

SyoSil ApS UVM Scoreboard

1.0.3.0

Generated by Doxygen 1.8.14

Contents

1	Main Page	1
2	Getting started	3
3	How to integrate the UVM scoreboard	5
3.1	Compiling the UVM scoreboard	5
3.2	Accessing the UVM scoreboard from your own code	5
3.3	Factory overrides	6
3.4	Instantiating the UVM scoreboard	7
3.5	Configuring the UVM scoreboard	7
3.5.1	Full build phase	8
3.6	Add sequence items to the scoreboard	8
3.6.1	Function based API hook up	8
3.6.2	TLM based API hook up	9
3.7	Multiple SCB instances & filter transforms	10
3.7.1	Filter transforms	10
4	General implementation notes	13
4.1	General structure	13
4.2	Class diagram	13
4.3	General error handling	14
4.3.1	Error categories	14
4.4	Multiple queue references	15
4.5	Valid queue/compare type combinations	15

5	Queue implementation notes	17
6	Compare implementation notes	19
6.1	Available comparison algorithms	19
6.2	Implementing custom compare algorithms	21
7	Debugging features	23
7.1	Miscompare tables	23
7.2	Scoreboard dump	24
7.3	Orphan dump	25
7.4	XML printer	25
8	API Descriptions	27
8.1	Configuration API	27
8.2	Compare API	28
8.3	Queue API	28
8.4	Hash API	29
8.5	Item API	29
8.6	Iterator API	30
8.7	Locator API	30
8.8	Scoreboard API	31
8.9	Scoreboard Wrapper API	31
8.10	Compare Strategy API	32
9	Overview of included tests	33
10	Overview of configuration knobs	35
10.1	Included configuration knobs	35
10.2	Issues not resolved with config knobs	38
11	Hierarchical Index	39
11.1	Class Hierarchy	39

12 Class Index	43
12.1 Class List	43
13 Class Documentation	49
13.1 cl_scb_test_base Class Reference	49
13.1.1 Detailed Description	49
13.2 cl_scb_test_benchmark Class Reference	49
13.2.1 Detailed Description	50
13.3 cl_scb_test_cmp_base< ATYPE, suffix > Class Template Reference	50
13.3.1 Detailed Description	50
13.4 cl_scb_test_cmp_io< ATYPE, suffix > Class Template Reference	51
13.4.1 Detailed Description	52
13.5 cl_scb_test_cmp_ooo< ATYPE, suffix > Class Template Reference	52
13.5.1 Detailed Description	53
13.6 cl_scb_test_copy_cfg Class Reference	54
13.6.1 Detailed Description	55
13.7 cl_scb_test_double_scb Class Reference	55
13.7.1 Detailed Description	56
13.8 cl_scb_test_io_2hp_md5_simple Class Reference	56
13.8.1 Detailed Description	57
13.9 cl_scb_test_io_2hp_std_sbs_print Class Reference	57
13.9.1 Detailed Description	58
13.10 cl_scb_test_io_2hp_std_simple Class Reference	58
13.10.1 Detailed Description	59
13.11 cl_scb_test_io_md5_disable_compare Class Reference	59
13.11.1 Detailed Description	59
13.12 cl_scb_test_io_md5_dump_orphans Class Reference	59
13.12.1 Detailed Description	60
13.13 cl_scb_test_io_md5_simple Class Reference	60
13.13.1 Detailed Description	61
13.14 cl_scb_test_io_std_comparer_printer Class Reference	61

