %08lx, SP = %08lx SC-Nr: %lx\n",
               l4_utcb_exc_pc(&exc), exc.trapno, exc.err,
               exc.sp, exc.err >> 3);
        if (exc.err >> 3)
        {
            if (!(exc.err & 4))
            {
                tag = l4_mshtag(L4_PROTO_ALLOW_SYSCALL,
                                L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);
                if (ipc_stat)
                    enter_kdebug("Should not be 1");
            }
            else
            {
                tag = l4_mshtag(L4_PROTO_NONE,
                                L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);
                if (!ipc_stat)
                    enter_kdebug("Should not be 0");
            }
        }
    }
}

```

```

        }
        ipc_stat = !ipc_stat;
    }
    l4_sleep(100);
}
else
printf("Umm, non-handled request: %ld, %08lx %08lx\n",
       l4_mshtag_label(tag), mr0, mr1);

memcpy(l4_utcb_exc(), &exc, sizeof(exc));

/* Reply and wait */
if (l4_mshtag_has_error(tag = l4_ipc_call(th, u, tag, L4_IPC_NEVER)))
{
    printf("l4_ipc_call failed\n");
    return 5;
}
memcpy(&exc, l4_utcb_exc(), sizeof(exc));
mr0 = l4_utcb_mr()->mr[0];
mr1 = l4_utcb_mr()->mr[1];
}

return 0;
}

```

## 12.22 examples/sys/start-with-exc/main.c

This example shows how to start a newly created thread with a defined set of CPU registers.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>,
 *      BjÃ¼rn DÃ¶bel <doebel@os.inf.tu-dresden.de>,
 *      Frank Mehnert <fm3@os.inf.tu-dresden.de>
 *      economic rights: Technische UniversitÃ¤t Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
/*
 * Start a thread with an exception reply. This example does only work on
 * the x86-32 architecture.
 */

#include <l4/sys/thread.h>
#include <l4/sys/factory.h>
#include <l4/sys/ipc.h>
#include <l4/sys/utcb.h>
#include <l4/util/util.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>

#include <stdlib.h>
#include <stdio.h>

/* Stack for the thread to be created. 8kB are enough. */
static char thread_stack[8 << 10];

/* The thread to be created. For illustration it will print out its
 * register set.
 */
static void L4_STICKY(thread_func(l4_umword_t *d))
{

```

```

while (1)
{
    printf("hey, I'm a thread\n");
    printf("got register values: %ld %ld %ld %ld %ld %ld %ld\n",
           d[7], d[6], d[5], d[4], d[2], d[1], d[0]);
    l4_sleep(800);
}

/*
 * Startup trick for this example. Put all the CPU registers on the stack so
 * that the C function above can get it on the stack. */
asm(
".global thread      \n\t"
"thread:      \n\t"
" pusha      \n\t"
" push %esp      \n\t"
" call thread_func \n\t"
);
extern void thread(void);

/* Our main function */
int main(void)
{
    /* Get a capability slot for our new thread. */
    l4_cap_idx_t t1 = l4re_util_cap_alloc();
    l4_utcb_t *u = l4_utcb();
    l4_exc_regs_t *e = l4_utcb_exc_u(u);
    l4_mshtag_t tag;
    int err;
    extern char _start[], _end[], _etext[];

    if (l4_is_invalid_cap(t1))
        return 1;

    /* Prevent pagefaults of our new thread because we do not want to
     * implement a pager as well. */
    l4_touch_ro(_start, _end - _start + 1);
    l4_touch_rw(_etext, _end - _etext);

    /* Create the thread using our default factory */
    tag = l4_factory_create_thread(l4re_env()>factory, t1);
    if (l4_mshtag_has_error(tag))
        return 1;

    /* Setup the thread by setting the pager and task. */
    l4_thread_control_start();
    l4_thread_control_pager(l4re_env()>main_thread);
    l4_thread_control_exc_handler(l4re_env()>main_thread);
    l4_thread_control_bind((l4_utcb_t *)l4re_env()>first_free_utcb,
                           L4RE_THIS_TASK_CAP);
    tag = l4_thread_control_commit(t1);
    if (l4_mshtag_has_error(tag))
        return 2;

    /* Start the thread by finally setting instruction and stack pointer */
    tag = l4_thread_ex_regs(t1,
                           (l4_umword_t)thread,
                           (l4_umword_t)thread_stack + sizeof(thread_stack),
                           L4_THREAD_EX_REGS_TRIGGER_EXCEPTION);
    if (l4_mshtag_has_error(tag))
        return 3;

    /* Receive initial exception from just started thread */
    tag = l4_ipc_receive(t1, u, L4_IPC_NEVER);
    if ((err = l4_ipc_error(tag, u)))
    {
        printf("Umm, ipc error: %x\n", err);
}

```

```

        return 1;
    }
/* We expect an exception IPC */
if (!l4_mshtag_is_exception(tag))
{
    printf("PF?: %lx %lx (not prepared to handle this) %ld\n",
        l4_utcb_mr_u(u)->mr[0], l4_utcb_mr_u(u)->mr[1], l4_mshtag_label(tag));
    return 1;
}

/* Fill out the complete register set of the new thread */
e->ip = (l4_umword_t)thread;
e->sp = (l4_umword_t)(thread_stack + sizeof(thread_stack));
e->eax = 1;
e->ebx = 4;
e->ecx = 2;
e->edx = 3;
e->esi = 6;
e->edi = 7;
e->ebp = 5;
/* Send a complete exception */
tag = l4_mshtag(0, L4_UTCB_EXCEPTION_REGS_SIZE, 0, 0);

/* Send reply and start the thread with the defined CPU register set */
tag = l4_ipc_send(t1, u, tag, L4_IPC_NEVER);
if ((err = l4_ipc_error(tag, u)))
    printf("Error sending IPC: %x\n", err);

/* Idle around */
while (1)
    l4_sleep(10000);

return 0;
}

```

## 12.23 examples/sys/utcb-ipc/main.c

This example shows how to send IPC using the UTCB to store payload.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>,
 *      BjÃ¶rn DÃ¶bel <doebel@os.inf.tu-dresden.de>
 *      economic rights: Technische UniversitÃ¤t Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/ ipc.h>
#include <l4/sys/thread.h>
#include <l4/sys/factory.h>
#include <l4/sys/utcb.h>
#include <l4/re/env.h>
#include <l4/re/c/util/cap_alloc.h>

#include <stdio.h>
#include <string.h>

static unsigned char stack2[8 << 10];
static l4_cap_idx_t thread1_cap, thread2_cap;

static void thread1(void)

```

```

{
    14_msg_regs_t *mr = 14_utcb_mr();
    14_mshtag_t tag;
    int i, j;

    printf("Thread1 up (%p)\n", 14_utcb());

    for (i = 0; i < 10; i++)
    {
        for (j = 0; j < L4_UTCB_GENERIC_DATA_SIZE; j++)
            mr->mr[j] = 'A' + (i + j) % ('~' - 'A' + 1);
        tag = 14_mshtag(0, L4_UTCB_GENERIC_DATA_SIZE, 0, 0);
        if (14_mshtag_has_error(14_ipc_send(thread2_cap, 14_utcb(), tag,
                                             L4_IPC_NEVER)))
            printf("IPC-send error\n");
    }
}

static void thread2(void)
{
    14_mshtag_t tag;
    14_msg_regs_t mr;
    unsigned i;

    printf("Thread2 up (%p)\n", 14_utcb());

    while (1)
    {
        if (14_mshtag_has_error(tag = 14_ipc_receive(thread1_cap, 14_utcb(),
                                                       L4_IPC_NEVER)))
            printf("IPC receive error\n");
        memcpy(&mr, 14_utcb_mr(), sizeof(mr));
        printf("Thread2 receive (%d): ", 14_mshtag_words(tag));
        for (i = 0; i < 14_mshtag_words(tag); i++)
            printf("%c", (char)mr.mr[i]);
        printf("\n");
    }
}

int main(void)
{
    14_mshtag_t tag;

    thread1_cap = 14re_env()->main_thread;
    thread2_cap = 14re_util_cap_alloc();

    if (14_is_invalid_cap(thread2_cap))
        return 1;

    tag = 14_factory_create_thread(14re_env()->factory, thread2_cap);
    if (14_mshtag_has_error(tag))
        return 1;

    14_thread_control_start();
    14_thread_control_pager(14re_env()->rm);
    14_thread_control_exc_handler(14re_env()->rm);
    14_thread_control_bind((14_utcb_t *)14re_env()->first_free_utcb,
                           L4RE_THIS_TASK_CAP);
    tag = 14_thread_control_commit(thread2_cap);
    if (14_mshtag_has_error(tag))
        return 2;

    tag = 14_thread_ex_regs(thread2_cap,
                            (14_umword_t)thread2,
                            (14_umword_t)(stack2 + sizeof(stack2)), 0);
    if (14_mshtag_has_error(tag))
        return 3;
}

```

```

    thread1();

    return 0;
}

```

## 12.24 examples/sys/ux-vhw/main.c

This example shows how to iterate the virtual hardware descriptors under Fiasco-UX.

```

/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische Universitt Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <l4/sys/ipc.h>
#include <l4/sys/vhw.h>
#include <l4/util/util.h>
#include <l4/util/kip.h>
#include <l4/re/env.h>

#include <stdlib.h>
#include <stdio.h>

static void print_entry(struct l4_vhw_entry *e)
{
    printf("type: %d mem start: %08lx end: %08lx\n"
           "irq: %d pid %d\n",
           e->type, e->mem_start, e->mem_size,
           e->irq_no, e->provider_pid);
}

int main(void)
{
    l4_kernel_info_t *kip = l4re_kip();
    struct l4_vhw_descriptor *vhw;
    int i;

    if (!kip)
    {
        printf("KIP not available!\n");
        return 1;
    }

    vhw = l4_vhw_get(kip);

    printf("kip at %p, vhw at %p\n", kip, vhw);
    printf("magic: %08x, version: %08x, count: %02d\n",
           vhw->magic, vhw->version, vhw->count);

    for (i = 0; i < vhw->count; i++)
        print_entry(l4_vhw_get_entry(vhw, i));

    return 0;
}

```

## 12.25 hello/server/src/main.c

This is the famous "Hello World!" program.

```
/*
 * (c) 2008-2009 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Frank Mehnert <fm3@os.inf.tu-dresden.de>,
 *      Lukas GrÄijtzmacher <lg2@os.inf.tu-dresden.de>
 *      economic rights: Technische UniversitÄt Dresden (Germany)
 *
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU General Public License 2.
 * Please see the COPYING-GPL-2 file for details.
 */
#include <stdio.h>
#include <unistd.h>

int
main(void)
{
    for (;;)
    {
        puts("Hello World!");
        sleep(1);
    }
}
```

## 12.26 tmpfs/lib/src/fs.cc

Example file system for L4Re::Vfs.

```
/*
 * (c) 2010 Adam Lackorzynski <adam@os.inf.tu-dresden.de>,
 *      Alexander Warg <warg@os.inf.tu-dresden.de>
 *      economic rights: Technische UniversitÄt Dresden (Germany)
 * This file is part of TUD:OS and distributed under the terms of the
 * GNU Lesser General Public License 2.1.
 * Please see the COPYING-LGPL-2.1 file for details.
 */

#include <l4/l4re_vfs/backend>
#include <l4/cxx/string>
#include <l4/cxx/avl_tree>

#include <sys/stat.h>
#include <sys/ioctl.h>
#include <dirent.h>

#include <cstdio>

namespace {

using namespace L4Re::Vfs;
using cxx::Ref_ptr;

class File_data
{
public:
    File_data() : _buf(0), _size(0) {}

    unsigned long put(unsigned long offset,
                      unsigned long bufsize, void *srcbuf);
```

```

unsigned long get(unsigned long offset,
                  unsigned long bufsize, void *dstbuf);

unsigned long size() const { return _size; }

~File_data() throw() { free(_buf); }

private:
    void *_buf;
    unsigned long _size;
};

unsigned long
File_data::put(unsigned long offset, unsigned long bufsize, void *srcbuf)
{
    if (offset + bufsize > _size)
    {
        _size = offset + bufsize;
        _buf = realloc(_buf, _size);
    }

    if (!_buf)
        return 0;

    memcpy((char *)_buf + offset, srcbuf, bufsize);
    return bufsize;
}

unsigned long
File_data::get(unsigned long offset, unsigned long bufsize, void *dstbuf)
{
    unsigned long s = bufsize;

    if (offset > _size)
        return 0;

    if (offset + bufsize > _size)
        s = _size - offset;

    memcpy(dstbuf, (char *)_buf + offset, s);
    return s;
}

class Node : public cxx::Avl_tree_node
{
public:
    Node(const char *path, mode_t mode)
        : _ref_cnt(0), _path(strdup(path))
    {
        memset(&_info, 0, sizeof(_info));
        _info.st_mode = mode;
    }

    const char *path() const { return _path; }
    struct stat64 *info() { return &_info; }

    void add_ref() throw() { ++_ref_cnt; }
    int remove_ref() throw() { return --_ref_cnt; }

    bool is_dir() const { return S_ISDIR(_info.st_mode); }

    virtual ~Node() { free(_path); }

private:
    int             _ref_cnt;
    char           *_path;
};

```

```

    struct stat64 _info;
};

struct Node_get_key
{
    typedef cxx::String Key_type;
    static Key_type key_of(Node const *n)
    { return n->path(); }
};

struct Path_avl_tree_compare
{
    bool operator () (const char *l, const char *r) const
    { return strcmp(l, r) < 0; }
    bool operator () (const cxx::String l, const cxx::String r) const
    { return strncmp(l.start(), r.start(), cxx::min(l.len(), r.len())) < 0; }
};

class Pers_file : public Node
{
public:
    Pers_file(const char *name, mode_t mode)
        : Node(name, (mode & 0777) | __S_IFREG) {}
    File_data const &data() const { return _data; }
    File_data &data() { return _data; }
private:
    File_data _data;
};

class Pers_dir : public Node
{
private:
    typedef cxx::Avl_tree<Node, Node_get_key, Path_avl_tree_compare> Tree;
    Tree _tree;

public:
    Pers_dir(const char *name, mode_t mode)
        : Node(name, (mode & 0777) | __S_IFDIR) {}
    Ref_ptr<Node> find_path(cxx::String);
    bool add_node(Ref_ptr<Node> const &);

    typedef Tree::Const_iterator Const_iterator;
    Const_iterator begin() const { return _tree.begin(); }
    Const_iterator end() const { return _tree.end(); }
};

Ref_ptr<Node> Pers_dir::find_path(cxx::String path)
{
    return cxx::ref_ptr(_tree.find_node(path));
}

bool Pers_dir::add_node(Ref_ptr<Node> const &n)
{
    bool e = _tree.insert(n.ptr()).second;
    if (e)
        n->add_ref();
    return e;
}

class Tmpfs_dir : public Be_file
{
public:
    explicit Tmpfs_dir(Ref_ptr<Pers_dir> const &d) throw()
        : _dir(d), _getdents_state(false) {}
    int get_entry(const char *, int, mode_t, Ref_ptr<File> *) throw();
    ssize_t getdents(char *, size_t) throw();
    int fstat64(struct stat64 *buf) const throw();
}

```

```

int utime(const struct utimbuf * throw());
int fchmod(mode_t) throw();
int mkdir(const char *, mode_t) throw();
int unlink(const char *) throw();
int rename(const char *, const char *) throw();

private:
    int walk_path(cxx::String const &_s,
                  Ref_ptr<Node> *ret, cxx::String *remaining = 0);

    Ref_ptr<Pers_dir> _dir;
    bool _getdents_state;
    Pers_dir::Const_iterator _getdents_iter;
};

class Tmpfs_file : public Be_file_pos
{
public:
    explicit Tmpfs_file(Ref_ptr<Pers_file> const &f) throw()
        : Be_file_pos(), _file(f) {}

    off64_t size() const throw();
    int fstat64(struct stat64 *buf) const throw();
    int ioctl(unsigned long, va_list) throw();
    int utime(const struct utimbuf *) throw();
    int fchmod(mode_t) throw();

private:
    ssize_t preadv(const struct iovec *v, int iovcnt, off64_t p) throw();
    ssize_t pwritev(const struct iovec *v, int iovcnt, off64_t p) throw();
    Ref_ptr<Pers_file> _file;
};

ssize_t Tmpfs_file::preadv(const struct iovec *v, int iovcnt, off64_t p) throw()
{
    if (iovcnt < 0)
        return -EINVAL;

    ssize_t sum = 0;
    for (int i = 0; i < iovcnt; ++i)
    {
        sum += _file->data().get(p, v[i].iov_len, v[i].iov_base);
        p += v[i].iov_len;
    }
    return sum;
}

ssize_t Tmpfs_file::pwritev(const struct iovec *v, int iovcnt, off64_t p) throw()
{
    if (iovcnt < 0)
        return -EINVAL;

    ssize_t sum = 0;
    for (int i = 0; i < iovcnt; ++i)
    {
        sum += _file->data().put(p, v[i].iov_len, v[i].iov_base);
        p += v[i].iov_len;
    }
    return sum;
}

int Tmpfs_file::fstat64(struct stat64 *buf) const throw()
{
    _file->info()->st_size = _file->data().size();
    memcpy(buf, _file->info(), sizeof(*buf));
    return 0;
}

```

```

}

off64_t Tmpfs_file::size() const throw()
{ return _file->data().size(); }

int
Tmpfs_file::ioctl(unsigned long v, va_list args) throw()
{
    switch (v)
    {
        case FIONREAD: // return amount of data still available
            int *available = va_arg(args, int *);
            *available = _file->data().size() - pos();
            return 0;
    };
    return -EINVAL;
}

int
Tmpfs_file::utime(const struct utimbuf *times) throw()
{
    _file->info()->st_atime = times->actime;
    _file->info()->st_mtime = times->modtime;
    return 0;
}

int
Tmpfs_file::fchmod(mode_t m) throw()
{
    _file->info()->st_mode = m;
    return 0;
}

int
Tmpfs_dir::get_entry(const char *name, int flags, mode_t mode,
                     Ref_ptr<File> *file) throw()
{
    Ref_ptr<Node> path;
    if (!*name)
    {
        *file = this;
        return 0;
    }

    cxx::String n = name;

    int e = walk_path(n, &path, &n);

    if (e == -ENOTDIR)
        return e;

    if (!(flags & O_CREAT) && e < 0)
        return e;

    if ((flags & O_CREAT) && e == -ENOENT)
    {
        Ref_ptr<Node> node(new Pers_file(n.start(), mode));
        // when ENOENT is return, path is always a directory
        bool e = cxx::ref_ptr_static_cast<Pers_dir>(path)->add_node(node);
        if (!e)
            return -ENOMEM;
        path = node;
    }

    if (path->is_dir())
        *file = new Tmpfs_dir(cxx::ref_ptr_static_cast<Pers_dir>(path));
}

```

```

else
    *file = new Tmpfs_file(cxx::ref_ptr_static_cast<Pers_file>(path));

if (!*file)
    return -ENOMEM;

return 0;
}

ssize_t
Tmpfs_dir::getdents(char *buf, size_t sz) throw()
{
    struct dirent64 *d = (struct dirent64 *)buf;
    ssize_t ret = 0;

    if (!_getdents_state)
    {
        _getdents_iter = _dir->begin();
        _getdents_state = true;
    }
    else if (_getdents_iter == _dir->end())
    {
        _getdents_state = false;
        return 0;
    }

    for (; _getdents_iter != _dir->end(); ++_getdents_iter)
    {
        unsigned l = strlen(_getdents_iter->path()) + 1;
        if (l > sizeof(d->d_name))
            l = sizeof(d->d_name);

        unsigned n = offsetof(struct dirent64, d_name) + l;
        n = (n + sizeof(long) - 1) & ~(sizeof(long) - 1);

        if (n > sz)
            break;

        d->d_ino = 1;
        d->d_off = 0;
        memcpy(d->d_name, _getdents_iter->path(), l);
        d->d_reclen = n;
        d->d_type = DT_REG;
        ret += n;
        sz -= n;
        d = (struct dirent64 *)((unsigned long)d + n);
    }

    return ret;
}

int
Tmpfs_dir::fstat64(struct stat64 *buf) const throw()
{
    memcpy(buf, _dir->info(), sizeof(*buf));
    return 0;
}

int
Tmpfs_dir::utime(const struct utimbuf *times) throw()
{
    _dir->info()->st_atime = times->actime;
    _dir->info()->st_mtime = times->modtime;
    return 0;
}

```

```

int
Tmpfs_dir::fchmod(mode_t m) throw()
{
    _dir->info()->st_mode = m;
    return 0;
}

int
Tmpfs_dir::walk_path(cxx::String const &s,
                      Ref_ptr<Node> *ret, cxx::String *remaining)
{
    Ref_ptr<Pers_dir> p = _dir;
    cxx::String s = _s;
    Ref_ptr<Node> n;

    while (1)
    {
        if (s.len() == 0)
        {
            *ret = p;
            return 0;
        }

        cxx::String::Index sep = s.find("/");
        if (sep - s.start() == 1 && *s.start() == '.')
        {
            s = s.substr(s.start() + 2);
            continue;
        }

        n = p->find_path(s.head(sep - s.start()));
        if (!n)
        {
            *ret = p;
            if (remaining)
                *remaining = s.head(sep - s.start());
            return -ENOENT;
        }

        if (sep == s.end())
        {
            *ret = n;
            return 0;
        }

        if (!n->is_dir())
            return -ENOTDIR;

        s = s.substr(sep + 1);
        p = cxx::ref_ptr_static_cast<Pers_dir>(n);
    }

    *ret = n;
    return 0;
}

int
Tmpfs_dir::mkdir(const char *name, mode_t mode) throw()
{
    Ref_ptr<Node> node = _dir;
    cxx::String p = cxx::String(name);
    cxx::String path, last = p;
}

```