13.14.1 Detailed Description	61
13.15cl_scb_test_io_std_comparer_report Class Reference	61
13.15.1 Detailed Description	61
13.16cl_scb_test_io_std_disable_compare Class Reference	62
13.16.1 Detailed Description	62
13.17cl_scb_test_io_std_dump Class Reference	62
13.17.1 Detailed Description	63
13.18cl_scb_test_io_std_dump_default Class Reference	63
13.18.1 Detailed Description	64
13.19cl_scb_test_io_std_dump_max_size Class Reference	64
13.19.1 Detailed Description	64
13.20cl_scb_test_io_std_dump_max_size_less Class Reference	64
13.20.1 Detailed Description	64
13.21cl_scb_test_io_std_dump_mixed Class Reference	65
13.21.1 Detailed Description	65
13.22cl_scb_test_io_std_dump_simple Class Reference	66
13.22.1 Detailed Description	66
13.23cl_scb_test_io_std_dump_xml_join Class Reference	67
13.23.1 Detailed Description	67
13.24cl_scb_test_io_std_dump_xml_split Class Reference	68
13.24.1 Detailed Description	68
13.25cl_scb_test_io_std_insert_item Class Reference	69
13.25.1 Detailed Description	69
13.26cl_scb_test_io_std_insert_item_md5 Class Reference	69
13.26.1 Detailed Description	70
13.27cl_scb_test_io_std_intermediate_dump Class Reference	70
13.27.1 Detailed Description	71
13.28cl_scb_test_io_std_sbs_print Class Reference	71
13.28.1 Detailed Description	72
13.29cl_scb_test_io_std_simple Class Reference	72

13.29.1 Detailed Description	72
13.30cl_scb_test_io_std_simple_mutexed Class Reference	72
13.30.1 Detailed Description	73
13.31cl_scb_test_io_std_simple_real Class Reference	73
13.31.1 Detailed Description	73
13.32cl_scb_test_io_std_tlm_gp_test Class Reference	73
13.32.1 Detailed Description	73
13.33cl_scb_test_io_std_tlm_mutexed Class Reference	74
13.33.1 Detailed Description	74
13.34cl_scb_test_iop_md5_simple Class Reference	74
13.34.1 Detailed Description	74
13.35cl_scb_test_iop_std_msw Class Reference	74
13.35.1 Detailed Description	74
13.36cl_scb_test_iop_std_sbs_print Class Reference	75
13.36.1 Detailed Description	75
13.37cl_scb_test_iterator_correctness Class Reference	76
13.37.1 Detailed Description	76
13.38cl_scb_test_iterator_unit_tests Class Reference	76
13.38.1 Detailed Description	77
13.38.2 Member Function Documentation	77
13.38.2.1 check_first()	77
13.38.2.2 check_last()	78
13.38.2.3 check_names()	78
13.38.2.4 check_next()	78
13.38.2.5 check_prev()	79
13.38.2.6 check_set_queue()	79
13.39cl_scb_test_iterator_unit_tests_md5 Class Reference	80
13.39.1 Detailed Description	80
13.40cl_scb_test_md5 Class Reference	81
13.40.1 Detailed Description	81

13.41cl_scb_test_md5_hash_collisions Class Reference	81
13.41.1 Detailed Description	81
13.42cl_scb_test_ooo_heavy_base Class Reference	81
13.42.1 Detailed Description	82
13.43cl_scb_test_ooo_io_md5_simple Class Reference	82
13.43.1 Detailed Description	83
13.44cl_scb_test_ooo_io_std_simple Class Reference	83
13.44.1 Detailed Description	84
13.45cl_scb_test_ooo_md5_duplets Class Reference	84
13.45.1 Detailed Description	84
13.46cl_scb_test_ooo_md5_gp Class Reference	84
13.46.1 Detailed Description	84
13.47cl_scb_test_ooo_md5_heavy Class Reference	85
13.47.1 Detailed Description	85
13.48cl_scb_test_ooo_md5_simple Class Reference	85
13.48.1 Detailed Description	86
13.49cl_scb_test_ooo_md5_tlm Class Reference	86
13.49.1 Detailed Description	86
13.50cl_scb_test_ooo_md5_validate Class Reference	86
13.50.1 Detailed Description	86
13.51cl_scb_test_ooo_std_dump_orphans Class Reference	87
13.51.1 Detailed Description	87
13.52cl_scb_test_ooo_std_dump_orphans_abort Class Reference	87
13.52.1 Detailed Description	87
13.53cl_scb_test_ooo_std_dump_orphans_xml Class Reference	88
13.53.1 Detailed Description	88
13.54cl_scb_test_ooo_std_gp Class Reference	89
13.54.1 Detailed Description	89
13.55cl_scb_test_ooo_std_heavy Class Reference	89
13.55.1 Detailed Description	90

13.56cl_scb_test_ooo_std_max_search_window Class Reference	90
13.56.1 Detailed Description	90
13.57cl_scb_test_ooo_std_primary_multiple Class Reference	90
13.57.1 Detailed Description	90
13.58cl_scb_test_ooo_std_simple Class Reference	91
13.58.1 Detailed Description	91
13.59cl_scb_test_ooo_std_tlm Class Reference	91
13.59.1 Detailed Description	91
13.60cl_scb_test_ooo_std_tlm_filter_trfm Class Reference	92
13.60.1 Detailed Description	92
13.61cl_scb_test_ooo_std_trigger_greed Class Reference	92
13.61.1 Detailed Description	93
13.62cl_scb_test_queue_find_vs_search Class Reference	93
13.62.1 Detailed Description	93
13.63cl_scb_test_rnd Class Reference	93
13.63.1 Detailed Description	94
13.64cl_scb_test_uvm_xml_printer Class Reference	94
13.64.1 Detailed Description	95
13.65cl_scb_test_uvm_xml_printer_break Class Reference	95
13.65.1 Detailed Description	96
13.66cl_scbs_test_base< FIN, MON, FT > Class Template Reference	96
13.66.1 Detailed Description	97
13.67cl_scbs_test_filter_trfm_param Class Reference	97
13.67.1 Detailed Description	97
13.68cl_scbs_test_io_custom_filter_trfm Class Reference	98
13.68.1 Detailed Description	98
13.69cl_scbs_test_io_std_base Class Reference	98
13.69.1 Detailed Description	99
13.70cl_scbs_test_io_std_cc Class Reference	100
13.70.1 Detailed Description	101

13.71cl_scbs_test_ooo_std_base Class Reference	101
13.71.1 Detailed Description	102
13.72cl_scbs_test_ooo_std_flush Class Reference	102
13.72.1 Detailed Description	103
13.73cl_syoscb Class Reference	104
13.73.1 Detailed Description	106
13.73.2 Member Function Documentation	106
13.73.2.1 add_item()	106
13.73.2.2 add_item_mutexed()	107
13.73.2.3 build_phase()	107
13.73.2.4 check_phase()	108
13.73.2.5 compare_control()	108
13.73.2.6 config_validation()	108
13.73.2.7 create_queues_stats()	109
13.73.2.8 create_report()	109
13.73.2.9 create_report_contents()	110
13.73.2.10create_total_stats()	110
13.73.2.11dump_join_txt()	111
13.73.2.12dump_join_xml()	111
13.73.2.13dump_split_txt()	111
13.73.2.14dump_split_xml()	112
13.73.2.15dump_txt()	112
13.73.2.16dump_xml()	112
13.73.2.17empty_queues()	112
13.73.2.18end_of_elaboration_phase()	113
13.73.2.19flush_queues()	113
13.73.2.20get_failed_checks()	114
13.73.2.21get_queue_failed_checks()	114
13.73.2.22get_subscriber()	114
13.73.2.23insert_queues()	115

13.73.2.24	<code>intermediate_queue_stat_dump()</code>	115
13.73.2.25	<code>override_queue_type()</code>	116
13.73.2.26	<code>pre_abort()</code>	116
13.73.2.27	<code>print_header()</code>	116
13.74	<code>cl_syoscb_cfg</code> Class Reference	117
13.74.1	Detailed Description	123
13.74.2	Member Function Documentation	123
13.74.2.1	<code>dynamic_primary_queue()</code>	123
13.74.2.2	<code>exist_producer()</code>	123
13.74.2.3	<code>exist_queue()</code>	124
13.74.2.4	<code>get_comparer()</code>	124
13.74.2.5	<code>get_enable_comparer_report()</code>	125
13.74.2.6	<code>get_enable_queue_stats()</code>	126
13.74.2.7	<code>get_full_scb_max_queue_size()</code>	126
13.74.2.8	<code>get_max_queue_size()</code>	126
13.74.2.9	<code>get_max_search_window()</code>	127
13.74.2.10	<code>get_primary_queue()</code>	127
13.74.2.11	<code>get_printer()</code>	128
13.74.2.12	<code>get_printer_verbosity()</code>	128
13.74.2.13	<code>get_producer()</code>	129
13.74.2.14	<code>get_producers()</code>	129
13.74.2.15	<code>get_queue()</code>	129
13.74.2.16	<code>get_queue_stat_interval()</code>	130
13.74.2.17	<code>get_queues()</code>	130
13.74.2.18	<code>nit()</code>	131
13.74.2.19	<code>set_comparer()</code>	131
13.74.2.20	<code>set_default_enable_comparer_report()</code>	132
13.74.2.21	<code>set_default_printer_verbosity()</code>	132
13.74.2.22	<code>set_dump_orphans_to_files()</code>	132
13.74.2.23	<code>set_enable_comparer_report()</code>	133