```

cxx::String::Index s = p.rfind("/");
// trim '/'s at the end
while (p.len() && s == p.end() - 1)
{
    p.len(p.len() - 1);
    s = p.rfind("/");
}

//printf("MKDIR '%s' p=%p %p\n", name, p.start(), s);

if (s != p.end())
{
    path = p.head(s);
    last = p.substr(s + 1, p.end() - s);

    int e = walk_path(path, &node);
    if (e < 0)
        return e;
}

if (!node->is_dir())
    return -ENOTDIR;

// due to path walking we can end up with an empty name
if (p.len() == 0 || p == cxx::String("."))
    return 0;

Ref_ptr<Pers_dir> dnode = cxx::ref_ptr_static_cast<Pers_dir>(node);

Ref_ptr<Pers_dir> dir(new Pers_dir(last.start(), mode));
return dnode->add_node(dir) ? 0 : -EEXIST;
}

int
Tmpfs_dir::unlink(const char *name) throw()
{
    cxx::Ref_ptr<Node> n;

    int e = walk_path(name, &n);
    if (e < 0)
        return -ENOENT;

    printf("Unimplemented (if file exists): %s(%s)\n", __func__, name);
    return -ENOMEM;
}

int
Tmpfs_dir::rename(const char *old, const char *newn) throw()
{
    printf("Unimplemented: %s(%s, %s)\n", __func__, old, newn);
    return -ENOMEM;
}
}

class Tmpfs_fs : public Be_file_system
{
public:
    Tmpfs_fs() : Be_file_system("tmpfs") {}
    int mount(char const *source, unsigned long mountflags,
              void const *data, cxx::Ref_ptr<File> *dir) throw()
    {
        (void)mountflags;
        (void)source;
        (void)data;
        *dir = cxx::ref_ptr(new Tmpfs_dir(cxx::ref_ptr(new Pers_dir("root", 0777))));
}

```

```
    if (!*dir)
        return -ENOMEM;
    return 0;
};

static Tmpfs_fs _tmpfs_L4RE_VFS_FILE_SYSTEM_ATTRIBUTE;