13.74.2.24	set_enable_queue_stats()	133
13.74.2.25	set_full_scb_dump_split()	133
13.74.2.26	set_full_scb_max_queue_size()	134
13.74.2.27	set_max_queue_size()	134
13.74.2.28	set_max_search_window()	135
13.74.2.29	set_primary_queue()	135
13.74.2.30	set_printer()	136
13.74.2.31	set_printer_verbosity()	136
13.74.2.32	set_producer()	137
13.74.2.33	set_queue()	137
13.74.2.34	set_queue_stat_interval()	138
13.74.2.35	set_queues()	138
13.74.2.36	set_scb_stat_interval()	139
13.74.2.37	size_queues()	139
13.74.3	Member Data Documentation	139
13.74.3.1	comparers	139
13.74.3.2	default_comparer	140
13.74.3.3	default_enable_comparer_report	140
13.74.3.4	default_printer	140
13.74.3.5	default_printer_verbosity	141
13.74.3.6	disable_clone	141
13.74.3.7	disable_compare_after_error	141
13.74.3.8	disable_report	142
13.74.3.9	dump_orphans_to_files	142
13.74.3.10	enable_c2s_full_scb_dump	142
13.74.3.11	enable_comparer_report	143
13.74.3.12	enable_no_insert_check	143
13.74.3.13	enable_queue_stats	143
13.74.3.14	end_greediness	144
13.74.3.15	full_scb_dump	144

13.74.3.16full_scb_dump_split	144
13.74.3.17full_scb_dump_type	145
13.74.3.18full_scb_max_queue_size	145
13.74.3.19hash_compare_check	145
13.74.3.20max_print_orphans	146
13.74.3.21max_queue_size	146
13.74.3.22max_search_window	146
13.74.3.23mutexed_add_item_enable	147
13.74.3.24ordered_next	147
13.74.3.25orphan_dump_type	147
13.74.3.26primary_queue	148
13.74.3.27print_cfg	148
13.74.3.28print_orphans_as_errors	148
13.74.3.29printer_verbosity	149
13.74.3.30printers	149
13.74.3.31producers	149
13.74.3.32queue_stat_interval	150
13.74.3.33scb_stat_interval	150
13.74.3.34trigger_greediness	150
13.75cl_syoscb_cfg_pl Class Reference	151
13.75.1 Detailed Description	151
13.75.2 Member Function Documentation	151
13.75.2.1 exists()	151
13.76cl_syoscb_compare Class Reference	152
13.76.1 Detailed Description	152
13.76.2 Member Function Documentation	153
13.76.2.1 compare_control()	153
13.76.2.2 compare_trigger()	153
13.76.2.3 extract_phase()	153
13.77cl_syoscb_compare_base Class Reference	154