}
```

# Index

~Auto\_ptr  
    cxx::Auto\_ptr, 417

~Be\_file\_system  
    L4Re::Vfs::Be\_file\_system, 461

\_c  
    L4::Cap\_base, 503

a  
    L4Re::Video::Pixel\_info, 804, 805

add  
    L4::Thread::Modify\_senders, 776  
    L4vcpu::State, 855

add\_ku\_mem  
    L4::Task, 868

alloc  
    cxx::Base\_slab\_static, 452  
    cxx::List\_alloc, 751  
    cxx::Slab, 842  
    cxx::Slab\_static, 845  
    L4Re::Cap\_alloc, 492, 493  
    L4Re::Mem\_alloc, 763

alloc\_fd  
    L4Re::Vfs::Fs, 606

allocate  
    L4Re::Dataspace, 520

amd64 Virtual Registers (UTCB), 282

api\_l4re\_c\_rm  
    L4RE\_RM\_ATTACH\_FLAGS, 64  
    L4RE\_RM\_EAGER\_MAP, 64  
    L4RE\_RM\_IN\_AREA, 64  
    L4RE\_RM\_NO\_ALIAS, 64  
    L4RE\_RM\_OVERMAP, 64  
    L4RE\_RM\_PAGER, 64  
    L4RE\_RM\_READ\_ONLY, 64  
    L4RE\_RM\_REGION\_FLAGS, 64  
    L4RE\_RM\_RESERVED, 64  
    L4RE\_RM\_SEARCH\_ADDR, 64

api\_l4re\_c\_video  
    F\_l4re\_video\_goops\_auto\_refresh, 77  
    F\_l4re\_video\_goops\_dynamic\_buffers, 77  
    F\_l4re\_video\_goops\_dynamic\_views, 77  
    F\_l4re\_video\_goops\_pointer, 77  
    F\_l4re\_video\_view\_above, 78  
    F\_l4re\_video\_view\_dyn\_allocated, 78  
    F\_l4re\_video\_view\_flags\_mask, 78

F\_l4re\_video\_view\_none, 78  
F\_l4re\_video\_view\_set\_background, 78  
F\_l4re\_video\_view\_set\_buffer, 78  
F\_l4re\_video\_view\_set\_buffer\_offset, 78  
F\_l4re\_video\_view\_set\_bytes\_per\_line, 78  
F\_l4re\_video\_view\_set\_flags, 78  
F\_l4re\_video\_view\_set\_pixel, 78  
F\_l4re\_video\_view\_set\_position, 78

api\_l4re\_elf\_aux  
    L4RE\_ELF\_AUX\_T\_KIP\_ADDR, 86  
    L4RE\_ELF\_AUX\_T\_NONE, 86  
    L4RE\_ELF\_AUX\_T\_STACK\_ADDR, 86  
    L4RE\_ELF\_AUX\_T\_STACK\_SIZE, 86  
    L4RE\_ELF\_AUX\_T\_VMA, 86

api\_l4re\_protocols  
    Dataspace, 96  
    Debug, 96  
    Default, 96  
    Event, 96  
    Goos, 96  
    Mem\_alloc, 96  
    Namespace, 96  
    Parent, 96  
    Rm, 96

api\_libvcpu  
    L4VCPU\_IRQ\_STATE\_DISABLED, 371  
    L4VCPU\_IRQ\_STATE\_ENABLED, 371

api\_calls\_fiasco  
    fiasco\_gdt\_get\_entry\_offset, 120  
    fiasco\_gdt\_set, 120  
    fiasco\_ldt\_set, 119  
    fiasco\_tbuf\_clear, 118  
    fiasco\_tbuf\_dump, 118  
    fiasco\_tbuf\_get\_status, 116  
    fiasco\_tbuf\_get\_status\_phys, 116  
    fiasco\_tbuf\_log, 116  
    fiasco\_tbuf\_log\_3val, 117  
    fiasco\_tbuf\_log\_binary, 118  
    fiasco\_watchdog\_takeover, 119  
    fiasco\_watchdog\_touch, 119

api\_calls\_rt\_sched  
    l4\_next\_period\_id, 128  
    l4\_preemption\_id, 128  
    l4\_rt\_begin\_minimal\_periodic, 124  
    l4\_rt\_begin\_strictly\_periodic, 123

l4\_rt\_end\_periodic, 125  
 l4\_rt\_generic, 129  
 l4\_rt\_next\_period, 127  
 l4\_rt\_next\_reservation, 127  
 l4\_rt\_remove, 125  
 l4\_rt\_set\_period, 126  
 slice, 122, 123  
**api\_gfxbitmap\_bitmap**  
 gfxbitmap\_bmap, 350  
 gfxbitmap\_color\_pix\_t, 349  
 gfxbitmap\_color\_t, 349  
 gfxbitmap\_convert\_color, 349  
 gfxbitmap\_copy, 351  
 gfxbitmap\_fill, 349  
 gfxbitmap\_set, 350  
**api\_gfxbitmap\_font**  
 gfxbitmap\_font\_data, 353  
 gfxbitmap\_font\_get, 353  
 gfxbitmap\_font\_height, 353  
 gfxbitmap\_font\_init, 353  
 gfxbitmap\_font\_text, 354  
 gfxbitmap\_font\_text\_scale, 354  
 gfxbitmap\_font\_width, 353  
**api\_l4re\_c\_debug**  
 l4re\_debug\_obj\_debug, 53  
**api\_l4re\_c\_ds**  
 l4re\_ds\_allocate, 51  
 l4re\_ds\_clear, 51  
 l4re\_ds\_copy\_in, 51  
 l4re\_ds\_flags, 51  
 l4re\_ds\_info, 51  
 l4re\_ds\_phys, 52  
 l4re\_ds\_size, 51  
**api\_l4re\_c\_event**  
 l4re\_event\_get, 53  
**api\_l4re\_c\_log**  
 l4re\_log\_print, 54  
 l4re\_log\_print\_srv, 56  
 l4re\_log\_printf, 55  
 l4re\_log\_printf\_srv, 56  
**api\_l4re\_c\_mem\_alloc**  
 l4re\_ma\_alloc, 58  
 l4re\_ma\_alloc\_srv, 60  
 l4re\_ma\_flags, 58  
 l4re\_ma\_free, 59  
 l4re\_ma\_free\_srv, 60  
**api\_l4re\_c\_ns**  
 l4re\_ns\_query\_to\_srv, 62  
 l4re\_ns\_register\_flags, 62  
 l4re\_ns\_register\_obj\_srv, 62  
**api\_l4re\_c\_rm**  
 l4re\_rm\_attach, 65  
 l4re\_rm\_attach\_srv, 71  
 l4re\_rm\_detach, 67  
 l4re\_rm\_detach\_ds, 67  
 l4re\_rm\_detach\_ds\_unmap, 69  
 l4re\_rm\_detach\_srv, 72  
 l4re\_rm\_detach\_unmap, 68  
 l4re\_rm\_find, 69  
 l4re\_rm\_find\_srv, 72  
 l4re\_rm\_flags\_t, 64  
 l4re\_rm\_free\_area, 65  
 l4re\_rm\_free\_area\_srv, 71  
 l4re\_rm\_reserve\_area, 64  
 l4re\_rm\_reserve\_area\_srv, 70  
 l4re\_rm\_show\_lists, 70  
**api\_l4re\_c\_util**  
 l4re\_util\_cap\_release, 103  
**api\_l4re\_c\_util\_cap**  
 l4re\_util\_cap\_last, 74  
**api\_l4re\_c\_util\_kumem\_alloc**  
 l4re\_util\_kumem\_alloc, 74  
**api\_l4re\_c\_video**  
 l4re\_video\_goops\_create\_buffer, 78  
 l4re\_video\_goops\_create\_view, 79  
 l4re\_video\_goops\_delete\_buffer, 79  
 l4re\_video\_goops\_delete\_view, 79  
 l4re\_video\_goops\_get\_static\_buffer, 79  
 l4re\_video\_goops\_get\_view, 80  
 l4re\_video\_goops\_info, 78  
 l4re\_video\_goops\_info\_flags\_t, 77  
 l4re\_video\_goops\_refresh, 78  
 l4re\_video\_view\_get\_info, 80  
 l4re\_video\_view\_info\_flags\_t, 77  
 l4re\_video\_view\_refresh, 80  
 l4re\_video\_view\_set\_info, 80  
 l4re\_video\_view\_set\_viewport, 81  
 l4re\_video\_view\_stack, 81  
 l4re\_video\_view\_t, 77  
**api\_l4re\_elf\_aux**  
 L4RE\_ELF\_AUX\_ELEM, 85  
 L4RE\_ELF\_AUX\_ELEM\_T, 85  
**api\_l4re\_env**  
 l4re\_env, 88  
 l4re\_env\_t, 88  
 l4re\_get\_env\_cap, 89  
 l4re\_get\_env\_cap\_e, 89  
 l4re\_get\_env\_cap\_l, 90  
 l4re\_kip, 88  
**api\_l4re\_protocols**  
 Protocols, 96  
**api\_l4shm**  
 l4shmc\_area\_size, 379  
 l4shmc\_attach, 378  
 l4shmc\_attach\_to, 378  
 l4shmc\_connect\_chunk\_signal, 379  
 l4shmc\_create, 378  
**api\_l4shmc\_chunk**

l4shmc\_add\_chunk, 381  
l4shmc\_chunk\_capacity, 382  
l4shmc\_chunk\_ptr, 382  
l4shmc\_chunk\_signal, 382  
l4shmc\_get\_chunk, 381  
l4shmc\_get\_chunk\_to, 381  
api\_l4shmc\_chunk\_cons  
    l4shmc\_chunk\_consumed, 386  
    l4shmc\_chunk\_size, 387  
    l4shmc\_enable\_chunk, 385  
    l4shmc\_is\_chunk\_ready, 387  
    l4shmc\_wait\_chunk, 386  
    l4shmc\_wait\_chunk\_to, 386  
    l4shmc\_wait\_chunk\_try, 386  
api\_l4shmc\_chunk\_prod  
    l4shmc\_chunk\_ready, 383  
    l4shmc\_chunk\_ready\_sig, 384  
    l4shmc\_chunk\_try\_to\_take, 383  
    l4shmc\_is\_chunk\_clear, 384  
api\_l4shmc\_signal  
    l4shmc\_add\_signal, 389  
    l4shmc\_attach\_signal, 389  
    l4shmc\_attach\_signal\_to, 389  
    l4shmc\_check\_magic, 390  
    l4shmc\_get\_signal\_to, 390  
    l4shmc\_signal\_cap, 390  
api\_l4shmc\_signal\_cons  
    l4shmc\_enable\_signal, 392  
    l4shmc\_wait\_any, 392  
    l4shmc\_wait\_any\_to, 393  
    l4shmc\_wait\_any\_try, 392  
    l4shmc\_wait\_signal, 393  
    l4shmc\_wait\_signal\_to, 393  
    l4shmc\_wait\_signal\_try, 394  
api\_l4shmc\_signal\_prod  
    l4shmc\_trigger, 391  
api\_libvcpu  
    l4vcpu\_halt, 374  
    l4vcpu\_irq\_disable, 371  
    l4vcpu\_irq\_disable\_save, 372  
    l4vcpu\_irq\_enable, 373  
    l4vcpu\_irq\_restore, 374  
    l4vcpu\_irq\_state\_t, 371  
    l4vcpu\_is\_irq\_entry, 375  
    l4vcpu\_is\_page\_fault\_entry, 375  
    l4vcpu\_print\_state, 375  
    l4vcpu\_state, 371  
api\_libvcpu\_ext  
    l4vcpu\_ext\_alloc, 376  
ARM Virtual Registers (UTCB), 280  
asm\_enter\_kdebug  
    l4\_debugger\_api, 161, 163  
Atomic Instructions, 294  
attach  
    L4::Irq, 654  
    L4Re::Rm, 819, 820  
    L4Re::Util::Event\_buffer\_t, 584  
    Attach\_flags  
        L4Re::Rm, 817  
    Attach\_flags  
        L4Re::Rm, 816  
    Attr  
        L4::Thread::Attr, 410  
    Auto\_ptr  
        cxx::Auto\_ptr, 417  
    auto\_refresh  
        L4Re::Video::Goos::Info, 634  
    Auxiliary data, 92  
    avail  
        cxx::List\_alloc, 751  
    Avl\_map  
        cxx::Avl\_map, 423  
    Avl\_set  
        cxx::Avl\_set, 430, 431  
    ax  
        l4\_vcpu\_regs\_t, 713  
    b  
        L4Re::Video::Pixel\_info, 804, 805  
    Base API, 103  
    Basic Macros, 110  
    Be\_file\_system  
        L4Re::Vfs::Be\_file\_system, 461  
    begin  
        cxx::Avl\_set, 433, 434  
        cxx::Bits::Bst, 477, 478  
    bind  
        L4::Icu, 625  
        L4::Thread::Attr, 412  
    bind\_thread  
        L4::Ipc\_gate, 651  
    bit  
        cxx::Bitmap\_base, 466  
    Bit Manipulation, 301  
    Bitmap  
        cxx::Bitmap, 464  
    Bitmap graphics and fonts, 347  
    bits\_per\_pixel  
        L4Re::Video::Pixel\_info, 804  
    bp  
        l4\_vcpu\_regs\_t, 713  
    buf  
        L4::Ipc::Buf\_cp\_out, 487  
        L4Re::Util::Event\_buffer\_t, 584  
    Buf\_cp\_in  
        L4::Ipc::Buf\_cp\_in, 485  
    Buf\_cp\_out  
        L4::Ipc::Buf\_cp\_out, 486

Buf\_in  
     L4::Ipc::Buf\_in, 488  
 buffer  
     L4Re::Util::Event\_t, 587  
 Buffer Registers (BRs), 265  
 bx  
     l4\_vcpu\_regs\_t, 713  
 bytes\_per\_pixel  
     L4Re::Video::Pixel\_info, 804, 806  
  
 C++ Exceptions, 43  
 Cache Consistency, 152  
 call  
     L4::Ipc::Iostream, 646  
 Cap  
     L4::Cap, 490, 491  
 cap  
     L4::Cap\_base, 499  
     L4::Invalid\_capability, 641  
     L4::Kobject, 676  
 cap\_alloc  
     l4re\_cap\_api, 98  
 Cap\_base  
     L4::Cap\_base, 498  
 cap\_cast  
     l4\_cap\_api, 258  
 cap\_dynamic\_cast  
     l4\_cap\_api, 259  
 cap\_equal  
     L4::Task, 867  
 cap\_has\_child  
     L4::Task, 866  
 cap\_reinterpret\_cast  
     l4\_cap\_api, 258  
 Cap\_type  
     L4::Cap\_base, 498  
 cap\_valid  
     L4::Task, 866  
 Capabilities, 255  
 Capability allocator, 73  
 cast  
     L4vcpu::Vcpu, 898  
 chain  
     L4::Irq, 655  
 chars  
     cxx::Bitmap\_base, 466  
 Chunks, 380  
 clear  
     L4Re::Dataspace, 519  
     L4Re::Util::Dataspace\_svr, 527  
     L4vcpu::State, 855  
 clear\_all  
     cxx::Bitmap, 464  
 clear\_bit

    cxx::Bitmap\_base, 467  
 cnt\_iobmap\_tlb\_flush  
     l4\_tracebuffer\_status\_t, 707  
 Color\_component  
     L4Re::Video::Color\_component, 505  
 Com\_error  
     L4::Com\_error, 510  
 Comfortable Command Line Parsing, 334  
 Console API, 81  
 Consumer, 385, 391  
 Continuous  
     L4Re::Mem\_alloc, 763  
 control  
     L4::Thread, 874  
 copy  
     L4Re::Util::Dataspace\_svr, 527  
 copy\_in  
     L4Re::Dataspace, 520  
 count  
     L4::Kip::Mem\_desc, 766  
     l4\_vhw\_descriptor, 719  
 CPU related functions, 283  
 create  
     L4::Factory, 592  
 create\_buffer  
     L4Re::Video::Goos, 616  
 create\_factory  
     L4::Factory, 595  
 create\_gate  
     L4::Factory, 595  
 create\_irq  
     L4::Factory, 597  
 create\_semaphore  
     L4::Factory, 596  
 create\_task  
     L4::Factory, 593  
 create\_thread  
     L4::Factory, 594  
 create\_view  
     L4Re::Video::Goos, 617  
 create\_vm  
     L4::Factory, 598  
 cx  
     l4\_vcpu\_regs\_t, 713  
 cxx, 399  
     cxx::Auto\_ptr, 416  
         ~Auto\_ptr, 417  
         Auto\_ptr, 417  
         get, 418  
         operator Priv\_type \*, 419  
         operator\*, 418  
         operator->, 418  
         operator=, 418  
         Ref\_type, 417

release, 418  
cxx::Avl\_map, 419  
    Avl\_map, 423  
    erase, 425  
    find, 424  
    find\_node, 424  
    insert, 423  
    lower\_bound\_node, 424  
    remove, 424  
cxx::Avl\_set, 426  
    Avl\_set, 430, 431  
    begin, 433, 434  
    end, 433, 434  
    find\_node, 432  
    insert, 431  
    lower\_bound\_node, 432  
    rbegin, 434, 435  
    remove, 432  
    rend, 434, 435  
cxx::Avl\_set::Node, 783  
    valid, 784  
cxx::Avl\_tree, 435  
    insert, 439  
    Iterator, 439  
    remove, 440  
cxx::Avl\_tree\_node, 441  
cxx::Base\_slab  
    max\_free\_slabs, 448  
    object\_size, 448  
    objects\_per\_slab, 448  
    slab\_size, 448  
cxx::Base\_slab\_static  
    max\_free\_slabs, 452  
    object\_size, 452  
    objects\_per\_slab, 452  
    slab\_size, 452  
cxx::Base\_slab, 446  
    free\_objects, 449  
    total\_objects, 449  
cxx::Base\_slab\_static, 449  
    alloc, 452  
    free, 453  
    free\_objects, 453  
    total\_objects, 453  
cxx::Bitmap, 461  
    Bitmap, 464  
    clear\_all, 464  
cxx::Bitmap\_base, 464  
    bit, 466  
    chars, 466  
    clear\_bit, 467  
    scan\_zero, 468  
    set\_bit, 467  
    words, 466  
cxx::Bitmap\_base::Char, 504  
cxx::Bitmap\_base::Word, 907  
cxx::Bits, 401  
cxx::Bits::Bst, 472  
    begin, 477, 478  
    dir, 476  
    end, 477, 478  
    find, 481  
    find\_node, 479  
    lower\_bound\_node, 480  
    rbegin, 478, 479  
    rend, 479  
cxx::Bits::Bst\_node, 482  
cxx::Bits::Direction, 537  
    Direction\_e, 538  
    L, 538  
    N, 538  
    R, 538  
cxx::List, 747  
    items, 750  
    push\_back, 748  
    push\_front, 748  
    remove, 749  
    size, 749  
cxx::List::Iter, 670  
cxx::List\_alloc, 750  
    alloc, 751  
    avail, 751  
    free, 751  
    List\_alloc, 750  
cxx::List\_item, 752  
    get\_next\_item, 753  
    get\_prev\_item, 753  
    insert\_next\_item, 753  
    insert\_prev\_item, 753  
    push\_back, 754  
    push\_front, 754  
    remove, 755  
    remove\_me, 754  
cxx::List\_item::Iter, 672  
    remove\_me, 675  
cxx::List\_item::T\_iter, 857  
    remove\_me, 860  
cxx::Lt\_functor, 760  
cxx::New\_allocator, 783  
cxx::Nothrow, 784  
cxx::Pair, 795  
    Pair, 796  
cxx::Pair\_first\_compare, 796  
    operator(), 797  
    Pair\_first\_compare, 797  
cxx::Slab, 839  
    alloc, 842  
    free, 842

cxx::Slab\_static, 843  
     alloc, 845  
 cxx\_api  
     max, 46  
     min, 46  
     operator new, 46  
  
 d\_val  
     Elf32\_Dyn, 550  
     Elf64\_Dyn, 555  
 Data-Space API, 82  
 data\_space  
     L4Re::Vfs::Regular\_file, 810  
 Dataspace  
     api\_l4re\_protocols, 96  
 Dataspace interface, 50  
 Debug  
     api\_l4re\_protocols, 96  
 debug  
     L4Re::Debug\_obj, 530  
 Debug interface, 52  
 Debugging API, 83  
 dec\_refcnt  
     L4::Kobject, 678  
 Default  
     api\_l4re\_protocols, 96  
 delete\_buffer  
     L4Re::Video::Goos, 616  
 delete\_view  
     L4Re::Video::Goos, 617  
 descs  
     l4\_vhw\_descriptor, 719  
 detach  
     L4::Irq, 656  
     L4Re::Rm, 820, 821  
     L4Re::Util::Event\_buffer\_t, 585  
 Detach\_again  
     L4Re::Rm, 816  
 Detach\_exact  
     L4Re::Rm, 817  
 Detach\_free  
     L4Re::Rm, 816  
 Detach\_overlap  
     L4Re::Rm, 817  
 Detach\_flags  
     L4Re::Rm, 817  
 Detach\_result  
     L4Re::Rm, 816  
 Detached\_ds  
     L4Re::Rm, 816  
 DF\_1\_CONFALT  
     l4util\_elf, 331  
 DF\_1\_DIRECT  
     l4util\_elf, 331  
     DF\_1\_DISPRLDNE  
         l4util\_elf, 332  
     DF\_1\_DISPRELPND  
         l4util\_elf, 332  
     DF\_1\_ENDFILTEE  
         l4util\_elf, 332  
     DF\_1\_GLOBAL  
         l4util\_elf, 330  
     DF\_1\_GROUP  
         l4util\_elf, 330  
     DF\_1\_INTERPOSE  
         l4util\_elf, 331  
     DF\_1\_LOADFLTR  
         l4util\_elf, 331  
     DF\_1\_NODEFLIB  
         l4util\_elf, 331  
     DF\_1\_NODEDELETE  
         l4util\_elf, 331  
     DF\_1\_NODUMP  
         l4util\_elf, 331  
     DF\_1\_NOOPEN  
         l4util\_elf, 331  
     DF\_1\_NOW  
         l4util\_elf, 330  
     DF\_1\_ORIGIN  
         l4util\_elf, 331  
     DF\_P1\_GROUPEPERM  
         l4util\_elf, 332  
     DF\_P1\_LAZYLOAD  
         l4util\_elf, 332  
     di  
         l4\_vcpu\_regs\_t, 713  
     dir  
         cxx::Bits::Bst, 476  
     Direction\_e  
         cxx::Bits::Direction, 538  
     dispatch  
         L4::Basic\_registry, 454  
         L4::Server\_object, 839  
         L4Re::Util::Vcon\_svr, 889  
         L4Re::Util::Video::Goos\_svr, 621  
     drive\_cylinders  
         l4util\_mb\_drive\_t, 742  
     drive\_mode  
         l4util\_mb\_drive\_t, 742  
     drive\_number  
         l4util\_mb\_drive\_t, 742  
     DT\_HIPROC  
         l4util\_elf, 330  
     DT\_LOPROC  
         l4util\_elf, 330  
     DT\_NULL  
         l4util\_elf, 330  
     dump

L4Re::Video::Color\_component, 506  
L4Re::Video::Pixel\_info, 806  
dx  
    l4\_vcpu\_regs\_t, 713  
  
e\_phnum  
    Elf32\_Ehdr, 552  
    Elf64\_Ehdr, 556  
e\_shnum  
    Elf32\_Ehdr, 552  
    Elf64\_Ehdr, 556  
Eager\_map  
    L4Re::Rm, 817  
EI\_CLASS  
    l4util\_elf, 324  
EI\_DATA  
    l4util\_elf, 324  
EI\_OSABI  
    l4util\_elf, 326  
EI\_PAD  
    l4util\_elf, 328  
EI\_VERSION  