13.77.1 Detailed Description	157
13.77.2 Member Function Documentation	157
13.77.2.1 check_queues()	157
13.77.2.2 compare_control()	157
13.77.2.3 compare_do_greed()	158
13.77.2.4 compare_init()	158
13.77.2.5 compare_main()	159
13.77.2.6 compare_trigger()	159
13.77.2.7 count_producers()	160
13.77.2.8 delete()	160
13.77.2.9 dynamic_queue_split_do()	160
13.77.2.10 generate_miscmp_table()	161
13.77.2.11 get_count_producer()	161
13.77.2.12 get_primary_queue_name()	162
13.77.2.13 get_queues_item_cnt()	162
13.77.2.14 primary_loop_do()	162
13.77.2.15 primary_loop_init()	163
13.77.2.16 secondary_loop_do()	163
13.77.2.17 set_cfg()	163
13.77.2.18 split_queues()	164
13.77.2.19 static_queue_split_do()	164
13.77.3 Member Data Documentation	164
13.77.3.1 do_split	164
13.77.3.2 secondary_item_found	165
13.78 cl_syoscb_compare_io Class Reference	165
13.78.1 Detailed Description	166
13.78.2 Member Function Documentation	166
13.78.2.1 count_producers() [1/2]	166
13.78.2.2 count_producers() [2/2]	167
13.78.2.3 primary_loop_do() [1/2]	167

13.78.2.4 primary_loop_do() [2/2]	168
13.78.2.5 secondary_loop_do() [1/2]	168
13.78.2.6 secondary_loop_do() [2/2]	168
13.79cl_syoscb_compare_io_2hp Class Reference	169
13.79.1 Detailed Description	170
13.79.2 Member Function Documentation	170
13.79.2.1 compare_do()	170
13.79.2.2 primary_loop_do() [1/2]	171
13.79.2.3 primary_loop_do() [2/2]	171
13.80cl_syoscb_compare_iop Class Reference	172
13.80.1 Detailed Description	173
13.80.2 Member Function Documentation	173
13.80.2.1 compare_init() [1/2]	173
13.80.2.2 compare_init() [2/2]	174
13.80.2.3 get_count_producer() [1/2]	174
13.80.2.4 get_count_producer() [2/2]	174
13.80.2.5 primary_loop_do() [1/2]	175
13.80.2.6 primary_loop_do() [2/2]	175
13.80.2.7 secondary_loop_do() [1/2]	176
13.80.2.8 secondary_loop_do() [2/2]	176
13.81cl_syoscb_compare_ooo Class Reference	177
13.81.1 Detailed Description	178
13.81.2 Member Function Documentation	178
13.81.2.1 get_count_producer() [1/2]	178
13.81.2.2 get_count_producer() [2/2]	178
13.81.2.3 primary_loop_do() [1/2]	179
13.81.2.4 primary_loop_do() [2/2]	179
13.81.2.5 secondary_loop_do() [1/2]	179
13.81.2.6 secondary_loop_do() [2/2]	180
13.82cl_syoscb_comparer_config Class Reference	180

13.82.1 Detailed Description	180
13.82.2 Member Function Documentation	181
13.82.2.1 copy_comparer()	181
13.82.2.2 do_help_pack()	181
13.82.2.3 do_help_unpack()	181
13.82.2.4 get_miscompares_from_comparer()	182
13.82.2.5 get_show_max()	182
13.82.2.6 get_verbosity()	183
13.82.2.7 set_show_max()	183
13.82.2.8 set_verbosity()	183
13.83cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH > Class Template Reference	184
13.83.1 Detailed Description	185
13.83.2 Member Function Documentation	185
13.83.2.1 delete()	186
13.83.2.2 exists()	186
13.83.2.3 first()	186
13.83.2.4 get_hash_item()	187
13.83.2.5 get_item()	187
13.83.2.6 get_size()	188
13.83.2.7 insert()	189
13.83.2.8 last()	189
13.83.2.9 next()	189
13.83.2.10prev()	190
13.83.2.11size()	190
13.84cl_syoscb_hash_base< HASH_DIGEST_WIDTH > Class Template Reference	191
13.84.1 Detailed Description	193
13.84.2 Member Function Documentation	193
13.84.2.1 do_hash()	193
13.84.2.2 hash()	193
13.84.2.3 hash_str()	194