    l4util\_elf, 325, 326  
ELF binary format, 307  
Elf32\_Dyn, 550  
    d\_val, 550  
Elf32\_Ehdr, 550  
    e\_phnum, 552  
    e\_shnum, 552  
Elf32\_Phdr, 552  
Elf32\_Shdr, 553  
Elf32\_Sym, 554  
Elf64\_Dyn, 554  
    d\_val, 555  
Elf64\_Ehdr, 555  
    e\_phnum, 556  
    e\_shnum, 556  
Elf64\_Phdr, 557  
Elf64\_Shdr, 557  
Elf64\_Sym, 558  
ELFCLASSNONE  
    l4util\_elf, 324  
ELFDATA2LSB  
    l4util\_elf, 325  
ELFDATA2MSB  
    l4util\_elf, 325  
ELFDATANONE  
    l4util\_elf, 325  
ELFOSABI\_AIX  
    l4util\_elf, 327  
ELFOSABI\_FREEBSD  
    l4util\_elf, 327  
ELFOSABI\_HPUX  
    l4util\_elf, 326, 327  
  
ELFOSABI\_IRIX  
    l4util\_elf, 327  
ELFOSABI\_LINUX  
    l4util\_elf, 327  
ELFOSABI\_MODESTO  
    l4util\_elf, 327  
ELFOSABI\_NETBSD  
    l4util\_elf, 327  
ELFOSABI\_OPENBSD  
    l4util\_elf, 328  
ELFOSABI\_SOLARIS  
    l4util\_elf, 327  
ELFOSABI\_SYSV  
    l4util\_elf, 326  
ELFOSABI\_TRU64  
    l4util\_elf, 327  
EM\_ARC  
    l4util\_elf, 328  
end  
    cxx::Avl\_set, 433, 434  
    cxx::Bits::Bst, 477, 478  
    L4::Kip::Mem\_desc, 767  
enter\_kdebug  
    l4\_debugger\_api, 161, 162  
entry\_ip  
    L4vcpu::Vcpu, 897  
entry\_sp  
    L4vcpu::Vcpu, 897  
env  
    L4Re::Env, 563  
erase  
    cxx::Avl\_map, 425  
Error codes, 167  
Error Handling, 193  
Event  
    api\_l4re\_protocols, 96  
Event API, 91  
Event interface, 53  
Event\_buffer\_t  
    L4Re::Event\_buffer\_t, 580  
ex\_regs  
    L4::Thread, 872, 873  
exc\_handler  
    L4::Thread::Attr, 411, 412  
Exception registers, 267  
ext\_alloc  
    L4vcpu::Vcpu, 898  
Extended vCPU support, 376  
  
F\_above  
    L4Re::Video::View, 902  
F\_auto\_refresh  
    L4Re::Video::Goos, 615  
F\_dyn\_allocated

L4Re::Video::View, 902  
 F\_dynamic\_buffers  
     L4Re::Video::Goos, 616  
 F\_dynamic\_views  
     L4Re::Video::Goos, 615  
 F\_flags\_mask  
     L4Re::Video::View, 902  
 F\_fully\_dynamic  
     L4Re::Video::View, 902  
 F\_l4re\_video\_goos\_auto\_refresh  
     api\_l4re\_c\_video, 77  
 F\_l4re\_video\_goos\_dynamic\_buffers  
     api\_l4re\_c\_video, 77  
 F\_l4re\_video\_goos\_dynamic\_views  
     api\_l4re\_c\_video, 77  
 F\_l4re\_video\_goos\_pointer  
     api\_l4re\_c\_video, 77  
 F\_l4re\_video\_view\_above  
     api\_l4re\_c\_video, 78  
 F\_l4re\_video\_view\_dyn\_allocated  
     api\_l4re\_c\_video, 78  
 F\_l4re\_video\_view\_flags\_mask  
     api\_l4re\_c\_video, 78  
 F\_l4re\_video\_view\_none  
     api\_l4re\_c\_video, 78  
 F\_l4re\_video\_view\_set\_background  
     api\_l4re\_c\_video, 78  
 F\_l4re\_video\_view\_set\_buffer  
     api\_l4re\_c\_video, 78  
 F\_l4re\_video\_view\_set\_buffer\_offset  
     api\_l4re\_c\_video, 78  
 F\_l4re\_video\_view\_set\_bytes\_per\_line  
     api\_l4re\_c\_video, 78  
 F\_l4re\_video\_view\_set\_flags  
     api\_l4re\_c\_video, 78  
 F\_l4re\_video\_view\_set\_pixel  
     api\_l4re\_c\_video, 78  
 F\_l4re\_video\_view\_set\_position  
     api\_l4re\_c\_video, 78  
 F\_none  
     L4Re::Video::View, 902  
 F\_pointer  
     L4Re::Video::Goos, 615  
 F\_set\_background  
     L4Re::Video::View, 902  
 F\_set\_buffer  
     L4Re::Video::View, 902  
 F\_set\_buffer\_offset  
     L4Re::Video::View, 902  
 F\_set\_bytes\_per\_line  
     L4Re::Video::View, 902  
 F\_set\_flags  
     L4Re::Video::View, 902  
 F\_set\_pixel

L4Re::Video::View, 902  
 F\_set\_position  
     L4Re::Video::View, 902  
 faccessat  
     L4Re::Vfs::Directory, 541  
 Factory, 169  
 factory  
     L4Re::Env, 564, 567  
 fchmod  
     L4Re::Vfs::Generic\_file, 611  
 fd  
     l4\_vhw\_entry, 721  
 fdatasync  
     L4Re::Vfs::Regular\_file, 811  
 features  
     l4\_icu\_info\_t, 688  
 Fiasco extensions, 113  
 Fiasco real time scheduling extensions, 121  
 Fiasco-UX Virtual devices, 277  
 fiasco\_gdt\_get\_entry\_offset  
     api\_calls\_fiasco, 120  
 fiasco\_gdt\_set  
     api\_calls\_fiasco, 120  
 fiasco\_ldt\_set  
     api\_calls\_fiasco, 119  
 fiasco\_tbuf\_clear  
     api\_calls\_fiasco, 118  
 fiasco\_tbuf\_dump  
     api\_calls\_fiasco, 118  
 fiasco\_tbuf\_get\_status  
     api\_calls\_fiasco, 116  
 fiasco\_tbuf\_get\_status\_phys  
     api\_calls\_fiasco, 116  
 fiasco\_tbuf\_log  
     api\_calls\_fiasco, 116  
 fiasco\_tbuf\_log\_3val  
     api\_calls\_fiasco, 117  
 fiasco\_tbuf\_log\_binary  
     api\_calls\_fiasco, 118  
 fiasco\_watchdog\_takeover  
     api\_calls\_fiasco, 119  
 fiasco\_watchdog\_touch  
     api\_calls\_fiasco, 119  
 find  
     cxx::Avl\_map, 424  
     cxx::Bits::Bst, 481  
     L4Re::Rm, 822  
 find\_node  
     cxx::Avl\_map, 424  
     cxx::Avl\_set, 432  
     cxx::Bits::Bst, 479  
 first  
     L4::Kip::Mem\_desc, 766  
 first\_free\_cap

L4Re::Env, 564, 568  
first\_free\_utcb  
    L4Re::Env, 565, 568  
Flags  
    L4Re::Video::Goos, 615  
    L4Re::Video::View, 902  
flags  
    l4\_exc\_regs\_t, 686  
    l4\_msgtag\_t, 694  
    L4Re::Dataspace, 522  
    L4Re::Video::View::Info, 638  
    l4re\_env\_cap\_entry\_t, 729  
Flex pages, 130  
foreach\_available\_event  
    L4Re::Util::Event\_buffer\_consumer\_t, 576  
fpage  
    L4::Cap\_base, 501  
free  
    cxx::Base\_slab\_static, 453  
    cxx::List\_alloc, 751  
    cxx::Slab, 842  
    L4Re::Cap\_alloc, 493  
    L4Re::Mem\_alloc, 764  
free\_area  
    L4Re::Rm, 819  
free\_fd  
    L4Re::Vfs::Fs, 607  
free\_objects  
    cxx::Base\_slab, 449  
    cxx::Base\_slab\_static, 453  
fstat64  
    L4Re::Vfs::Generic\_file, 610  
fsync  
    L4Re::Vfs::Regular\_file, 811  
ftruncate64  
    L4Re::Vfs::Regular\_file, 811  
Functions for rendering bitmap data in frame buffers, 348  
Functions for rendering bitmap fonts to frame buffers, 351  
Functions to manipulate the local IDT, 286

g  
    L4Re::Video::Pixel\_info, 804, 805  
get  
    cxx::Auto\_ptr, 418  
    L4::Ipc::Istream, 663, 664  
    L4Re::Env, 565  
    L4Re::Video::Color\_component, 506  
get\_attr  
    L4::Vcon, 888  
get\_buffer  
    L4Re::Event, 573  
get\_cap  
    L4Re::Env, 566  
get\_cap\_alloc  
    L4Re::Cap\_alloc, 493  
get\_fb  
    L4Re::Util::Video::Goos\_svr, 620  
get\_file  
    L4Re::Vfs::Fs, 606  
get\_infos  
    L4::Ipc\_gate, 651  
get\_lock  
    L4Re::Vfs::Regular\_file, 812  
get\_next\_item  
    cxx::List\_item, 753  
get\_object\_name  
    L4::Debugger, 534  
get\_prev\_item  
    cxx::List\_item, 753  
get\_static\_buffer  
    L4Re::Video::Goos, 616  
get\_status\_flags  
    L4Re::Vfs::Generic\_file, 611  
gfxbitmap\_bmap  
    api\_gfxbitmap\_bitmap, 350  
gfxbitmap\_color\_pix\_t  
    api\_gfxbitmap\_bitmap, 349  
gfxbitmap\_color\_t  
    api\_gfxbitmap\_bitmap, 349  
gfxbitmap\_convert\_color  
    api\_gfxbitmap\_bitmap, 349  
gfxbitmap\_copy  
    api\_gfxbitmap\_bitmap, 351  
gfxbitmap\_fill  
    api\_gfxbitmap\_bitmap, 349  
gfxbitmap\_font\_data  
    api\_gfxbitmap\_font, 353  
gfxbitmap\_font\_get  
    api\_gfxbitmap\_font, 353  
gfxbitmap\_font\_height  
    api\_gfxbitmap\_font, 353  
gfxbitmap\_font\_init  
    api\_gfxbitmap\_font, 353  
gfxbitmap\_font\_text  
    api\_gfxbitmap\_font, 354  
gfxbitmap\_font\_text\_scale  
    api\_gfxbitmap\_font, 354  
gfxbitmap\_font\_width  
    api\_gfxbitmap\_font, 353  
gfxbitmap\_offset, 612  
gfxbitmap\_set  
    api\_gfxbitmap\_bitmap, 350  
global\_id  
    L4::Debugger, 533  
Goos  
    api\_l4re\_protocols, 96

Goos video API, 100  
 halt  
     L4vcpu::Vcpu, 896  
 has\_alpha  
     L4Re::Video::Pixel\_info, 805  
 i  
     L4vcpu::Vcpu, 897  
 IA32 Port I/O API, 344  
 idle\_time  
     L4::Scheduler, 833  
 In\_area  
     L4Re::Rm, 817  
 info  
     L4::Icu, 626  
     L4::Scheduler, 832  
     L4Re::Dataspace, 523  
     L4Re::Video::Goos, 616  
     L4Re::Video::View, 902  
 init  
     L4Re::Util::Event\_t, 587  
 init\_infos  
     L4Re::Util::Video::Goos\_svr, 622  
 Initial Environment, 86  
 initial\_caps  
     L4Re::Env, 565, 569  
 insert  
     cxx::Avl\_map, 423  
     cxx::Avl\_set, 431  
     cxx::Avl\_tree, 439  
 insert\_next\_item  
     cxx::List\_item, 753  
 insert\_prev\_item  
     cxx::List\_item, 753  
 Integer Types, 394  
 interface  
     L4::Meta, 772  
 Interface for asynchronous ISR handlers with a given IRQ capability., 362  
 Interface for asynchronous ISR handlers., 359  
 Interface using direct functionality., 355, 360  
 Internal constants, 367  
 Internal functions, 300  
 internal\_loop  
     L4::Server, 836  
 Interrupt controller, 175  
 Invalid  
     L4::Cap\_base, 498  
 Invalid\_capability  
     L4::Invalid\_capability, 641  
 ioctl  
     L4Re::Vfs::Special\_file, 854  
 Iostream  
     L4::Ipc::Iostream, 645  
 IPC-Gate API, 107  
 irq  
     L4Re::Util::Event\_t, 587  
 IRQ handling library, 355  
 irq\_disable\_save  
     L4vcpu::Vcpu, 894  
 irq\_enable  
     L4vcpu::Vcpu, 895  
 irq\_no  
     l4\_vhw\_entry, 721  
 irq\_restore  
     L4vcpu::Vcpu, 895  
 IRQs, 198  
 is\_irq\_entry  
     L4vcpu::Vcpu, 896  
 is\_online  
     L4::Scheduler, 833  
 is\_page\_fault\_entry  
     L4vcpu::Vcpu, 896  
 is\_valid  
     L4::Cap\_base, 500  
 is\_virtual  
     L4::Kip::Mem\_desc, 768  
 Istream  
     L4::Ipc::Istream, 663  
 items  
     cxx::List, 750  
 Iterator  
     cxx::Avl\_tree, 439  
 kd\_display  
     l4\_debugger\_api, 162, 163  
 kdebug\_config  
     l4\_kernel\_info\_t, 691  
 kdebug\_permission  
     l4\_kernel\_info\_t, 692  
 Kept\_ds  
     L4Re::Rm, 816  
 Kernel Debugger, 159  
 Kernel Interface Page, 208  
 Kernel Interface Page API, 332  
 Kernel Objects, 204  
 ko  
     l4\_debugger\_api, 162, 163  
 kobj\_to\_id  
     L4::Debugger, 533  
 kobject\_typeid  
     L4, 404  
     L4::Kobject, 679  
     L4::Kobject\_2t, 681  
     L4::Kobject\_t, 683  
 Kumem allocator utility, 74  
 Kumem utilties, 99

kumem\_alloc  
l4re\_util\_kumem, 99

L  
    cxx::Bits::Direction, 538

L4, 401  
    kobject\_typeid, 404

L4::Alloc\_list, 409

L4::Base\_exception, 444

L4::Basic\_registry, 454  
    dispatch, 454

L4::Bounds\_error, 469

L4::Cap, 488  
    Cap, 490, 491  
    move, 491

L4::Cap\_base  
    Invalid, 498  
    No\_init, 498

L4::Cap\_base, 496  
    \_c, 503  
    cap, 499  
    Cap\_base, 498  
    Cap\_type, 498  
    fpage, 501  
    is\_valid, 500  
    No\_init\_type, 498  
    snd\_base, 502  
    validate, 503

L4::Com\_error, 507  
    Com\_error, 510

L4::Debugger, 530  
    get\_object\_name, 534  
    global\_id, 533  
    kobj\_to\_id, 533  
    query\_log\_name, 534  
    query\_log\_typeid, 534  
    set\_object\_name, 533  
    switch\_log, 534

L4::Element\_already\_exists, 544

L4::Element\_not\_found, 547

L4::Exception\_tracer, 588

L4::Factory, 589  
    create, 592  
    create\_factory, 595  
    create\_gate, 595  
    create\_irq, 597  
    create\_semaphore, 596  
    create\_task, 593  
    create\_thread, 594  
    create\_vm, 598

L4::Factory::Lstr, 759

L4::Factory::Nil, 783

L4::Factory::S, 825  
    operator l4\_mshtag\_t, 828

operator<<, 828  
S, 827

L4::Icu, 622  
    bind, 625  
    info, 626  
    mask, 628  
    msi\_info, 627  
    set\_mode, 629  
    unbind, 626  
    unmask, 628

L4::Icu::Info, 631

L4::Invalid\_capability, 638  
    cap, 641  
    Invalid\_capability, 641

L4::IOModifier, 641

L4::Ipc::Buf\_cp\_in, 485  
    Buf\_cp\_in, 485

L4::Ipc::Buf\_cp\_out, 486  
    buf, 487  
    Buf\_cp\_out, 486  
    size, 487

L4::Ipc::Buf\_in, 487  
    Buf\_in, 488

L4::Ipc::Iostream, 642  
    call, 646  
    Iostream, 645  
    reply\_and\_wait, 647, 648  
    reset, 646

L4::Ipc::Istream, 659  
    get, 663, 664  
    Istream, 663  
    receive, 667  
    reset, 663  
    skip, 664  
    tag, 665  
    wait, 665, 666

L4::Ipc::Msg\_ptr, 777  
    Msg\_ptr, 777

L4::Ipc::Ostream, 787  
    put, 791  
    send, 792  
    tag, 791

L4::Ipc::Small\_buf, 845

L4::Ipc\_svr  
    Reply\_compound, 406  
    Reply\_separate, 406

L4::Ipc\_gate, 649  
    bind\_thread, 651  
    get\_infos, 651

L4::Ipc\_svr, 405  
    Reply\_mode, 405

L4::Ipc\_svr::Compound\_reply, 510

L4::Ipc\_svr::Default\_loop\_hooks, 535

L4::Ipc\_svr::Default\_setup\_wait, 536

L4::Ipc\_svr::Default\_timeout, 536  
 L4::Ipc\_svr::Ignore\_errors, 630  
 L4::Irq, 651  
     attach, 654  
     chain, 655  
     detach, 656  
     receive, 656  
     trigger, 658  
     unmask, 658  
     wait, 657  
 L4::Kip::Mem\_desc, 764  
     count, 766  
     end, 767  
     first, 766  
     is\_virtual, 768  
     Mem\_desc, 765  
     set, 769  
     size, 767  
     start, 766  
     sub\_type, 768  
     type, 768  
 L4::Kobject, 675  
     cap, 676  
     dec\_refcnt, 678  
     kobject\_typeid, 679  
 L4::Kobject\_2t, 680  
     kobject\_typeid, 681  
 L4::Kobject\_t, 682  
     kobject\_typeid, 683  
 L4::Meta, 769  
     interface, 772  
     num\_interfaces, 772  
     supports, 773  
 L4::Out\_of\_memory, 792  
 L4::Runtime\_error, 823  
 L4::Scheduler, 829  
     idle\_time, 833  
     info, 832  
     is\_online, 833  
     run\_thread, 832  
 L4::Server, 834  
     internal\_loop, 836  
     Server, 836  
 L4::Server\_object, 837  
     dispatch, 839  
 L4::Smart\_cap, 846  
     Smart\_cap, 849  
 L4::String, 856  
 L4::Task, 860  
     add\_ku\_mem, 868  
     cap\_equal, 867  
     cap\_has\_child, 866  
     cap\_valid, 866  
     map, 863  
     unmap, 864  
     unmap\_batch, 865  
 L4::Thread, 869  
     control, 874  
     ex\_regs, 872, 873  
     modify\_senders, 878  
     register\_del\_irq, 877  
     stats\_time, 875  
     switch\_to, 874  
     vcpu\_control, 876  
     vcpu\_control\_ext, 877  
     vcpu\_resume\_commit, 876  
     vcpu\_resume\_start, 875  
 L4::Thread::Attr, 409  
     Attr, 410  
     bind, 412  
     exc\_handler, 411, 412  
     pager, 410, 411  
     ux\_host\_syscall, 413  
 L4::Thread::Modify\_senders, 776  
     add, 776  
 L4::Type\_info, 879  
 L4::Unknown\_error, 879  
 L4::Vcon, 882  
     get\_attr, 888  
     read, 886  
     send, 885  
     set\_attr, 887  
     write, 886  
 L4::Vm, 904  
     run, 907  
 L4\_BASE\_FACTORY\_CAP  
     l4\_cap\_api, 257  
 L4\_BASE\_ICU\_CAP  
     l4\_cap\_api, 257  
 L4\_BASE\_LOG\_CAP  
     l4\_cap\_api, 257  
 L4\_BASE\_PAGER\_CAP  
     l4\_cap\_api, 257  
 L4\_BASE\_SCHEDULER\_CAP  
     l4\_cap\_api, 257  
 L4\_BASE\_TASK\_CAP  
     l4\_cap\_api, 257  
 L4\_BASE\_THREAD\_CAP  
     l4\_cap\_api, 257  
 L4\_BDR\_IO\_SHIFT  
     l4\_utcb\_br\_api, 266  
 L4\_BDR\_MEM\_SHIFT  
     l4\_utcb\_br\_api, 266  
 L4\_BDR\_OBJ\_SHIFT  
     l4\_utcb\_br\_api, 266  
 l4\_cap\_api  
     L4\_BASE\_FACTORY\_CAP, 257  
     L4\_BASE\_ICU\_CAP, 257

L4\_BASE\_LOG\_CAP, 257  
L4\_BASE\_PAGER\_CAP, 257  
L4\_BASE\_SCHEDULER\_CAP, 257  
L4\_BASE\_TASK\_CAP, 257  
L4\_BASE\_THREAD\_CAP, 257  
L4\_CAP\_MASK, 257  
L4\_CAP\_SHIFT, 257  
L4\_CAP\_SIZE, 257  
L4\_INVALID\_CAP, 257  
L4\_CAP\_FPAGE\_R  
    l4\_fpage\_api, 133  
L4\_CAP\_FPAGE\_RO  
    l4\_fpage\_api, 133  
L4\_CAP\_FPAGE\_RW  
    l4\_fpage\_api, 133  
L4\_CAP\_MASK  
    l4\_cap\_api, 257  
L4\_CAP\_SHIFT  
    l4\_cap\_api, 257  
L4\_CAP\_SIZE  
    l4\_cap\_api, 257  
L4\_EACCESS  
    l4\_error\_api, 168  
L4\_EADDRNOTAVAIL  
    l4\_error\_api, 168  
L4\_EAGAIN  
    l4\_error\_api, 168  
L4\_EBADPROTO  
    l4\_error\_api, 168  
L4\_EBUSY  
    l4\_error\_api, 168  
L4\_EEXIST  
    l4\_error\_api, 168  
L4\_EINVAL  
    l4\_error\_api, 168  
L4\_EIO  
    l4\_error\_api, 168  
L4\_EIPC\_HI  
    l4\_error\_api, 168  
L4\_EIPC\_LO  
    l4\_error\_api, 168  
L4\_ENAMETOOLONG  
    l4\_error\_api, 168  
L4\_ENODEV  
    l4\_error\_api, 168  
L4\_ENOENT  
    l4\_error\_api, 168  
L4\_ENOMEM  
    l4\_error\_api, 168  
L4\_ENOREPLY  
    l4\_error\_api, 168  
L4\_ENOSYS  
    l4\_error\_api, 168  
L4\_EOK  
    l4\_error\_api, 168  
    l4\_error\_api, 168  
L4\_ERROR\_API  
    l4\_error\_api, 168  
L4\_ERANGE  
    l4\_error\_api, 168  
L4\_ERRNOMAX  
    l4\_error\_api, 168  
L4\_EPERM  
    l4\_error\_api, 168  
L4\_EBUSY, 168  
L4\_EEXIST, 168  
L4\_EINVAL, 168  
L4\_EIO, 168  
L4\_EIPC\_HI, 168  
L4\_EIPC\_LO, 168  
L4\_ENAMETOOLONG, 168  
L4\_ENODEV, 168  
L4\_ENOENT, 168  
L4\_ENOMEM, 168  
L4\_ENOREPLY, 168  
L4\_ENOSYS, 168  
L4\_EOK, 168  
L4\_EPERM, 168  
L4\_ERANGE, 168  
L4\_ERRNOMAX, 168  
L4\_FP\_ALL\_SPACES  
    l4\_task\_api, 221  
L4\_FP\_DELETE\_OBJ  
    l4\_task\_api, 221  
L4\_FP\_OTHER\_SPACES  
    l4\_task\_api, 221  
L4\_FPAGE\_ADDR\_BITS  
    l4\_fpage\_api, 133  
L4\_FPAGE\_ADDR\_SHIFT  
    l4\_fpage\_api, 133  
l4\_fpage\_api  
    L4\_CAP\_FPAGE\_R, 133  
    L4\_CAP\_FPAGE\_RO, 133  
    L4\_CAP\_FPAGE\_RW, 133  
    L4\_FPAGE\_ADDR\_BITS, 133  
    L4\_FPAGE\_ADDR\_SHIFT, 133  
    L4\_FPAGE\_BUFFERABLE, 134  
    L4\_FPAGE\_CACHE\_OPT, 134  
    L4\_FPAGE\_CACHEABLE, 134  
    L4\_FPAGE\_RIGHTS\_BITS, 133  
    L4\_FPAGE\_RIGHTS\_SHIFT, 133  
    L4\_FPAGE\_RO, 133  
    L4\_FPAGE\_RW, 133  
    L4\_FPAGE\_SIZE\_BITS, 133  
    L4\_FPAGE\_SIZE\_SHIFT, 133  
    L4\_FPAGE\_TYPE\_BITS, 133

L4\_FPAGE\_TYPE\_SHIFT, 133  
L4\_FPAGE\_UNCACHEABLE, 134  
L4\_IOPORT\_MAX, 134  
L4\_WHOLE\_ADDRESS\_SPACE, 133  
L4\_WHOLE\_IOADDRESS\_SPACE, 134  
L4\_FPAGE\_BUFFERABLE  
  l4\_fpage\_api, 134  
L4\_FPAGE\_CACHE\_OPT  
  l4\_fpage\_api, 134  
L4\_FPAGE\_CACHEABLE  
  l4\_fpage\_api, 134  
L4\_FPAGE\_RIGHTS\_BITS  
  l4\_fpage\_api, 133  
L4\_FPAGE\_RIGHTS\_SHIFT  
  l4\_fpage\_api, 133  
L4\_FPAGE\_RO  
  l4\_fpage\_api, 133  
L4\_FPAGE\_RW  
  l4\_fpage\_api, 133  
L4\_FPAGE\_SIZE\_BITS  
  l4\_fpage\_api, 133  
L4\_FPAGE\_SIZE\_SHIFT  
  l4\_fpage\_api, 133  
L4\_FPAGE\_TYPE\_BITS  
  l4\_fpage\_api, 133  
L4\_FPAGE\_TYPE\_SHIFT  
  l4\_fpage\_api, 133  
L4\_FPAGE\_UNCACHEABLE  
  l4\_fpage\_api, 134  
l4\_icu\_api  
  L4\_ICU\_FLAG\_MSI, 177  
L4\_ICU\_FLAG\_MSI  
  l4\_icu\_api, 177  
L4\_INVALID\_ADDR  
  l4\_memory\_api, 157  
L4\_INVALID\_CAP  
  l4\_cap\_api, 257  
L4\_IOPORT\_MAX  
  l4\_fpage\_api, 134  
l4\_ipc\_api  
  L4\_SYSF\_CALL, 184  
  L4\_SYSF\_NONE, 184  
  L4\_SYSF\_OPEN\_WAIT, 184  
  L4\_SYSF\_RECV, 184  
  L4\_SYSF\_REPLY, 184  
  L4\_SYSF\_REPLY\_AND\_WAIT, 185  
  L4\_SYSF\_SEND, 184  
  L4\_SYSF\_SEND\_AND\_WAIT, 185  
  L4\_SYSF\_WAIT, 185  
L4\_IPC\_ENOT\_EXISTENT  
  l4\_ipc\_err\_api, 194  
l4\_ipc\_err\_api  
  L4\_IPC\_ENOT\_EXISTENT, 194  
  L4\_IPC\_ERROR\_MASK, 194  
L4\_IPC\_REABORTED, 194  
L4\_IPC\_RECANCELED, 194  
L4\_IPC\_REMAPFAILED, 194  
L4\_IPC\_REMSGCUT, 194  
L4\_IPC\_RERCVPFTO, 194  
L4\_IPC\_RESNDPFTO, 194  
L4\_IPC RETIMEOUT, 194  
L4\_IPC\_SEABORTED, 194  
L4\_IPC\_SECANCELED, 194  
L4\_IPC\_SEMAPFAILED, 194  
L4\_IPC\_SEMSGCUT, 194  
L4\_IPC\_SERCVPFTO, 194  
L4\_IPC\_SESNDPFTO, 194  
L4\_IPC\_SETIMEOUT, 194  
L4\_IPC SND\_ERR\_MASK, 194  
L4\_IPC\_ERROR\_MASK  
  l4\_ipc\_err\_api, 194  
L4\_IPC\_GATE\_OP\_BIND  
  l4\_kernel\_object\_gate\_api, 108  
L4\_IPC\_GATE\_OP\_GET\_INFO  
  l4\_kernel\_object\_gate\_api, 108  
L4\_IPC\_REABORTED  
  l4\_ipc\_err\_api, 194  
L4\_IPC\_RECANCELED  
  l4\_ipc\_err\_api, 194  
L4\_IPC\_REMAPFAILED  
  l4\_ipc\_err\_api, 194  
L4\_IPC\_REMSGCUT  
  l4\_ipc\_err\_api, 194  
L4\_IPC\_RERCVPFTO  
  l4\_ipc\_err\_api, 194  
L4\_IPC\_RESNDPFTO  
  l4\_ipc\_err\_api, 194  
L4\_IPC RETIMEOUT  
  l4\_ipc\_err\_api, 194  
L4\_IPC\_SEABORTED  
  l4\_ipc\_err\_api, 194  
L4\_IPC\_SECANCELED  
  l4\_ipc\_err\_api, 194  
L4\_IPC\_SEMAPFAILED  
  l4\_ipc\_err\_api, 194  
L4\_IPC\_SEMSGCUT  
  l4\_ipc\_err\_api, 194  
L4\_IPC\_SERCVPFTO  
  l4\_ipc\_err\_api, 194  
L4\_IPC\_SESNDPFTO  
  l4\_ipc\_err\_api, 194  
L4\_IPC\_SETIMEOUT  
  l4\_ipc\_err\_api, 194  
L4\_IPC SND\_ERR\_MASK  
  l4\_ipc\_err\_api, 194  
l4\_irq\_api  
  L4\_IRQ\_F\_EDGE, 199  
  L4\_IRQ\_F\_LEVEL, 199

L4\_IRQ\_F\_LEVEL\_HIGH, 199  
L4\_IRQ\_F\_LEVEL\_LOW, 199  
L4\_IRQ\_F\_MASK, 200  
L4\_IRQ\_F\_NEG, 199  
L4\_IRQ\_F\_NEG\_EDGE, 200  
L4\_IRQ\_F\_NONE, 199  
L4\_IRQ\_F\_POS, 199  
L4\_IRQ\_F\_POS\_EDGE, 199  
L4\_IRQ\_F\_EDGE  
    l4\_irq\_api, 199  
L4\_IRQ\_F\_LEVEL  
    l4\_irq\_api, 199  
L4\_IRQ\_F\_LEVEL\_HIGH  
    l4\_irq\_api, 199  
L4\_IRQ\_F\_LEVEL\_LOW  
    l4\_irq\_api, 199  
L4\_IRQ\_F\_MASK  
    l4\_irq\_api, 200  
L4\_IRQ\_F\_NEG  
    l4\_irq\_api, 199  
L4\_IRQ\_F\_NEG\_EDGE  
    l4\_irq\_api, 200  
L4\_IRQ\_F\_NONE  
    l4\_irq\_api, 199  
L4\_IRQ\_F\_POS  
    l4\_irq\_api, 199  
L4\_IRQ\_F\_POS\_EDGE  
    l4\_irq\_api, 199  
L4\_ITEM\_CONT  
    l4\_msgetitem\_api, 141  
L4\_ITEM\_MAP  
    l4\_msgetitem\_api, 141  
l4\_kernel\_object\_gate\_api  
    L4\_IPC\_GATE\_OP\_BIND, 108  
    L4\_IPC\_GATE\_OP\_GET\_INFO, 108  
l4\_kip\_memdesc\_api  
    l4\_mem\_type\_archspecific, 213  
    l4\_mem\_type\_bootloader, 213  
    l4\_mem\_type\_conventional, 212  
    l4\_mem\_type\_dedicated, 212  
    l4\_mem\_type\_reserved, 212  
    l4\_mem\_type\_shared, 212  
    l4\_mem\_type\_undefined, 212  
l4\_kip\_vhw\_api  
    L4\_TYPE\_VHW\_FRAMEBUFFER, 278  
    L4\_TYPE\_VHW\_INPUT, 278  
    L4\_TYPE\_VHW\_NET, 278  
    L4\_TYPE\_VHW\_NONE, 278  
L4\_MAP\_ITEM\_GRANT  
    l4\_msgetitem\_api, 141  
L4\_MAP\_ITEM\_MAP  
    l4\_msgetitem\_api, 141  
l4\_mem\_op\_api  
    L4\_MEM\_WIDTH\_1BYTE, 279  
L4\_MEM\_WIDTH\_2BYTE, 279  
L4\_MEM\_WIDTH\_4BYTE, 279  
l4\_mem\_type\_archspecific  
    l4\_kip\_memdesc\_api, 213  
l4\_mem\_type\_bootloader  
    l4\_kip\_memdesc\_api, 213  
l4\_mem\_type\_conventional  
    l4\_kip\_memdesc\_api, 212  
l4\_mem\_type\_dedicated  
    l4\_kip\_memdesc\_api, 212  
l4\_mem\_type\_reserved  
    l4\_kip\_memdesc\_api, 212  
l4\_mem\_type\_shared  
    l4\_kip\_memdesc\_api, 212  
l4\_mem\_type\_undefined  
    l4\_kip\_memdesc\_api, 212  
L4\_MEM\_WIDTH\_1BYTE  
    l4\_mem\_op\_api, 279  
L4\_MEM\_WIDTH\_2BYTE  
    l4\_mem\_op\_api, 279  
L4\_MEM\_WIDTH\_4BYTE  
    l4\_mem\_op\_api, 279  
l4\_memory\_api  
    L4\_INVALID\_ADDR, 157  
l4\_msgetitem\_api  
    L4\_ITEM\_CONT, 141  
    L4\_ITEM\_MAP, 141  
    L4\_MAP\_ITEM\_GRANT, 141  
    L4\_MAP\_ITEM\_MAP, 141  
    L4\_RCV\_ITEM\_LOCAL\_ID, 141  
    L4\_RCV\_ITEM\_SINGLE\_CAP, 141  
l4\_mshtag\_api  
    L4\_MSGTAG\_ERROR, 248  
    L4\_MSGTAG\_FLAGS, 248  
    L4\_MSGTAG\_PROPAGATE, 248  
    L4\_MSGTAG\_SCHEDULE, 248  
    L4\_MSGTAG\_TRANSFER\_FPU, 248  
    L4\_MSGTAG\_XCPU, 248  
    L4\_PROTO\_ALLOW\_SYSCALL, 247  
    L4\_PROTO\_EXCEPTION, 247  
    L4\_PROTO\_FACTORY, 247  
    L4\_PROTO\_IO\_PAGE\_FAULT, 247  
    L4\_PROTO\_IRQ, 247  
    L4\_PROTO\_KOBJECT, 247  
    L4\_PROTO\_LOG, 247  
    L4\_PROTO\_NONE, 247  
    L4\_PROTO\_PAGE\_FAULT, 247  
    L4\_PROTO\_PF\_EXCEPTION, 247  
    L4\_PROTO\_PREEMPTION, 247  
    L4\_PROTO\_SCHEDULER, 247  
    L4\_PROTO\_SIGMA0, 247  
    L4\_PROTO\_SYS\_EXCEPTION, 247  
    L4\_PROTO\_TASK, 247  
    L4\_PROTO\_THREAD, 247

L4\_MSGTAG\_ERROR  
     l4\_mshtag\_api, 248

L4\_MSGTAG\_FLAGS  
     l4\_mshtag\_api, 248

L4\_MSGTAG\_PROPAGATE  
     l4\_mshtag\_api, 248

L4\_MSGTAG\_SCHEDULE  
     l4\_mshtag\_api, 248

L4\_MSGTAG\_TRANSFER\_FPU  
     l4\_mshtag\_api, 248

L4\_MSGTAG\_XCPU  
     l4\_mshtag\_api, 248

L4\_PROTO\_ALLOW\_SYSCALL  
     l4\_mshtag\_api, 247

L4\_PROTO\_EXCEPTION  
     l4\_mshtag\_api, 247

L4\_PROTO\_FACTORY  
     l4\_mshtag\_api, 247

L4\_PROTO\_IO\_PAGE\_FAULT  
     l4\_mshtag\_api, 247

L4\_PROTO\_IRQ  
     l4\_mshtag\_api, 247

L4\_PROTO\_KOBJECT  
     l4\_mshtag\_api, 247

L4\_PROTO\_LOG  
     l4\_mshtag\_api, 247

L4\_PROTO\_NONE  
     l4\_mshtag\_api, 247

L4\_PROTO\_PAGE\_FAULT  
     l4\_mshtag\_api, 247

L4\_PROTO\_PF\_EXCEPTION  
     l4\_mshtag\_api, 247

L4\_PROTO\_PREEMPTION  
     l4\_mshtag\_api, 247

L4\_PROTO\_SCHEDULER  
     l4\_mshtag\_api, 247

L4\_PROTO\_SIGMA0  
     l4\_mshtag\_api, 247

L4\_PROTO\_SYS\_EXCEPTION  
     l4\_mshtag\_api, 247

L4\_PROTO\_TASK  
     l4\_mshtag\_api, 247

L4\_PROTO\_THREAD  
     l4\_mshtag\_api, 247

L4\_RCV\_ITEM\_LOCAL\_ID  
     l4\_msgetitem\_api, 141

L4\_RCV\_ITEM\_SINGLE\_CAP  
     l4\_msgetitem\_api, 141

l4\_scheduler\_api  
     L4\_SCHEDULER\_IDLE\_TIME\_OP, 216  
     L4\_SCHEDULER\_INFO\_OP, 216  
     L4\_SCHEDULER\_RUN\_THREAD\_OP, 216

L4\_SCHEDULER\_IDLE\_TIME\_OP  
     l4\_scheduler\_api, 216

L4\_SCHEDULER\_INFO\_OP  
     l4\_scheduler\_api, 216

L4\_SCHEDULER\_RUN\_THREAD\_OP  
     l4\_scheduler\_api, 216

L4\_SCHEDULER\_INFO\_OP  
     l4\_scheduler\_api, 216

L4\_SCHEDULER\_RUN\_THREAD\_OP  
     l4\_scheduler\_api, 216

L4\_SYSF\_CALL  
     l4\_ipc\_api, 184

L4\_SYSF\_NONE  
     l4\_ipc\_api, 184

L4\_SYSF\_OPEN\_WAIT  
     l4\_ipc\_api, 184

L4\_SYSF\_RECV  
     l4\_ipc\_api, 184

L4\_SYSF\_REPLY  
     l4\_ipc\_api, 184

L4\_SYSF\_REPLY\_AND\_WAIT  
     l4\_ipc\_api, 185

L4\_SYSF\_SEND  
     l4\_ipc\_api, 184

L4\_SYSF\_SEND\_AND\_WAIT  
     l4\_ipc\_api, 185

L4\_SYSF\_WAIT  
     l4\_ipc\_api, 185

l4\_task\_api  
     L4\_FP\_ALL\_SPACES, 221  
     L4\_FP\_DELETE\_OBJ, 221  
     L4\_FP\_OTHER\_SPACES, 221

l4\_thread\_api  
     L4\_THREAD\_CONTROL\_ALIEN, 230  
     L4\_THREAD\_CONTROL\_BIND\_TASK,  
         230  
     L4\_THREAD\_CONTROL\_MR\_IDX\_-  
         BIND\_TASK, 230  
     L4\_THREAD\_CONTROL\_MR\_IDX\_-  
         BIND\_UTCB, 230  
     L4\_THREAD\_CONTROL\_MR\_IDX\_EXC\_-  
         HANDLER, 230  
     L4\_THREAD\_CONTROL\_MR\_IDX\_-  
         FLAG\_VALS, 230  
     L4\_THREAD\_CONTROL\_MR\_IDX\_-  
         FLAGS, 230  
     L4\_THREAD\_CONTROL\_MR\_IDX\_-  
         PAGER, 230  
     L4\_THREAD\_CONTROL\_OP, 229  
     L4\_THREAD\_CONTROL\_SET\_EXC\_-  
         HANDLER, 230  
     L4\_THREAD\_CONTROL\_SET\_PAGER, 230  
     L4\_THREAD\_CONTROL\_SET\_PAGER,  
         230  
     L4\_THREAD\_CONTROL\_SET\_PAGER,  
         230  
     L4\_THREAD\_EX\_REGS\_CANCEL, 230  
     L4\_THREAD\_EX\_REGS\_OP, 229  
     L4\_THREAD\_EX\_REGS\_TRIGGER\_-  
         EXCEPTION, 230  
     L4\_THREAD\_GDT\_X86\_OP, 229  
     L4\_THREAD\_MODIFY\_SENDER, 229

L4\_THREAD\_OPCODE\_MASK, 229  
L4\_THREAD\_REGISTER\_DELETE\_IRQ,  
    229  
L4\_THREAD\_STATS\_OP, 229  
L4\_THREAD\_SWITCH\_OP, 229  
L4\_THREAD\_VCPU\_CONTROL, 229  
L4\_THREAD\_VCPU\_RESUME\_OP, 229  
L4\_THREAD\_CONTROL\_ALIEN  
    l4\_thread\_api, 230  
L4\_THREAD\_CONTROL\_BIND\_TASK  
    l4\_thread\_api, 230  
L4\_THREAD\_CONTROL\_MR\_IDX\_BIND\_-  
    TASK  
    l4\_thread\_api, 230  
L4\_THREAD\_CONTROL\_MR\_IDX\_BIND\_-  
    UTCB  
    l4\_thread\_api, 230  
L4\_THREAD\_CONTROL\_MR\_IDX\_EXC\_-  
    HANDLER  
    l4\_thread\_api, 230  
L4\_THREAD\_CONTROL\_MR\_IDX\_FLAG\_-  
    VALS  
    l4\_thread\_api, 230  
L4\_THREAD\_CONTROL\_MR\_IDX\_FLAGS  
    l4\_thread\_api, 230  
L4\_THREAD\_CONTROL\_MR\_IDX\_PAGER  
    l4\_thread\_api, 230  
L4\_THREAD\_CONTROL\_OP  
    l4\_thread\_api, 229  
L4\_THREAD\_CONTROL\_SET\_EXC\_-  
    HANDLER  
    l4\_thread\_api, 230  
L4\_THREAD\_CONTROL\_SET\_PAGER  
    l4\_thread\_api, 230  
L4\_THREAD\_CONTROL\_UX\_NATIVE  
    l4\_thread\_api, 230  
L4\_THREAD\_EX\_REGS\_CANCEL  
    l4\_thread\_api, 230  
L4\_THREAD\_EX\_REGS\_OP  
    l4\_thread\_api, 229  
L4\_THREAD\_EX\_REGS\_TRIGGER\_-  
    EXCEPTION  
    l4\_thread\_api, 230  
L4\_THREAD\_GDT\_X86\_OP  
    l4\_thread\_api, 229  
L4\_THREAD MODIFY\_SENDER  
    l4\_thread\_api, 229  
L4\_THREAD\_OPCODE\_MASK  
    l4\_thread\_api, 229  
L4\_THREAD\_REGISTER\_DELETE\_IRQ  
    l4\_thread\_api, 229  
L4\_THREAD\_STATS\_OP  
    l4\_thread\_api, 229  
L4\_THREAD\_SWITCH\_OP  
    l4\_thread\_api, 229  
L4\_THREAD\_API, 229  
L4\_THREAD\_VCPU\_CONTROL  
    l4\_thread\_api, 229  
L4\_THREAD\_VCPU\_RESUME\_OP  
    l4\_thread\_api, 229  
L4\_TYPE\_VHW\_FRAMEBUFFER  
    l4\_kip\_vhw\_api, 278  
L4\_TYPE\_VHW\_INPUT  
    l4\_kip\_vhw\_api, 278  
L4\_TYPE\_VHW\_NET  
    l4\_kip\_vhw\_api, 278  
L4\_TYPE\_VHW\_NONE  
    l4\_kip\_vhw\_api, 278  
l4\_utcb\_api\_x86  
    L4\_UTCB\_BUF\_REGS\_OFFSET, 283  
    L4\_UTCB\_EXCEPTION\_REGS\_SIZE, 283  
    L4\_UTCB\_GENERIC\_BUFFERS\_SIZE, 283  
    L4\_UTCB\_GENERIC\_DATA\_SIZE, 283  
    L4\_UTCB\_INHERIT\_FPU, 283  
    L4\_UTCB\_MSG\_REGS\_OFFSET, 283  
    L4\_UTCB\_OFFSET, 283  
    L4\_UTCB\_THREAD\_REGS\_OFFSET, 283  
l4\_utcb\_br\_api  
    L4\_BDR\_IO\_SHIFT, 266  
    L4\_BDR\_MEM\_SHIFT, 266  
    L4\_BDR\_OBJ\_SHIFT, 266  
L4\_UTCB\_BUF\_REGS\_OFFSET  
    l4\_utcb\_api\_x86, 283  
L4\_UTCB\_EXCEPTION\_REGS\_SIZE  
    l4\_utcb\_api\_x86, 283  
L4\_UTCB\_GENERIC\_BUFFERS\_SIZE  
    l4\_utcb\_api\_x86, 283  
L4\_UTCB\_GENERIC\_DATA\_SIZE  
    l4\_utcb\_api\_x86, 283  
L4\_UTCB\_INHERIT\_FPU  
    l4\_utcb\_api\_x86, 283  
L4\_UTCB\_MSG\_REGS\_OFFSET  
    l4\_utcb\_api\_x86, 283  
L4\_UTCB\_OFFSET  
    l4\_utcb\_api\_x86, 283  
L4\_UTCB\_THREAD\_REGS\_OFFSET  
    l4\_utcb\_api\_x86, 283  
l4\_vcon\_api  
    L4\_VCON\_ECHO, 271  
    L4\_VCON\_GET\_ATTR\_OP, 272  
    L4\_VCON\_ICRNL, 271  
    L4\_VCON\_IGNCR, 271  
    L4\_VCON\_INLCR, 271  
    L4\_VCON\_OCRNL, 271  
    L4\_VCON\_ONLCR, 271  
    L4\_VCON\_ONLRET, 271  
    L4\_VCON\_SET\_ATTR\_OP, 272  
    L4\_VCON\_WRITE\_OP, 272  
    L4\_VCON\_WRITE\_SIZE, 271

L4\_VCON\_ECHO  
     l4\_vcon\_api, 271

L4\_VCON\_GET\_ATTR\_OP  
     l4\_vcon\_api, 272

L4\_VCON\_ICRNL  
     l4\_vcon\_api, 271

L4\_VCON\_IGNCR  
     l4\_vcon\_api, 271

L4\_VCON\_INLCR  
     l4\_vcon\_api, 271

L4\_VCON\_OCRNL  
     l4\_vcon\_api, 271

L4\_VCON\_ONLCR  
     l4\_vcon\_api, 271

L4\_VCON\_ONLRET  
     l4\_vcon\_api, 271

L4\_VCON\_SET\_ATTR\_OP  
     l4\_vcon\_api, 272

L4\_VCON\_WRITE\_OP  
     l4\_vcon\_api, 272

L4\_VCON\_WRITE\_SIZE  
     l4\_vcon\_api, 271

l4\_vcpu\_api  
     L4\_VCPU\_F\_DEBUG\_EXC, 276  
     L4\_VCPU\_F\_EXCEPTIONS, 276  
     L4\_VCPU\_F\_FPU\_ENABLED, 276  
     L4\_VCPU\_F\_IRQ, 276  
     L4\_VCPU\_F\_PAGEFAULTS, 276  
     L4\_VCPU\_F\_USER\_MODE, 276  
     L4\_VCPU\_OFFSET\_EXT\_STATE, 277  
     L4\_VCPU\_SF\_IRQ\_PENDING, 277

L4\_VCPU\_F\_DEBUG\_EXC  
     l4\_vcpu\_api, 276

L4\_VCPU\_F\_EXCEPTIONS  
     l4\_vcpu\_api, 276

L4\_VCPU\_F\_FPU\_ENABLED  
     l4\_vcpu\_api, 276

L4\_VCPU\_F\_IRQ  
     l4\_vcpu\_api, 276

L4\_VCPU\_F\_PAGEFAULTS  
     l4\_vcpu\_api, 276

L4\_VCPU\_F\_USER\_MODE  
     l4\_vcpu\_api, 276

L4\_VCPU\_OFFSET\_EXT\_STATE  
     l4\_vcpu\_api, 277

L4\_VCPU\_SF\_IRQ\_PENDING  
     l4\_vcpu\_api, 277

L4\_WHOLE\_ADDRESS\_SPACE  
     l4\_fpage\_api, 133

L4\_WHOLE\_IOADDRESS\_SPACE  
     l4\_fpage\_api, 134

l4\_addr\_consts\_t  
     l4\_memory\_api, 157

l4\_basic\_types

l4\_int16\_t, 396

l4\_int32\_t, 397

l4\_int64\_t, 397

l4\_int8\_t, 396

l4\_uint16\_t, 397

l4\_uint32\_t, 397

l4\_uint64\_t, 397

l4\_uint8\_t, 396

l4\_buf\_regs\_t, 683

l4\_buffer\_desc\_consts\_t  
     l4\_utcb\_br\_api, 266

l4\_busy\_wait\_ns  
     l4util\_tsc, 290

l4\_busy\_wait\_us  
     l4util\_tsc, 291

l4\_cache\_api  
     l4\_cache\_clean\_data, 153  
     l4\_cache\_coherent, 153  
     l4\_cache\_dma\_coherent, 154  
     l4\_cache\_flush\_data, 153  
     l4\_cache\_inv\_data, 153

l4\_cache\_clean\_data  
     l4\_cache\_api, 153

l4\_cache\_coherent  
     l4\_cache\_api, 153

l4\_cache\_dma\_coherent  
     l4\_cache\_api, 154

l4\_cache\_flush\_data  
     l4\_cache\_api, 153

l4\_cache\_inv\_data  
     l4\_cache\_api, 153

l4\_calibrate\_tsc  
     l4util\_tsc, 291

l4\_cap\_api  
     cap\_cast, 258  
     cap\_dynamic\_cast, 259  
     cap\_reinterpret\_cast, 258  
     l4\_cap\_consts\_t, 257  
     l4\_cap\_idx\_t, 256  
     l4\_capability\_equal, 260  
     l4\_default\_caps\_t, 257  
     l4\_is\_invalid\_cap, 260  
     l4\_is\_valid\_cap, 260

l4\_cap\_consts\_t  
     l4\_cap\_api, 257

L4\_cap\_fpage\_rights  
     l4\_fpage\_api, 133

l4\_cap\_idx\_t  
     l4\_cap\_api, 256

l4\_capability\_equal  
     l4\_cap\_api, 260

l4\_config  
     l4\_kernel\_info\_t, 691

l4\_debugger\_api

asm\_enter\_kdebug, 161, 163  
enter\_kdebug, 161, 162  
kd\_display, 162, 163  
ko, 162, 163  
l4\_debugger\_global\_id, 164  
l4\_debugger\_kobj\_to\_id, 165  
l4\_debugger\_set\_object\_name, 164  
l4kd\_inchar, 167  
outchar, 165  
outdec, 167  
outhex12, 166  
outhex16, 166  
outhex20, 166  
outhex32, 166  
outhex8, 167  
outnstring, 166  
outstring, 165  
l4\_debugger\_global\_id, 164  
    l4\_debugger\_api, 164  
l4\_debugger\_kobj\_to\_id, 165  
    l4\_debugger\_api, 165  
l4\_debugger\_set\_object\_name, 164  
    l4\_debugger\_api, 164  
**L4\_DECLARE\_CONSTRUCTOR**  
    l4sys\_defines, 111  
l4\_default\_caps\_t  
    l4\_cap\_api, 257  
**L4\_DISABLE\_COPY**  
    l4\_kernel\_object\_api, 206  
l4\_error  
    l4\_ipc\_err\_api, 195  
l4\_error\_api  
    l4\_error\_code\_t, 168  
l4\_error\_code\_t  
    l4\_error\_api, 168  
l4\_exc\_regs\_t, 684  
    flags, 686  
**L4\_EXPORT**  
    l4sys\_defines, 112  
l4\_factory\_api  
    l4\_factory\_create\_factory, 171  
    l4\_factory\_create\_gate, 172  
    l4\_factory\_create\_irq, 173  
    l4\_factory\_create\_semaphore, 172  
    l4\_factory\_create\_task, 170  
    l4\_factory\_create\_thread, 170  
    l4\_factory\_create\_vm, 174  
l4\_factory\_create\_factory  
    l4\_factory\_api, 171  
l4\_factory\_create\_gate  
    l4\_factory\_api, 172  
l4\_factory\_create\_irq  
    l4\_factory\_api, 173  
l4\_factory\_create\_semaphore

L4\_HIDDEN  
   l4sys\_defines, 112

l4\_icu\_api  
   l4\_icu\_bind, 177  
   L4\_icu\_flags, 177  
   l4\_icu\_info, 179  
   l4\_icu\_info\_t, 177  
   l4\_icu\_mask, 181  
   l4\_icu\_msi\_info, 180  
   l4\_icu\_set\_mode, 179  
   l4\_icu\_unbind, 178  
   l4\_icu\_unmask, 180

l4\_icu\_bind  
   l4\_icu\_api, 177

L4\_icu\_flags  
   l4\_icu\_api, 177

l4\_icu\_info  
   l4\_icu\_api, 179

l4\_icu\_info\_t, 687  
   features, 688  
   l4\_icu\_api, 177

l4\_icu\_mask  
   l4\_icu\_api, 181

l4\_icu\_msi\_info  
   l4\_icu\_api, 180

l4\_icu\_set\_mode  
   l4\_icu\_api, 179

l4\_icu\_unbind  
   l4\_icu\_api, 178

l4\_icu\_unmask  
   l4\_icu\_api, 180

l4\_int16\_t  
   l4\_basic\_types, 396

l4\_int32\_t  
   l4\_basic\_types, 397

l4\_int64\_t  
   l4\_basic\_types, 397

l4\_int8\_t  
   l4\_basic\_types, 396

l4\_iofpage  
   l4\_fpage\_api, 135

l4\_ipc  
   l4\_ipc\_api, 190

l4\_ipc\_api  
   l4\_ipc, 190  
   l4\_ipc\_call, 188  
   l4\_ipc\_receive, 187  
   l4\_ipc\_reply\_and\_wait, 189  
   l4\_ipc\_send, 185  
   l4\_ipc\_send\_and\_wait, 190  
   l4\_ipc\_sleep, 191  
   l4\_ipc\_wait, 186  
   l4\_sndfpage\_add, 192  
   l4\_syscall\_flags\_t, 184

l4\_ipc\_call  
   l4\_ipc\_api, 188

l4\_ipc\_err\_api  
   l4\_error, 195  
   l4\_ipc\_error, 195  
   l4\_ipc\_error\_code, 197  
   l4\_ipc\_is\_rcv\_error, 197  
   l4\_ipc\_is\_snd\_error, 196  
   l4\_ipc\_tcr\_error\_t, 194

l4\_ipc\_error  
   l4\_ipc\_err\_api, 195

l4\_ipc\_error\_code  
   l4\_ipc\_err\_api, 197

l4\_ipc\_gate\_bind\_thread  
   l4\_kernel\_object\_gate\_api, 108

l4\_ipc\_gate\_get\_infos  
   l4\_kernel\_object\_gate\_api, 109

L4\_ipc\_gate\_ops  
   l4\_kernel\_object\_gate\_api, 108

l4\_ipc\_is\_rcv\_error  
   l4\_ipc\_err\_api, 197

l4\_ipc\_is\_snd\_error  
   l4\_ipc\_err\_api, 196

l4\_ipc\_receive  
   l4\_ipc\_api, 187

l4\_ipc\_reply\_and\_wait  
   l4\_ipc\_api, 189

l4\_ipc\_send  
   l4\_ipc\_api, 185

l4\_ipc\_send\_and\_wait  
   l4\_ipc\_api, 190

l4\_ipc\_sleep  
   l4\_ipc\_api, 191

l4\_ipc\_tcr\_error\_t  
   l4\_ipc\_err\_api, 194

l4\_ipc\_timeout  
   l4\_timeout\_api, 145

L4\_IPC\_TIMEOUT\_0  
   l4\_timeout\_api, 145

l4\_ipc\_wait  
   l4\_ipc\_api, 186

l4\_irq\_api  
   l4\_irq\_attach, 200  
   l4\_irq\_chain, 200  
   l4\_irq\_detach, 201  
   L4\_irq\_flow\_type, 199  
   l4\_irq\_receive, 202  
   l4\_irq\_trigger, 202  
   l4\_irq\_unmask, 204  
   l4\_irq\_wait, 203

l4\_irq\_attach  
   l4\_irq\_api, 200

l4\_irq\_chain  
   l4\_irq\_api, 200

l4\_irq\_detach  
    l4\_irq\_api, 201  
L4\_irq\_flow\_type  
    l4\_irq\_api, 199  
l4\_irq\_receive  
    l4\_irq\_api, 202  
l4\_irq\_trigger  
    l4\_irq\_api, 202  
l4\_irq\_unmask  
    l4\_irq\_api, 204  
l4\_irq\_wait  
    l4\_irq\_api, 203  
l4\_is\_fpage\_writable  
    l4\_fpage\_api, 136  
l4\_is\_invalid\_cap  
    l4\_cap\_api, 260  
l4\_is\_valid\_cap  
    l4\_cap\_api, 260  
l4\_kernel\_info\_get\_mem\_desc\_end  
    l4\_kip\_memdesc\_api, 213  
l4\_kernel\_info\_get\_mem\_desc\_is\_virtual  
    l4\_kip\_memdesc\_api, 214  
l4\_kernel\_info\_get\_mem\_desc\_start  
    l4\_kip\_memdesc\_api, 213  
l4\_kernel\_info\_get\_mem\_desc\_subtype  
    l4\_kip\_memdesc\_api, 214  
l4\_kernel\_info\_get\_mem\_desc\_type  
    l4\_kip\_memdesc\_api, 214  
l4\_kernel\_info\_get\_num\_mem\_descs  
    l4\_kip\_memdesc\_api, 213  
l4\_kernel\_info\_mem\_desc\_t, 688  
    l4\_kip\_memdesc\_api, 212  
l4\_kernel\_info\_set\_mem\_desc  
    l4\_kip\_memdesc\_api, 213  
l4\_kernel\_info\_t, 689  
    kdebug\_config, 691  
    kdebug\_permission, 692  
    l4\_config, 691  
l4\_kernel\_info\_version\_offset  
    l4\_kip\_api, 210  
l4\_kernel\_object\_api  
    L4\_DISABLE\_COPY, 206  
    L4\_KOBJECT, 207  
    L4\_KOBJECT\_DISABLE\_COPY, 207  
l4\_kernel\_object\_gate\_api  
    l4\_ipc\_gate\_bind\_thread, 108  
    l4\_ipc\_gate\_get\_infos, 109  
    L4\_ipc\_gate\_ops, 108  
l4\_kip\_api  
    l4\_kernel\_info\_version\_offset, 210  
    l4\_kip\_version, 209  
    l4\_kip\_version\_string, 209  
l4\_kip\_memdesc\_api  
    l4\_kernel\_info\_get\_mem\_desc\_end, 213  
l4\_kernel\_info\_get\_mem\_desc\_is\_virtual, 214  
l4\_kernel\_info\_get\_mem\_desc\_start, 213  
l4\_kernel\_info\_get\_mem\_desc\_subtype, 214  
l4\_kernel\_info\_get\_mem\_desc\_type, 214  
l4\_kernel\_info\_get\_num\_mem\_descs, 213  
l4\_kernel\_info\_mem\_desc\_t, 212  
l4\_kernel\_info\_set\_mem\_desc, 213  
l4\_mem\_type\_t, 212  
l4\_kip\_version  
    l4\_kip\_api, 209  
l4\_kip\_version\_string  
    l4\_kip\_api, 209  
l4\_kip\_vhw\_api  
    l4\_vhw\_entry\_type, 278  
L4\_KOBJECT  
    l4\_kernel\_object\_api, 207  
L4\_KOBJECT\_DISABLE\_COPY  
    l4\_kernel\_object\_api, 207  
L4\_LOG2\_PAGESIZE  
    l4\_memory\_api, 156  
L4\_LOG2\_SUPERPAGESIZE  
    l4\_memory\_api, 156  
l4\_map\_control  
    l4\_msgitem\_api, 141  
l4\_map\_obj\_control  
    l4\_msgitem\_api, 141  
l4\_mem\_op\_api  
    L4\_mem\_op\_widths, 279  
    l4\_mem\_read, 279  
    l4\_mem\_write, 279  
L4\_mem\_op\_widths  
    l4\_mem\_op\_api, 279  
l4\_mem\_read  
    l4\_mem\_op\_api, 279  
l4\_mem\_type\_t  
    l4\_kip\_memdesc\_api, 212  
l4\_mem\_write  
    l4\_mem\_op\_api, 279  
l4\_memory\_api  
    l4\_addr\_consts\_t, 157  
    L4\_LOG2\_PAGESIZE, 156  
    L4\_LOG2\_SUPERPAGESIZE, 156  
    L4\_PAGEMASK, 156  
    l4\_round\_page, 158  
    l4\_round\_size, 158  
    L4\_SUPERPAGEMASK, 156  
    L4\_SUPERPAGESIZE, 156  
    l4\_trunc\_page, 157  
    l4\_trunc\_size, 157  
l4\_msgitem\_consts\_t  
    l4\_msgitem\_api, 141  
l4\_msgrgs\_t, 692  
l4\_msgitem\_api  
    l4\_map\_control, 141

l4\_map\_obj\_control, 141  
 l4\_msg\_item\_consts\_t, 141  
 l4\_mshtag  
     l4\_mshtag\_api, 248  
 l4\_mshtag\_api  
     l4\_mshtag, 248  
     l4\_mshtag\_flags, 247, 251  
     l4\_mshtag\_has\_error, 251  
     l4\_mshtag\_is\_exception, 253  
     l4\_mshtag\_is\_io\_page\_fault, 254  
     l4\_mshtag\_is\_page\_fault, 251  
     l4\_mshtag\_is\_preemption, 252  
     l4\_mshtag\_is\_sigma0, 254  
     l4\_mshtag\_is\_sys\_exception, 252  
     l4\_mshtag\_items, 250  
     l4\_mshtag\_label, 249  
     l4\_mshtag\_protocol, 247  
     l4\_mshtag\_t, 247  
     l4\_mshtag\_words, 250  
 l4\_mshtag\_flags  
     l4\_mshtag\_api, 247, 251  
 l4\_mshtag\_has\_error  
     l4\_mshtag\_api, 251  
 l4\_mshtag\_is\_exception  
     l4\_mshtag\_api, 253  
 l4\_mshtag\_is\_io\_page\_fault  
     l4\_mshtag\_api, 254  
 l4\_mshtag\_is\_page\_fault  
     l4\_mshtag\_api, 251  
 l4\_mshtag\_is\_preemption  
     l4\_mshtag\_api, 252  
 l4\_mshtag\_is\_sigma0  
     l4\_mshtag\_api, 254  
 l4\_mshtag\_is\_sys\_exception  
     l4\_mshtag\_api, 252  
 l4\_mshtag\_items  
     l4\_mshtag\_api, 250  
 l4\_mshtag\_label  
     l4\_mshtag\_api, 249  
 l4\_mshtag\_protocol  
     l4\_mshtag\_api, 247  
 l4\_mshtag\_t, 693  
     flags, 694  
     l4\_mshtag\_api, 247  
 l4\_mshtag\_words  
     l4\_mshtag\_api, 250  
 l4\_next\_period\_id  
     api\_calls\_rt\_sched, 128  
 L4\_NOTHROW  
     l4sysDefines, 112  
 l4\_ns\_to\_tsc  
     l4util\_tsc, 290  
 l4\_obj\_fpage  
     l4\_fpage\_api, 135  
 L4\_PAGEMASK  
     l4\_memory\_api, 156  
 l4\_preemption\_id  
     api\_calls\_rt\_sched, 128  
 l4\_rcv\_timeout  
     l4\_timeout\_api, 146  
 l4\_rdpmc  
     l4util\_tsc, 288  
 l4\_rdpmc\_32  
     l4util\_tsc, 289  
 l4\_rdtsc  
     l4util\_tsc, 288  
 l4\_rdtsc\_32  
     l4util\_tsc, 288  
 l4\_round\_page  
     l4\_memory\_api, 158  
 l4\_round\_size  
     l4\_memory\_api, 158  
 l4\_rt\_begin\_minimal\_periodic  
     api\_calls\_rt\_sched, 124  
 l4\_rt\_begin\_strictly\_periodic  
     api\_calls\_rt\_sched, 123  
 l4\_rt\_end\_periodic  
     api\_calls\_rt\_sched, 125  
 l4\_rt\_generic  
     api\_calls\_rt\_sched, 129  
 l4\_rt\_next\_period  
     api\_calls\_rt\_sched, 127  
 l4\_rt\_next\_reservation  
     api\_calls\_rt\_sched, 127  
 l4\_rt\_preemption\_t, 695  
 l4\_rt\_preemption\_val32\_t, 696  
 l4\_rt\_preemption\_val\_t, 696  
 l4\_rt\_remove  
     api\_calls\_rt\_sched, 125  
 l4\_rt\_set\_period  
     api\_calls\_rt\_sched, 126  
 l4\_sched\_cpu\_set  
     l4\_scheduler\_api, 216  
 l4\_sched\_cpu\_set\_t, 697  
 l4\_sched\_param\_t, 697  
 l4\_scheduler\_api  
     l4\_sched\_cpu\_set, 216  
     l4\_scheduler\_idle\_time, 218  
     l4\_scheduler\_info, 217  
     l4\_scheduler\_is\_online, 219  
     L4\_scheduler\_ops, 216  
     l4\_scheduler\_run\_thread, 218  
 l4\_scheduler\_idle\_time  
     l4\_scheduler\_api, 218  
 l4\_scheduler\_info  
     l4\_scheduler\_api, 217  
 l4\_scheduler\_is\_online  
     l4\_scheduler\_api, 219

L4\_scheduler\_ops  
    l4\_scheduler\_api, 216

l4\_scheduler\_run\_thread  
    l4\_scheduler\_api, 218

l4\_sleep\_forever  
    l4util\_api, 342

l4\_snd\_fpage\_t, 699

l4\_snd\_timeout  
    l4\_timeout\_api, 146

l4\_sndfpage\_add  
    l4\_ipc\_api, 192

L4\_SUPERPAGEMASK  
    l4\_memory\_api, 156

L4\_SUPERPAGESIZE  
    l4\_memory\_api, 156

l4\_syscall\_flags\_t  
    l4\_ipc\_api, 184

l4\_task\_add\_ku\_mem  
    l4\_task\_api, 225

l4\_task\_api  
    l4\_task\_add\_ku\_mem, 225  
    l4\_task\_cap\_equal, 225  
    l4\_task\_cap\_has\_child, 224  
    l4\_task\_cap\_valid, 224  
    l4\_task\_map, 221  
    l4\_task\_unmap, 222  
    l4\_task\_unmap\_batch, 223  
    l4\_unmap\_flags\_t, 221

l4\_task\_cap\_equal  
    l4\_task\_api, 225

l4\_task\_cap\_has\_child  
    l4\_task\_api, 224

l4\_task\_cap\_valid  
    l4\_task\_api, 224

l4\_task\_map  
    l4\_task\_api, 221

l4\_task\_unmap  
    l4\_task\_api, 222

l4\_task\_unmap\_batch  
    l4\_task\_api, 223

l4\_thread\_api  
    L4\_thread\_control\_flags, 229  
    L4\_thread\_control\_mr\_indices, 230  
    l4\_thread\_ex\_regs, 230  
    L4\_thread\_ex\_regs\_flags, 230  
    l4\_thread\_ex\_regs\_ret, 231  
    l4\_thread\_modify\_sender\_add, 237  
    l4\_thread\_modify\_sender\_commit, 238  
    l4\_thread\_modify\_sender\_start, 237  
    L4\_thread\_ops, 229  
    l4\_thread\_register\_del\_irq, 236  
    l4\_thread\_stats\_time, 233  
    l4\_thread\_switch, 232  
    l4\_thread\_vcpu\_control, 235

l4\_thread\_vcpu\_control\_ext, 235

l4\_thread\_vcpu\_resume\_commit, 234

l4\_thread\_vcpu\_resume\_start, 233

l4\_thread\_yield, 232

l4\_thread\_control\_alien  
    l4\_thread\_control\_api, 243

l4\_thread\_control\_api  
    l4\_thread\_control\_alien, 243  
    l4\_thread\_control\_bind, 242  
    l4\_thread\_control\_commit, 244  
    l4\_thread\_control\_exc\_handler, 241  
    l4\_thread\_control\_pager, 241  
    l4\_thread\_control\_start, 240  
    l4\_thread\_control\_ux\_host\_syscall, 243

l4\_thread\_control\_bind  
    l4\_thread\_control\_api, 242

l4\_thread\_control\_commit  
    l4\_thread\_control\_api, 244

l4\_thread\_control\_exc\_handler  
    l4\_thread\_control\_api, 241

L4\_thread\_control\_flags  
    l4\_thread\_api, 229

L4\_thread\_control\_mr\_indices  
    l4\_thread\_api, 230

l4\_thread\_control\_pager  
    l4\_thread\_control\_api, 241

l4\_thread\_control\_start  
    l4\_thread\_control\_api, 240

l4\_thread\_control\_ux\_host\_syscall  
    l4\_thread\_control\_api, 243

l4\_thread\_ex\_regs  
    l4\_thread\_api, 230

L4\_thread\_ex\_regs\_flags  
    l4\_thread\_api, 230

l4\_thread\_ex\_regs\_ret  
    l4\_thread\_api, 231

l4\_thread\_modify\_sender\_add  
    l4\_thread\_api, 237

l4\_thread\_modify\_sender\_commit  
    l4\_thread\_api, 238

l4\_thread\_modify\_sender\_start  
    l4\_thread\_api, 237

L4\_thread\_ops  
    l4\_thread\_api, 229

l4\_thread\_register\_del\_irq  
    l4\_thread\_api, 236

l4\_thread\_regs\_t, 700

l4\_thread\_stats\_time  
    l4\_thread\_api, 233

l4\_thread\_switch  
    l4\_thread\_api, 232

l4\_thread\_vcpu\_control  
    l4\_thread\_api, 235

l4\_thread\_vcpu\_control\_ext

l4\_thread\_api, 235  
 l4\_thread\_vcpu\_resume\_commit  
     l4\_thread\_api, 234  
 l4\_thread\_vcpu\_resume\_start  
     l4\_thread\_api, 233  
 l4\_thread\_yield  
     l4\_thread\_api, 232  
 l4\_timeout  
     l4\_timeout\_api, 146  
 l4\_timeout\_abs  
     l4\_timeout\_api, 148  
 l4\_timeout\_abs\_validity  
     l4\_timeout\_api, 145  
 l4\_timeout\_api  
     l4\_ipc\_timeout, 145  
     L4\_IPC\_TIMEOUT\_0, 145  
     l4\_rcv\_timeout, 146  
     l4\_snd\_timeout, 146  
     l4\_timeout, 146  
     l4\_timeout\_abs, 148  
     l4\_timeout\_abs\_validity, 145  
     l4\_timeout\_get, 148  
     l4\_timeout\_is\_absolute, 147  
     l4\_timeout\_rel, 145  
     l4\_timeout\_rel\_get, 147  
     l4\_timeout\_s, 145  
     l4\_timeout\_t, 145  
 l4\_timeout\_get  
     l4\_timeout\_api, 148  
 l4\_timeout\_is\_absolute  
     l4\_timeout\_api, 147  
 l4\_timeout\_rel  
     l4\_timeout\_api, 145  
 l4\_timeout\_rel\_get  
     l4\_timeout\_api, 147  
 l4\_timeout\_s, 701  
     l4\_timeout\_api, 145  
 l4\_timeout\_t, 701  
     l4\_timeout\_api, 145  
 l4\_tracebuffer\_status\_t, 703  
     cnt\_iobmap\_tlb\_flush, 707  
     size0, 707  
     size1, 707  
     tracebuffer0, 707  
     tracebuffer1, 707  
     version0, 707  
     version1, 707  
 l4\_tracebuffer\_status\_window\_t, 708  
 l4\_trunc\_page  
     l4\_memory\_api, 157  
 l4\_trunc\_size  
     l4\_memory\_api, 157  
 l4\_tsc\_init  
     l4util\_tsc, 292  
     l4\_tsc\_to\_ns  
         l4util\_tsc, 289  
     l4\_tsc\_to\_s\_and\_ns  
         l4util\_tsc, 289  
     l4\_tsc\_to\_us  
         l4util\_tsc, 289  
 l4\_uint16\_t  
     l4\_basic\_types, 397  
 l4\_uint32\_t  
     l4\_basic\_types, 397  
 l4\_uint64\_t  
     l4\_basic\_types, 397  
 l4\_uint8\_t  
     l4\_basic\_types, 396  
 l4\_unmap\_flags\_t  
     l4\_task\_api, 221  
 l4\_utcb\_api  
     l4\_utcb\_br, 263  
     l4\_utcb\_mr, 263  
     l4\_utcb\_t, 263  
     l4\_utcb\_tcr, 264  
 l4\_utcb\_api\_x86  
     L4\_utcb\_consts\_x86, 283  
 l4\_utcb\_br  
     l4\_utcb\_api, 263  
 l4\_utcb\_br\_api  
     l4\_buffer\_desc\_consts\_t, 266  
 L4\_utcb\_consts\_x86  
     l4\_utcb\_api\_x86, 283  
 l4\_utcb\_exc  
     l4\_utcb\_exc\_api, 268  
 l4\_utcb\_exc\_api  
     l4\_utcb\_exc, 268  
     l4\_utcb\_exc\_is\_pf, 269  
     l4\_utcb\_exc\_pc, 268  
     l4\_utcb\_exc\_pc\_set, 268  
 l4\_utcb\_exc\_is\_pf  
     l4\_utcb\_exc\_api, 269  
 l4\_utcb\_exc\_pc  
     l4\_utcb\_exc\_api, 268  
 l4\_utcb\_exc\_pc\_set  
     l4\_utcb\_exc\_api, 268  
 l4\_utcb\_mr  
     l4\_utcb\_api, 263  
 l4\_utcb\_t  
     l4\_utcb\_api, 263  
 l4\_utcb\_tcr  
     l4\_utcb\_api, 264  
 l4\_vcon\_api  
     l4\_vcon\_get\_attr, 274  
     L4\_vcon\_i\_flags, 271  
     L4\_vcon\_l\_flags, 271  
     L4\_vcon\_o\_flags, 271  
     L4\_vcon\_ops, 271

l4\_vcon\_read, 273  
l4\_vcon\_send, 272  
l4\_vcon\_set\_attr, 273  
l4\_vcon\_write, 272  
L4\_vcon\_write\_consts, 271  
l4\_vcon\_attr\_t, 708  
l4\_vcon\_get\_attr  
    l4\_vcon\_api, 274  
L4\_vcon\_i\_flags  
    l4\_vcon\_api, 271  
L4\_vcon\_l\_flags  
    l4\_vcon\_api, 271  
L4\_vcon\_o\_flags  
    l4\_vcon\_api, 271  
L4\_vcon\_ops  
    l4\_vcon\_api, 271  
l4\_vcon\_read  
    l4\_vcon\_api, 273  
l4\_vcon\_send  
    l4\_vcon\_api, 272  
l4\_vcon\_set\_attr  
    l4\_vcon\_api, 273  
l4\_vcon\_write  
    l4\_vcon\_api, 272  
L4\_vcon\_write\_consts  
    l4\_vcon\_api, 271  
l4\_vcpu\_api  
    L4\_vcpu\_state\_flags, 276  
    L4\_vcpu\_state\_offset, 277  
    L4\_vcpu\_sticky\_flags, 276  
l4\_vcpu\_ipc\_regs\_t, 709  
l4\_vcpu\_regs\_t, 711  
    ax, 713  
    bp, 713  
    bx, 713  
    cx, 713  
    di, 713  
    dx, 713  
    si, 713  
L4\_vcpu\_state\_flags  
    l4\_vcpu\_api, 276  
L4\_vcpu\_state\_offset  
    l4\_vcpu\_api, 277  
l4\_vcpu\_state\_t, 714  
L4\_vcpu\_sticky\_flags  
    l4\_vcpu\_api, 276  
l4\_vhw\_descriptor, 717  
    count, 719  
    descs, 719  
    magic, 719  
    version, 719  
l4\_vhw\_entry, 720  
    fd, 721  
    irq\_no, 721  
mem\_size, 721  
mem\_start, 721  
provider\_pid, 721  
type, 721  
l4\_vhw\_entry\_type  
    l4\_kip\_vhw\_api, 278  
l4\_vm\_run  
    l4\_vm\_tz\_api, 281  
l4\_vm\_state, 722  
l4\_vm\_svm\_vmcb\_control\_area, 722  
l4\_vm\_svm\_vmcb\_state\_save\_area, 722  
l4\_vm\_svm\_vmcb\_state\_save\_area\_seg, 724  
l4\_vm\_svm\_vmcb\_t, 724  
l4\_vm\_tz\_api  
    l4\_vm\_run, 281  
l4\_vm\_vmx\_api  
    l4\_vm\_vmx\_field\_len, 151  
    l4\_vm\_vmx\_field\_ptr, 151  
l4\_vm\_vmx\_field\_len  
    l4\_vm\_vmx\_api, 151  
l4\_vm\_vmx\_field\_ptr  
    l4\_vm\_vmx\_api, 151  
l4irq\_api\_async  
    l4irq\_request\_cap, 363  
l4irq\_api\_irq  
    l4irq\_attach, 356  
    l4irq\_attach\_ft, 356  
    l4irq\_attach\_thread, 357  
    l4irq\_attach\_thread\_ft, 357  
    l4irq\_detach, 358  
    l4irq\_unmask, 358  
    l4irq\_unmask\_and\_wait\_any, 357  
    l4irq\_wait, 357  
    l4irq\_wait\_any, 358  
l4irq\_api\_irq\_cap  
    l4irq\_attach\_cap, 361  
    l4irq\_attach\_cap\_ft, 361  
    l4irq\_attach\_thread\_cap, 361  
    l4irq\_attach\_thread\_cap\_ft, 361  
l4irq\_attach  
    l4irq\_api\_irq, 356  
l4irq\_attach\_cap  
    l4irq\_api\_irq\_cap, 361  
l4irq\_attach\_cap\_ft  
    l4irq\_api\_irq\_cap, 361  
l4irq\_attach\_ft  
    l4irq\_api\_irq, 356  
l4irq\_attach\_thread  
    l4irq\_api\_irq, 357  
l4irq\_attach\_thread\_cap  
    l4irq\_api\_irq\_cap, 361

l4irq\_attach\_thread\_cap\_ft  
     l4irq\_api\_irq\_cap, 361  
 l4irq\_attach\_thread\_ft  
     l4irq\_api\_irq, 357  
 l4irq\_detach  
     l4irq\_api\_irq, 358  
 l4irq\_release  
     l4irq\_api\_async, 360  
 l4irq\_request  
     l4irq\_api\_async, 359  
 l4irq\_request\_cap  
     l4irq\_api\_async\_cap, 363  
 l4irq\_unmask  
     l4irq\_api\_irq, 358  
 l4irq\_unmask\_and\_wait\_any  
     l4irq\_api\_irq, 357  
 l4irq\_wait  
     l4irq\_api\_irq, 357  
 l4irq\_wait\_any  
     l4irq\_api\_irq, 358  
 l4kd\_inchar  
     l4\_debugger\_api, 167  
 L4Re, 406  
 L4Re C Interface, 100  
 L4Re C++ Interface, 46  
 L4Re Capability API, 97  
 L4Re ELF Auxiliary Information, 83  
 L4Re Protocol identifiers, 95  
 L4Re Util C Interface, 102  
 L4Re Util C++ Interface, 49  
 L4Re::Cap\_alloc, 491  
     alloc, 492, 493  
     free, 493  
     get\_cap\_alloc, 493  
 L4Re::Console, 511  
 L4Re::Dataspace, 514  
     allocate, 520  
     clear, 519  
     copy\_in, 520  
     flags, 522  
     info, 523  
     map, 518  
     Map\_ro, 518  
     Map\_rw, 518  
     Map\_flags, 517  
     map\_region, 518  
     phys, 521  
     size, 522  
 L4Re::Dataspace::Stats, 856  
 L4Re::Debug\_obj, 528  
     debug, 530  
 L4Re::Env, 559  
     env, 563  
     factory, 564, 567  
     first\_free\_cap, 564, 568  
     first\_free\_utcb, 565, 568  
     get, 565  
     get\_cap, 566  
     initial\_caps, 565, 569  
     log, 564, 567  
     main\_thread, 564, 567  
     mem\_alloc, 563, 567  
     parent, 563, 566  
     rm, 563, 567  
     scheduler, 568  
     task, 564  
     utcb\_area, 565, 568  
 L4Re::Event, 570  
     get\_buffer, 573  
 L4Re::Event\_buffer\_t, 577  
     Event\_buffer\_t, 580  
     next, 580  
     put, 580  
 L4Re::Event\_buffer\_t::Event, 569  
 L4Re::Log, 756  
     print, 759  
     printfn, 759  
 L4Re::Mem\_alloc  
     Continuous, 763  
     Pinned, 763  
     Super\_pages, 763  
 L4Re::Mem\_alloc, 760  
     alloc, 763  
     free, 764  
     Mem\_alloc\_flags, 763  
 L4Re::Namespace, 778  
     query, 781, 782  
     Register\_flags, 781  
     register\_obj, 782  
     Ro, 781  
     Rw, 781  
     Strong, 781  
 L4Re::Parent, 797  
     signal, 800  
 L4Re::Rm, 812  
     attach, 819, 820  
     Attach\_flags, 817  
     Attach\_flags, 816  
     detach, 820, 821  
     Detach\_again, 816  
     Detach\_exact, 817  
     Detach\_free, 816  
     Detach\_overlap, 817  
     Detach\_flags, 817  
     Detach\_result, 816  
     Detached\_ds, 816  
     Eager\_map, 817  
     find, 822

free\_area, 819  
In\_area, 817  
Kept\_ds, 816  
Pager, 816  
Read\_only, 816  
Region\_flags, 816  
Region\_flags, 816  
reserve\_area, 817, 818  
Reserved, 816  
Search\_addr, 817  
Split\_ds, 816  
L4Re::Smart\_cap\_auto, 849  
L4Re::Util::Auto\_cap, 413  
L4Re::Util::Auto\_del\_cap, 414  
L4Re::Util::Cap\_alloc\_base, 494  
L4Re::Util::Counting\_cap\_alloc, 513  
L4Re::Util::Dataspace\_svr, 524  
    clear, 527  
    copy, 527  
    map, 525  
    map\_hook, 525  
    page\_shift, 527  
    phys, 526  
    release, 526  
    take, 526  
L4Re::Util::Event\_t  
    Mode\_irq, 587  
    Mode\_polling, 587  
L4Re::Util::Event\_buffer\_consumer\_t, 573  
    foreach\_available\_event, 576  
    process, 576  
L4Re::Util::Event\_buffer\_t, 581  
    attach, 584  
    buf, 584  
    detach, 585  
L4Re::Util::Event\_t, 585  
    buffer, 587  
    init, 587  
    irq, 587  
    Mode, 587  
L4Re::Util::Item\_alloc\_base, 668  
L4Re::Util::Names::Name, 778  
L4Re::Util::Ref\_cap, 807  
L4Re::Util::Ref\_del\_cap, 807  
L4Re::Util::Smart\_cap\_auto, 851  
L4Re::Util::Smart\_count\_cap, 851  
L4Re::Util::Vcon\_svr, 889  
    dispatch, 889  
L4Re::Util::Video::Goos\_svr, 618  
    dispatch, 621  
    get\_fb, 620  
    init\_infos, 622  
    refresh, 621  
    screen\_info, 621  
                view\_info, 621  
L4Re::Vfs, 407  
L4Re::Vfs::Be\_file, 454  
L4Re::Vfs::Be\_file\_system, 458  
    ~Be\_file\_system, 461  
    Be\_file\_system, 461  
    type, 461  
L4Re::Vfs::Directory, 539  
    faccessat, 541  
    link, 542  
    mkdir, 541  
    rename, 542  
    rmdir, 543  
    symlink, 543  
    unlink, 542  
L4Re::Vfs::File, 599  
L4Re::Vfs::File\_system, 601  
    mount, 604  
    type, 604  
L4Re::Vfs::Fs, 605  
    alloc\_fd, 606  
    free\_fd, 607  
    get\_file, 606  
    mount, 607  
    set\_fd, 607  
L4Re::Vfs::Generic\_file, 608  
    fchmod, 611  
    fstat64, 610  
    get\_status\_flags, 611  
    set\_status\_flags, 611  
    unlock\_all\_locks, 610  
L4Re::Vfs::Mman, 774  
L4Re::Vfs::Ops, 785  
L4Re::Vfs::Regular\_file, 808  
    data\_space, 810  
    fdatsync, 811  
    fsync, 811  
    ftruncate64, 811  
    get\_lock, 812  
    lseek64, 811  
    readv, 810  
    set\_lock, 812  
    writev, 811  
L4Re::Vfs::Special\_file, 852  
    ioctl, 854  
L4Re::Video::Color\_component, 504  
    Color\_component, 505  
    dump, 506  
    get, 506  
    operator==, 505  
    set, 506  
    shift, 505  
    size, 505  
L4Re::Video::Goos, 612

create\_buffer, 616  
 create\_view, 617  
 delete\_buffer, 616  
 delete\_view, 617  
 F\_auto\_refresh, 615  
 F\_dynamic\_buffers, 616  
 F\_dynamic\_views, 615  
 F\_pointer, 615  
 Flags, 615  
 get\_static\_buffer, 616  
 info, 616  
 view, 617  
**L4Re::Video::Goos::Info**, 632  
 auto\_refresh, 634  
**L4Re::Video::Pixel\_info**, 800  
 a, 804, 805  
 b, 804, 805  
 bits\_per\_pixel, 804  
 bytes\_per\_pixel, 804, 806  
 dump, 806  
 g, 804, 805  
 has\_alpha, 805  
 operator==, 806  
 Pixel\_info, 803  
 r, 804, 805  
**L4Re::Video::View**, 899  
 F\_above, 902  
 F\_dyn\_allocated, 902  
 F\_flags\_mask, 902  
 F\_fully\_dynamic, 902  
 F\_none, 902  
 F\_set\_background, 902  
 F\_set\_buffer, 902  
 F\_set\_buffer\_offset, 902  
 F\_set\_bytes\_per\_line, 902  
 F\_set\_flags, 902  
 F\_set\_pixel, 902  
 F\_set\_position, 902  
 Flags, 902  
 info, 902  
 refresh, 903  
 set\_info, 903  
 set\_viewport, 903  
 stack, 903  
 V\_flags, 902  
**L4Re::Video::View::Info**, 635  
 flags, 638  
**L4RE\_ELF\_AUX\_T\_KIP\_ADDR**  
 api\_l4re\_elf\_aux, 86  
**L4RE\_ELF\_AUX\_T\_NONE**  
 api\_l4re\_elf\_aux, 86  
**L4RE\_ELF\_AUX\_T\_STACK\_ADDR**  
 api\_l4re\_elf\_aux, 86  
**L4RE\_ELF\_AUX\_T\_STACK\_SIZE**  
 api\_l4re\_elf\_aux, 86  
**L4RE\_ELF\_AUX\_T\_VMA**  
 api\_l4re\_elf\_aux, 86  
**L4RE\_RM\_ATTACH\_FLAGS**  
 api\_l4re\_c\_rm, 64  
**L4RE\_RM\_EAGER\_MAP**  
 api\_l4re\_c\_rm, 64  
**L4RE\_RM\_IN\_AREA**  
 api\_l4re\_c\_rm, 64  
**L4RE\_RM\_NO\_ALIAS**  
 api\_l4re\_c\_rm, 64  
**L4RE\_RM\_OVERMAP**  
 api\_l4re\_c\_rm, 64  
**L4RE\_RM\_PAGER**  
 api\_l4re\_c\_rm, 64  
**L4RE\_RM\_READ\_ONLY**  
 api\_l4re\_c\_rm, 64  
**L4RE\_RM\_REGION\_FLAGS**  
 api\_l4re\_c\_rm, 64  
**L4RE\_RM\_RESERVED**  
 api\_l4re\_c\_rm, 64  
**L4RE\_RM\_SEARCH\_ADDR**  
 api\_l4re\_c\_rm, 64  
**l4re\_aux\_t**, 726  
**l4re\_cap\_api**  
 cap\_alloc, 98  
**l4re\_debug\_obj\_debug**  
 api\_l4re\_c\_debug, 53  
**l4re\_ds\_allocate**  
 api\_l4re\_c\_ds, 51  
**l4re\_ds\_clear**  
 api\_l4re\_c\_ds, 51  
**l4re\_ds\_copy\_in**  
 api\_l4re\_c\_ds, 51  
**l4re\_ds\_flags**  
 api\_l4re\_c\_ds, 51  
**l4re\_ds\_info**  
 api\_l4re\_c\_ds, 51  
**l4re\_ds\_phys**  
 api\_l4re\_c\_ds, 52  
**l4re\_ds\_size**  
 api\_l4re\_c\_ds, 51  
**l4re\_ds\_stats\_t**, 726  
**L4RE\_ELF\_AUX\_ELEM**  
 api\_l4re\_elf\_aux, 85  
**L4RE\_ELF\_AUX\_ELEM\_T**  
 api\_l4re\_elf\_aux, 85  
**l4re\_elf\_aux\_mword\_t**, 727  
**l4re\_elf\_aux\_t**, 727  
**l4re\_elf\_aux\_vma\_t**, 727  
**l4re\_env**  
 api\_l4re\_env, 88  
**l4re\_env\_cap\_entry\_t**, 728  
 flags, 729

l4re\_env\_cap\_entry\_t, 729  
l4re\_env\_cap\_entry\_t, 729  
l4re\_env\_t, 729  
    api\_l4re\_env, 88  
l4re\_event\_get  
    api\_l4re\_c\_event, 53  
l4re\_event\_t, 731  
l4re\_get\_env\_cap  
    api\_l4re\_env, 89  
l4re\_get\_env\_cap\_e  
    api\_l4re\_env, 89  
l4re\_get\_env\_cap\_l  
    api\_l4re\_env, 90  
l4re\_kip  
    api\_l4re\_env, 88  
l4re\_log\_print  
    api\_l4re\_c\_log, 54  
l4re\_log\_print\_srv  
    api\_l4re\_c\_log, 56  
l4re\_log\_printf  
    api\_l4re\_c\_log, 55  
l4re\_log\_printf\_srv  
    api\_l4re\_c\_log, 56  
l4re\_ma\_alloc  
    api\_l4re\_c\_mem\_alloc, 58  
l4re\_ma\_alloc\_srv  
    api\_l4re\_c\_mem\_alloc, 60  
l4re\_ma\_flags  
    api\_l4re\_c\_mem\_alloc, 58  
l4re\_ma\_free  
    api\_l4re\_c\_mem\_alloc, 59  
l4re\_ma\_free\_srv  
    api\_l4re\_c\_mem\_alloc, 60  
l4re\_ns\_query\_to\_srv  
    api\_l4re\_c\_ns, 62  
l4re\_ns\_register\_flags  
    api\_l4re\_c\_ns, 62  
l4re\_ns\_register\_obj\_srv  
    api\_l4re\_c\_ns, 62  
l4re\_rm\_attach  
    api\_l4re\_c\_rm, 65  
l4re\_rm\_attach\_srv  
    api\_l4re\_c\_rm, 71  
l4re\_rm\_detach  
    api\_l4re\_c\_rm, 67  
l4re\_rm\_detach\_ds  
    api\_l4re\_c\_rm, 67  
l4re\_rm\_detach\_ds\_unmap  
    api\_l4re\_c\_rm, 69  
l4re\_rm\_detach\_srv  
    api\_l4re\_c\_rm, 72  
l4re\_rm\_detach\_unmap  
    api\_l4re\_c\_rm, 68  
l4re\_rm\_find  
    api\_l4re\_c\_rm, 69  
l4re\_rm\_find\_srv  
    api\_l4re\_c\_rm, 72  
l4re\_rm\_flags\_t  
    api\_l4re\_c\_rm, 64  
l4re\_rm\_free\_area  
    api\_l4re\_c\_rm, 65  
l4re\_rm\_free\_area\_srv  
    api\_l4re\_c\_rm, 71  
l4re\_rm\_reserve\_area  
    api\_l4re\_c\_rm, 64  
l4re\_rm\_reserve\_area\_srv  
    api\_l4re\_c\_rm, 70  
l4re\_rm\_show\_lists  
    api\_l4re\_c\_rm, 70  
l4re\_util\_cap\_last  
    api\_l4re\_c\_util\_cap, 74  
l4re\_util\_cap\_release  
    api\_l4re\_c\_util, 103  
l4re\_util\_kumem  
    kumem\_alloc, 99  
l4re\_util\_kumem\_alloc  
    api\_l4re\_c\_util\_kumem\_alloc, 74  
l4re\_video\_color\_component\_t, 732  
l4re\_video\_goops\_create\_buffer  
    api\_l4re\_c\_video, 78  
l4re\_video\_goops\_create\_view  
    api\_l4re\_c\_video, 79  
l4re\_video\_goops\_delete\_buffer  
    api\_l4re\_c\_video, 79  
l4re\_video\_goops\_delete\_view  
    api\_l4re\_c\_video, 79  
l4re\_video\_goops\_get\_static\_buffer  
    api\_l4re\_c\_video, 79  
l4re\_video\_goops\_get\_view  
    api\_l4re\_c\_video, 80  
l4re\_video\_goops\_info  
    api\_l4re\_c\_video, 78  
l4re\_video\_goops\_info\_flags\_t  
    api\_l4re\_c\_video, 77  
l4re\_video\_goops\_info\_t, 732  
l4re\_video\_goops\_refresh  
    api\_l4re\_c\_video, 78  
l4re\_video\_pixel\_info\_t, 734  
l4re\_video\_view\_get\_info  
    api\_l4re\_c\_video, 80  
l4re\_video\_view\_info\_flags\_t  
    api\_l4re\_c\_video, 77  
l4re\_video\_view\_info\_t, 736  
l4re\_video\_view\_refresh  
    api\_l4re\_c\_video, 80  
l4re\_video\_view\_set\_info  
    api\_l4re\_c\_video, 80  
l4re\_video\_view\_set\_viewport

api\_l4re\_c\_video, 81  
 l4re\_video\_view\_stack  
     api\_l4re\_c\_video, 81  
 l4re\_video\_view\_t, 738  
     api\_l4re\_c\_video, 77  
 l4shmc\_add\_chunk  
     api\_l4shmc\_chunk, 381  
 l4shmc\_add\_signal  
     api\_l4shmc\_signal, 389  
 l4shmc\_area\_size  
     api\_l4shm, 379  
 l4shmc\_attach  
     api\_l4shm, 378  
 l4shmc\_attach\_signal  
     api\_l4shmc\_signal, 389  
 l4shmc\_attach\_signal\_to  
     api\_l4shmc\_signal, 389  
 l4shmc\_attach\_to  
     api\_l4shm, 378  
 l4shmc\_check\_magic  
     api\_l4shmc\_signal, 390  
 l4shmc\_chunk\_capacity  
     api\_l4shmc\_chunk, 382  
 l4shmc\_chunk\_consumed  
     api\_l4shmc\_chunk\_cons, 386  
 l4shmc\_chunk\_ptr  
     api\_l4shmc\_chunk, 382  
 l4shmc\_chunk\_ready  
     api\_l4shmc\_chunk\_prod, 383  
 l4shmc\_chunk\_ready\_sig  
     api\_l4shmc\_chunk\_prod, 384  
 l4shmc\_chunk\_signal  
     api\_l4shmc\_chunk, 382  
 l4shmc\_chunk\_size  
     api\_l4shmc\_chunk\_cons, 387  
 l4shmc\_chunk\_try\_to\_take  
     api\_l4shmc\_chunk\_prod, 383  
 l4shmc\_connect\_chunk\_signal  
     api\_l4shm, 379  
 l4shmc\_create  
     api\_l4shm, 378  
 l4shmc\_enable\_chunk  
     api\_l4shmc\_chunk\_cons, 385  
 l4shmc\_enable\_signal  
     api\_l4shmc\_signal\_cons, 392  
 l4shmc\_get\_chunk  
     api\_l4shmc\_chunk, 381  
 l4shmc\_get\_chunk\_to  
     api\_l4shmc\_chunk, 381  
 l4shmc\_get\_signal\_to  
     api\_l4shmc\_signal, 390  
 l4shmc\_is\_chunk\_clear  
     api\_l4shmc\_chunk\_prod, 384  
 l4shmc\_is\_chunk\_ready  
     api\_l4shmc\_chunk\_cons, 387  
 l4shmc\_signal\_cap  
     api\_l4shmc\_signal, 390  
 l4shmc\_trigger  
     api\_l4shmc\_signal\_prod, 391  
 l4shmc\_wait\_any  
     api\_l4shmc\_signal\_cons, 392  
 l4shmc\_wait\_any\_to  
     api\_l4shmc\_signal\_cons, 393  
 l4shmc\_wait\_any\_try  
     api\_l4shmc\_signal\_cons, 392  
 l4shmc\_wait\_chunk  
     api\_l4shmc\_chunk\_cons, 386  
 l4shmc\_wait\_chunk\_to  
     api\_l4shmc\_chunk\_cons, 386  
 l4shmc\_wait\_chunk\_try  
     api\_l4shmc\_chunk\_cons, 386  
 l4shmc\_wait\_signal  
     api\_l4shmc\_signal\_cons, 393  
 l4shmc\_wait\_signal\_to  
     api\_l4shmc\_signal\_cons, 393  
 l4shmc\_wait\_signal\_try  
     api\_l4shmc\_signal\_cons, 394  
 l4sigma0\_api  
     L4SIGMA0\_IPCERROR, 365  
     L4SIGMA0\_NOFPAGE, 365  
     L4SIGMA0\_NOTALIGNED, 365  
     L4SIGMA0\_OK, 365  
     L4SIGMA0\_SMALLERFPAGE, 365  
 L4SIGMA0\_IPCERROR  
     l4sigma0\_api, 365  
 L4SIGMA0\_NOFPAGE  
     l4sigma0\_api, 365  
 L4SIGMA0\_NOTALIGNED  
     l4sigma0\_api, 365  
 L4SIGMA0\_OK  
     l4sigma0\_api, 365  
 L4SIGMA0\_SMALLERFPAGE  
     l4sigma0\_api, 365  
 l4sigma0\_api  
     l4sigma0\_debug\_dump, 366  
     l4sigma0\_map\_anypage, 366  
     l4sigma0\_map\_errstr, 367  
     l4sigma0\_map\_iomem, 365  
     l4sigma0\_map\_kip, 365  
     l4sigma0\_map\_mem, 365  
     l4sigma0\_map\_tbuf, 366  
     l4sigma0\_new\_client, 367  
     l4sigma0\_return\_flags\_t, 364  
 l4sigma0\_debug\_dump  
     l4sigma0\_api, 366  
 l4sigma0\_map\_anypage  
     l4sigma0\_api, 366  
 l4sigma0\_map\_errstr

l4sigma0\_api, 367  
l4sigma0\_map\_iomem  
    l4sigma0\_api, 365  
l4sigma0\_map\_kip  
    l4sigma0\_api, 365  
l4sigma0\_map\_mem  
    l4sigma0\_api, 365  
l4sigma0\_map\_tbuf  
    l4sigma0\_api, 366  
l4sigma0\_new\_client  
    l4sigma0\_api, 367  
l4sigma0\_return\_flags\_t  
    l4sigma0\_api, 364  
l4sysDefines  
    L4\_DECLARE\_CONSTRUCTOR, 111  
    L4\_EXPORT, 112  
    L4\_HIDDEN, 112  
    L4\_NOTHROW, 112  
l4util\_add8  
    l4util\_atomic, 299  
l4util\_add8\_res  
    l4util\_atomic, 299  
l4util\_api  
    l4\_sleep\_forever, 342  
    l4util\_micros2l4to, 344  
    l4util\_splitlog2\_hdl, 342  
    l4util\_splitlog2\_size, 343  
l4util\_atomic  
    l4util\_add8, 299  
    l4util\_add8\_res, 299  
    l4util\_atomic\_add, 300  
    l4util\_atomic\_inc, 300  
    l4util\_cmpxchg, 297  
    l4util\_cmpxchg16, 296  
    l4util\_cmpxchg32, 296  
    l4util\_cmpxchg64, 296  
    l4util\_cmpxchg8, 297  
    l4util\_inc8, 299  
    l4util\_inc8\_res, 300  
    l4util\_xchg, 299  
    l4util\_xchg16, 298  
    l4util\_xchg32, 298  
    l4util\_xchg8, 299  
l4util\_atomic\_add  
    l4util\_atomic, 300  
l4util\_atomic\_inc  
    l4util\_atomic, 300  
l4util\_bitops  
    l4util\_bsf, 306  
    l4util\_bsr, 305  
    l4util\_btc, 305  
    l4util\_btr, 304  
    l4util\_bts  
    l4util\_bitops, 303  
l4util\_clear\_bit  
    l4util\_bitops, 302  
l4util\_cmpxchg  
    l4util\_atomic, 297  
l4util\_cmpxchg16  
    l4util\_atomic, 296  
l4util\_cmpxchg32  
    l4util\_atomic, 296  
l4util\_cmpxchg64  
    l4util\_atomic, 296  
l4util\_cmpxchg8  
    l4util\_atomic, 297  
l4util\_complement\_bit  
    l4util\_bitops, 303  
l4util\_cpu  
    l4util\_cpu\_capabilities, 284  
    l4util\_cpu\_capabilities\_nocheck, 285  
    l4util\_cpu\_has\_cpuid, 284  
l4util\_cpu\_capabilities  
    l4util\_cpu, 284  
l4util\_cpu\_capabilities\_nocheck  
    l4util\_cpu, 285  
l4util\_cpu\_has\_cpuid  
    l4util\_cpu, 284  
l4util\_elf  
    DF\_1\_CONFALT, 331  
    DF\_1\_DIRECT, 331  
    DF\_1\_DISPRELDNE, 332  
    DF\_1\_DISPRELPND, 332  
    DF\_1\_ENDFILTEE, 332  
    DF\_1\_GLOBAL, 330  
    DF\_1\_GROUP, 330  
    DF\_1\_INTERPOSE, 331  
    DF\_1\_LOADFLTR, 331  
    DF\_1\_NODEFLIB, 331  
    DF\_1\_NODEDELETE, 331  
    DF\_1\_NODUMP, 331  
    DF\_1\_NOOPEN, 331

DF\_1\_NOW, 330  
 DF\_1\_ORIGIN, 331  
 DF\_P1\_GROUPEPERM, 332  
 DF\_P1\_LAZYLOAD, 332  
 DT\_HIPROC, 330  
 DT\_LOPROC, 330  
 DT\_NULL, 330  
 EI\_CLASS, 324  
 EI\_DATA, 324  
 EI\_OSABI, 326  
 EI\_PAD, 328  
 EI\_VERSION, 325, 326  
 ELFCLASSNONE, 324  
 ELFDATA2LSB, 325  
 ELFDATA2MSB, 325  
 ELFDATANONE, 325  
 ELFOSABI\_AIX, 327  
 ELFOSABI\_FREEBSD, 327  
 ELFOSABI\_HPUX, 326, 327  
 ELFOSABI\_IRIX, 327  
 ELFOSABI\_LINUX, 327  
 ELFOSABI\_MODESTO, 327  
 ELFOSABI\_NETBSD, 327  
 ELFOSABI\_OPENBSD, 328  
 ELFOSABI\_SOLARIS, 327  
 ELFOSABI\_SYSV, 326  
 ELFOSABI\_TRU64, 327  
 EM\_ARC, 328  
 NT\_VERSION, 330  
 PT\_GNU\_EH\_FRAME, 329  
 PT\_GNU\_RELRO, 329  
 PT\_GNU\_STACK, 329  
 PT\_HIOS, 329  
 PT\_HIPROC, 329  
 PT\_L4\_AUX, 330  
 PT\_L4\_KIP, 329  
 PT\_L4\_STACK, 329  
 PT\_LOOS, 329  
 PT\_LOPROC, 329  
 SHF\_GROUP, 328  
 SHF\_MASKOS, 328  
 SHF\_TLS, 328  
 SHT\_NUM, 328  
 l4util\_find\_first\_set\_bit  
     l4util\_bitops, 306  
 l4util\_find\_first\_zero\_bit  
     l4util\_bitops, 306  
 l4util\_idt\_desc\_t, 739  
 l4util\_idt\_header\_t, 739  
 l4util\_in16  
     l4util\_portio, 345  
 l4util\_in32  
     l4util\_portio, 345  
 l4util\_in8

l4util\_portio, 345  
 l4util\_inc8  
     l4util\_atomic, 299  
 l4util\_inc8\_res  
     l4util\_atomic, 300  
 l4util\_ins16  
     l4util\_portio, 346  
 l4util\_ins32  
     l4util\_portio, 346  
 l4util\_ins8  
     l4util\_portio, 346  
 l4util\_kip\_api  
     l4util\_kip\_for\_each\_feature, 333  
     l4util\_kip\_kernel\_abi\_version, 334  
     l4util\_kip\_kernel\_has\_feature, 333  
     l4util\_kip\_kernel\_is\_ux, 333  
     l4util\_memdesc\_vm\_high, 334  
 l4util\_kip\_for\_each\_feature  
     l4util\_kip\_api, 333  
 l4util\_kip\_kernel\_abi\_version  
     l4util\_kip\_api, 334  
 l4util\_kip\_kernel\_has\_feature  
     l4util\_kip\_api, 333  
 l4util\_kip\_kernel\_is\_ux  
     l4util\_kip\_api, 333  
 l4util\_mb\_addr\_range\_t, 741  
 l4util\_mb\_apm\_t, 741  
 l4util\_mb\_drive\_t, 742  
     drive\_cylinders, 742  
     drive\_mode, 742  
     drive\_number, 742  
 l4util\_mb\_info\_t, 743  
 l4util\_mb\_mod\_t, 744  
     mod\_end, 745  
     mod\_start, 745  
 l4util\_mb\_vbe\_ctrl\_t, 745  
 l4util\_mb\_vbe\_mode\_t, 746  
 l4util\_memdesc\_vm\_high  
     l4util\_kip\_api, 334  
 l4util\_micros2l4to  
     l4util\_api, 344  
 l4util\_next\_power2  
     l4util\_bitops, 307  
 l4util\_out16  
     l4util\_portio, 347  
 l4util\_out32  
     l4util\_portio, 347  
 l4util\_out8  
     l4util\_portio, 347  
 l4util\_parse\_cmd  
     parse cmdline, 335  
 l4util\_portio  
     l4util\_in16, 345  
     l4util\_in32, 345

l4util\_in8, 345  
l4util\_ins16, 346  
l4util\_ins32, 346  
l4util\_ins8, 346  
l4util\_out16, 347  
l4util\_out32, 347  
l4util\_out8, 347  
l4util\_rand  
    l4util\_random, 337  
l4util\_random  
    l4util\_rand, 337  
    l4util\_srand, 337  
l4util\_set\_bit  
    l4util\_bitops, 302  
l4util\_splitlog2\_hdl  
    l4util\_api, 342  
l4util\_splitlog2\_size  
    l4util\_api, 343  
l4util\_srand  
    l4util\_random, 337  
l4util\_test\_bit  
    l4util\_bitops, 303  
l4util\_tsc  
    l4\_busy\_wait\_ns, 290  
    l4\_busy\_wait\_us, 291  
    l4\_calibrate\_tsc, 291  
    l4\_get\_hz, 293  
    l4\_ns\_to\_tsc, 290  
    l4\_rdpmc, 288  
    l4\_rdpmc\_32, 289  
    l4\_rdtsc, 288  
    l4\_rdtsc\_32, 288  
    l4\_tsc\_init, 292  
    l4\_tsc\_to\_ns, 289  
    l4\_tsc\_to\_s\_and\_ns, 289  
    l4\_tsc\_to\_us, 289  
l4util\_xchg  
    l4util\_atomic, 299  
l4util\_xchg16  
    l4util\_atomic, 298  
l4util\_xchg32  
    l4util\_atomic, 298  
l4util\_xchg8  
    l4util\_atomic, 299  
L4vcpu::State, 854  
    add, 855  
    clear, 855  
    set, 855  
    State, 855  
L4vcpu::Vcpu, 890  
    cast, 898  
    entry\_ip, 897  
    entry\_sp, 897  
    ext\_alloc, 898  
                halt, 896  
                i, 897  
                irq\_disable\_save, 894  
                irq\_enable, 895  
                irq\_restore, 895  
                is\_irq\_entry, 896  
                is\_page\_fault\_entry, 896  
                r, 896, 897  
                saved\_state, 895  
                state, 894, 895  
                task, 896  
L4VCPU\_IRQ\_STATE\_DISABLED  
    api\_libvcpu, 371  
L4VCPU\_IRQ\_STATE\_ENABLED  
    api\_libvcpu, 371  
l4vcpu\_ext\_alloc  
    api\_libvcpu\_ext, 376  
l4vcpu\_halt  
    api\_libvcpu, 374  
l4vcpu\_irq\_disable  
    api\_libvcpu, 371  
l4vcpu\_irq\_disable\_save  
    api\_libvcpu, 372  
l4vcpu\_irq\_enable  
    api\_libvcpu, 373  
l4vcpu\_irq\_restore  
    api\_libvcpu, 374  
l4vcpu\_irq\_state\_t  
    api\_libvcpu, 371  
l4vcpu\_is\_irq\_entry  
    api\_libvcpu, 375  
l4vcpu\_is\_page\_fault\_entry  
    api\_libvcpu, 375  
l4vcpu\_print\_state  
    api\_libvcpu, 375  
l4vcpu\_state  
    api\_libvcpu, 371  
link  
    L4Re::Vfs::Directory, 542  
List\_alloc  
    cxx::List\_alloc, 750  
log  
    L4Re::Env, 564, 567  
Log interface, 54  
Logging interface, 92  
Low-Level Thread Functions, 338  
lower\_bound\_node  
    cxx::Avl\_map, 424  
    cxx::Avl\_set, 432  
    cxx::Bits::Bst, 480  
lseek64  
    L4Re::Vfs::Regular\_file, 811  
Machine Restarting Function, 338

magic  
     l4\_vhw\_descriptor, 719  
 main\_thread  
     L4Re::Env, 564, 567  
 map  
     L4::Task, 863  
     L4Re::Dataspace, 518  
     L4Re::Util::Dataspace\_svr, 525  
 Map\_ro  
     L4Re::Dataspace, 518  
 Map\_rw  
     L4Re::Dataspace, 518  
 Map\_flags  
     L4Re::Dataspace, 517  
 map\_hook  
     L4Re::Util::Dataspace\_svr, 525  
 map\_region  
     L4Re::Dataspace, 518  
 mask  
     L4::Icu, 628  
 max  
     cxx\_api, 46  
 max\_free\_slabs  
     cxx::Base\_slab, 448  
     cxx::Base\_slab\_static, 452  
 Mem\_alloc  
     api\_l4re\_protocols, 96  
 mem\_alloc  
     L4Re::Env, 563, 567  
 Mem\_alloc\_flags  
     L4Re::Mem\_alloc, 763  
 Mem\_desc  
     L4::Kip::Mem\_desc, 765  
 mem\_size  
     l4\_vhw\_entry, 721  
 mem\_start  
     l4\_vhw\_entry, 721  
 Memory allocator, 57  
 Memory allocator API, 93  
 Memory descriptors (C version), 211  
 Memory operations., 278  
 Memory related, 154  
 Message Items, 140  
 Message Registers (MRs), 265  
 Message Tag, 245  
 min  
     cxx\_api, 46  
 mkdir  
     L4Re::Vfs::Directory, 541  
 mod\_end  
     l4util\_mb\_mod\_t, 745  
 mod\_start  
     l4util\_mb\_mod\_t, 745  
 Mode  
     L4Re::Util::Event\_t, 587  
 Mode\_irq  
     L4Re::Util::Event\_t, 587  
 Mode\_polling  
     L4Re::Util::Event\_t, 587  
 modify\_senders  
     L4::Thread, 878  
 mount  
     L4Re::Vfs::File\_system, 604  
     L4Re::Vfs::Fs, 607  
 move  
     L4::Cap, 491  
 Msg\_ptr  
     L4::Ipc::Msg\_ptr, 777  
 msi\_info  
     L4::Icu, 627

N

cxx::Bits::Direction, 538  
 Name-space API, 94  
 Namespace  
     api\_l4re\_protocols, 96  
 Namespace interface, 61  
 next  
     L4Re::Event\_buffer\_t, 580  
 No\_init  
     L4::Cap\_base, 498  
 No\_init\_type  
     L4::Cap\_base, 498  
 NT\_VERSION  
     l4util\_elf, 330  
 num\_interfaces  
     L4::Meta, 772

Object Invocation, 182

object\_size  
     cxx::Base\_slab, 448  
     cxx::Base\_slab\_static, 452  
 objects\_per\_slab  
     cxx::Base\_slab, 448  
     cxx::Base\_slab\_static, 452  
 operator l4\_mshtag\_t  
     L4::Factory::S, 828  
 operator new  
     cxx\_api, 46  
 operator Priv\_type \*  
     cxx::Auto\_ptr, 419  
 operator<<  
     L4::Factory::S, 828  
 operator\*  
     cxx::Auto\_ptr, 418  
 operator()  
     cxx::Pair\_first\_compare, 797  
 operator->

cxx::Auto\_ptr, 418  
operator=     cxx::Auto\_ptr, 418  
operator==     L4Re::Video::Color\_component, 505  
                L4Re::Video::Pixel\_info, 806  
outchar     l4\_debugger\_api, 165  
outdec     l4\_debugger\_api, 167  
outhex12     l4\_debugger\_api, 166  
outhex16     l4\_debugger\_api, 166  
outhex20     l4\_debugger\_api, 166  
outhex32     l4\_debugger\_api, 166  
outhex8     l4\_debugger\_api, 167  
outnstring     l4\_debugger\_api, 166  
outstring     l4\_debugger\_api, 165  
  
page\_shift     L4Re::Util::Dataspace\_svr, 527  
Pager     L4Re::Rm, 816  
pager     L4::Thread::Attr, 410, 411  
Pair     cxx::Pair, 796  
Pair\_first\_compare     cxx::Pair\_first\_compare, 797  
Parent     api\_l4re\_protocols, 96  
parent     L4Re::Env, 563, 566  
Parent API, 94  
parse\_cmdline     l4util\_parse\_cmd, 335  
phys     L4Re::Dataspace, 521  
           L4Re::Util::Dataspace\_svr, 526  
Pinned     L4Re::Mem\_alloc, 763  
Pixel\_info     L4Re::Video::Pixel\_info, 803  
print     L4Re::Log, 759  
printf     L4Re::Log, 759  
Priority related functions, 336  
  
process  
    L4Re::Util::Event\_buffer\_consumer\_t, 576  
Producer, 383, 391  
Protocols  
    api\_l4re\_protocols, 96  
provider\_pid  
    l4\_vhw\_entry, 721  
PT\_GNU\_EH\_FRAME  
    l4util\_elf, 329  
PT\_GNU\_RELRO  
    l4util\_elf, 329  
PT\_GNU\_STACK  
    l4util\_elf, 329  
PT\_HIOS  
    l4util\_elf, 329  
PT\_HIPROC  
    l4util\_elf, 329  
PT\_L4\_AUX  
    l4util\_elf, 330  
PT\_L4\_KIP  
    l4util\_elf, 329  
PT\_L4\_STACK  
    l4util\_elf, 329  
PT\_LOOS  
    l4util\_elf, 329  
PT\_LOPROC  
    l4util\_elf, 329  
push\_back  
    cxx::List, 748  
    cxx::List\_item, 754  
push\_front  
    cxx::List, 748  
    cxx::List\_item, 754  
put  
    L4::Ipc::Ostream, 791  
    L4Re::Event\_buffer\_t, 580  
  
query  
    L4Re::Namespace, 781, 782  
query\_log\_name  
    L4::Debugger, 534  
query\_log\_typeid  
    L4::Debugger, 534  
  
R  
    cxx::Bits::Direction, 538  
r  
    L4Re::Video::Pixel\_info, 804, 805  
    L4vcpu::Vcpu, 896, 897  
Random number support, 337  
rbegin  
    cxx::Avl\_set, 434, 435  
    cxx::Bits::Bst, 478, 479  
read

L4::Vcon, 886  
 Read\_only  
     L4Re::Rm, 816  
 ready  
     L4Re::Vfs::Regular\_file, 810  
 Realtime API, 198  
 receive  
     L4::Ipc::Istream, 667  
     L4::Irq, 656  
 Ref\_type  
     cxx::Auto\_ptr, 417  
 refresh  
     L4Re::Util::Video::Goos\_svr, 621  
     L4Re::Video::View, 903  
 Region map API, 96  
 Region map interface, 62  
 Region\_flags  
     L4Re::Rm, 816  
 Region\_flags  
     L4Re::Rm, 816  
 register\_del\_irq  
     L4::Thread, 877  
 Register\_flags  
     L4Re::Namespace, 781  
 register\_obj  
     L4Re::Namespace, 782  
 release  
     cxx::Auto\_ptr, 418  
     L4Re::Util::Dataspace\_svr, 526  
 remove  
     cxx::Avl\_map, 424  
     cxx::Avl\_set, 432  
     cxx::Avl\_tree, 440  
     cxx::List, 749  
     cxx::List\_item, 755  
 remove\_me  
     cxx::List\_item, 754  
     cxx::List\_item::Iter, 675  
     cxx::List\_item::T\_iter, 860  
 rename  
     L4Re::Vfs::Directory, 542  
 rend  
     cxx::Avl\_set, 434, 435  
     cxx::Bits::Bst, 479  
 Reply\_compound  
     L4::Ipc\_svr, 406  
 Reply\_separate  
     L4::Ipc\_svr, 406  
 reply\_and\_wait  
     L4::Ipc::Iostream, 647, 648  
 Reply\_mode  
     L4::Ipc\_svr, 405  
 reserve\_area  
     L4Re::Rm, 817, 818  
 Reserved  
     L4Re::Rm, 816  
 reset  
     L4::Ipc::Iostream, 646  
     L4::Ipc::Istream, 663  
 Rm  
     api\_l4re\_protocols, 96  
 rm  
     L4Re::Env, 563, 567  
 rmdir  
     L4Re::Vfs::Directory, 543  
 Ro  
     L4Re::Namespace, 781  
 run  
     L4::Vm, 907  
 run\_thread  
     L4::Scheduler, 832  
 Rw  
     L4Re::Namespace, 781  
 S  
     L4::Factory::S, 827  
 saved\_state  
     L4vcpu::Vcpu, 895  
 scan\_zero  
     cxx::Bitmap\_base, 468  
 Scheduler, 214  
 scheduler  
     L4Re::Env, 568  
 screen\_info  
     L4Re::Util::Video::Goos\_svr, 621  
 Search\_addr  
     L4Re::Rm, 817  
 send  
     L4::Ipc::Ostream, 792  
     L4::Vcon, 885  
 Server  
     L4::Server, 836  
 set  
     L4::Kip::Mem\_desc, 769  
     L4Re::Video::Color\_component, 506  
     L4vcpu::State, 855  
 set\_attr  
     L4::Vcon, 887  
 set\_bit  
     cxx::Bitmap\_base, 467  
 set\_fd  
     L4Re::Vfs::Fs, 607  
 set\_info  
     L4Re::Video::View, 903  
 set\_lock  
     L4Re::Vfs::Regular\_file, 812  
 set\_mode  
     L4::Icu, 629

set\_object\_name  
    L4::Debugger, 533

set\_status\_flags  
    L4Re::Vfs::Generic\_file, 611

set\_viewport  
    L4Re::Video::View, 903

Shared Memory Library, 377

SHF\_GROUP  
    l4util\_elf, 328

SHF\_MASKOS  
    l4util\_elf, 328

SHF\_TLS  
    l4util\_elf, 328

shift  
    L4Re::Video::Color\_component, 505

SHT\_NUM  
    l4util\_elf, 328

si  
    l4\_vcpu\_regs\_t, 713

Sigma0 API, 363

signal  
    L4Re::Parent, 800

Signals, 388

size  
    cxx::List, 749  
    L4::Ipc::Buf\_cp\_out, 487  
    L4::Kip::Mem\_desc, 767  
    L4Re::Dataspace, 522  
    L4Re::Video::Color\_component, 505

size0  
    l4\_tracebuffer\_status\_t, 707

size1  
    l4\_tracebuffer\_status\_t, 707

skip  
    L4::Ipc::Istream, 664

slab\_size  
    cxx::Base\_slab, 448  
    cxx::Base\_slab\_static, 452

slice  
    api\_calls\_rt\_sched, 122, 123

Small C++ Template Lib, 44

Smart\_cap  
    L4::Smart\_cap, 849

snd\_base  
    L4::Cap\_base, 502

Split\_ds  
    L4Re::Rm, 816

stack  
    L4Re::Video::View, 903

start  
    L4::Kip::Mem\_desc, 766

State  
    L4vcpu::State, 855

state

        L4vcpu::Vcpu, 894, 895

stats\_time  
    L4::Thread, 875

Strong  
    L4Re::Namespace, 781

sub\_type  
    L4::Kip::Mem\_desc, 768

Super\_pages  
    L4Re::Mem\_alloc, 763

supports  
    L4::Meta, 773

switch\_log  
    L4::Debugger, 534

switch\_to  
    L4::Thread, 874

symlink  
    L4Re::Vfs::Directory, 543

tag  
    L4::Ipc::Istream, 665  
    L4::Ipc::Ostream, 791

take  
    L4Re::Util::Dataspace\_svr, 526

Task, 219

task  
    L4Re::Env, 564  
    L4vcpu::Vcpu, 896

Thread, 226

Thread control, 239

Thread Control Registers (TCRs), 266

Timeouts, 142

Timestamp Counter, 286

total\_objects  
    cxx::Base\_slab, 449  
    cxx::Base\_slab\_static, 453

tracebuffer0  
    l4\_tracebuffer\_status\_t, 707

tracebuffer1  
    l4\_tracebuffer\_status\_t, 707

trigger  
    L4::Irq, 658

type  
    L4::Kip::Mem\_desc, 768  
    l4\_vhw\_entry, 721  
    L4Re::Vfs::Be\_file\_system, 461  
    L4Re::Vfs::File\_system, 604

unbind  
    L4::Icu, 626

unlink  
    L4Re::Vfs::Directory, 542

unlock\_all\_locks  
    L4Re::Vfs::Generic\_file, 610

unmap

L4::Task, [864](#)  
unmap\_batch  
    L4::Task, [865](#)  
unmask  
    L4::Icu, [628](#)  
    L4::Irq, [658](#)  
utcbs\_area  
    L4Re::Env, [565](#), [568](#)  
Utility Functions, [340](#)  
ux\_host\_syscall  
    L4::Thread::Attr, [413](#)

V\_flags  
    L4Re::Video::View, [902](#)

valid  
    cxx::Avl\_set::Node, [784](#)

validate  
    L4::Cap\_base, [503](#)

vCPU API, [275](#)

vCPU Support Library, [369](#)

vcpu\_control  
    L4::Thread, [876](#)

vcpu\_control\_ext  
    L4::Thread, [877](#)

vcpu\_resume\_commit  
    L4::Thread, [876](#)

vcpu\_resume\_start  
    L4::Thread, [875](#)

version  
    l4\_vhw\_descriptor, [719](#)

version0  
    l4\_tracebuffer\_status\_t, [707](#)

version1  
    l4\_tracebuffer\_status\_t, [707](#)

Video API, [75](#)

view  
    L4Re::Video::Goos, [617](#)

view\_info  
    L4Re::Util::Video::Goos\_svr, [621](#)

Virtual Console, [269](#)

Virtual Machines, [174](#)

Virtual Registers (UTCBs), [261](#)

VM API for SVM, [149](#)

VM API for TZ, [280](#)

VM API for VMX, [150](#)

wait  
    L4::Ipc::Istream, [665](#), [666](#)  
    L4::Irq, [657](#)

words  
    cxx::Bitmap\_base, [466](#)

write  
    L4::Vcon, [886](#)

writerv