13.84.3 Member Data Documentation	194
13.84.3.1 packer	194
13.85cl_syoscb_hash_item Class Reference	195
13.85.1 Detailed Description	195
13.85.2 Member Function Documentation	196
13.85.2.1 add_item()	196
13.85.2.2 delete_item()	197
13.85.2.3 get_item()	197
13.86cl_syoscb_hash_md5 Class Reference	198
13.86.1 Detailed Description	199
13.86.2 Member Function Documentation	199
13.86.2.1 do_hash() [1/2]	199
13.86.2.2 do_hash() [2/2]	200
13.87cl_syoscb_hash_packer Class Reference	200
13.87.1 Detailed Description	201
13.88cl_syoscb_item Class Reference	201
13.88.1 Detailed Description	202
13.88.2 Member Function Documentation	202
13.88.2.1 convert2string()	203
13.88.2.2 set_producer()	203
13.88.3 Member Data Documentation	203
13.88.3.1 queue_index	203
13.89cl_syoscb_md5_packer Class Reference	204
13.89.1 Detailed Description	204
13.90cl_syoscb_printer_config Class Reference	205
13.90.1 Detailed Description	206
13.90.2 Member Function Documentation	206
13.90.2.1 copy_printer()	206
13.90.2.2 do_help_pack()	206
13.90.2.3 do_help_unpack()	207

13.90.2.4 <code>get_file_descriptor()</code>	207
13.90.2.5 <code>get_printer_of_type()</code>	208
13.90.2.6 <code>get_printer_type()</code>	208
13.90.2.7 <code>set_file_descriptor()</code>	209
13.90.2.8 <code>set_printer_begin_elements()</code>	209
13.90.2.9 <code>set_printer_end_elements()</code>	209
13.91 <code>cl_syoscb_proxy_item_base</code> Class Reference	210
13.91.1 Detailed Description	211
13.91.2 Member Function Documentation	211
13.91.2.1 <code>get_item()</code>	211
13.91.2.2 <code>get_queue()</code>	211
13.91.2.3 <code>set_queue()</code>	211
13.92 <code>cl_syoscb_proxy_item_hash< HASH_DIGEST_WIDTH ></code> Class Template Reference	212
13.92.1 Detailed Description	213
13.92.2 Member Data Documentation	213
13.92.2.1 <code>idx</code>	213
13.93 <code>cl_syoscb_proxy_item_std</code> Class Reference	214
13.93.1 Detailed Description	214
13.94 <code>cl_syoscb_queue_base</code> Class Reference	215
13.94.1 Detailed Description	218
13.94.2 Member Function Documentation	218
13.94.2.1 <code>add_item()</code>	218
13.94.2.2 <code>check_phase()</code>	219
13.94.2.3 <code>create_iterator()</code>	219
13.94.2.4 <code>create_producer_stats()</code>	220
13.94.2.5 <code>create_queue_report()</code>	220
13.94.2.6 <code>decr_cnt_producer()</code>	221
13.94.2.7 <code>delete_item()</code>	221
13.94.2.8 <code>delete_iterator()</code>	222
13.94.2.9 <code>dump()</code>	223

13.94.2.10	<code>dump_orphans_to_file()</code>	223
13.94.2.11	<code>dump_orphans_to_stdout()</code>	224
13.94.2.12	<code>empty()</code>	224
13.94.2.13	<code>exists_cnt_producer()</code>	224
13.94.2.14	<code>flush_queue()</code>	225
13.94.2.15	<code>get_cnt_producer()</code>	225
13.94.2.16	<code>get_dump_extension()</code>	226
13.94.2.17	<code>get_failed_checks()</code>	226
13.94.2.18	<code>get_item()</code>	226
13.94.2.19	<code>get_iterator()</code>	227
13.94.2.20	<code>get_locator()</code>	228
13.94.2.21	<code>get_size()</code>	228
13.94.2.22	<code>incr_cnt_producer()</code>	228
13.94.2.23	<code>insert_item()</code>	229
13.94.2.24	<code>post_add_item()</code>	229
13.94.2.25	<code>pre_add_item()</code>	230
13.94.2.26	<code>print_orphan_xml_footer()</code>	230
13.94.2.27	<code>print_orphan_xml_header()</code>	231
13.94.3	Member Data Documentation	231
13.94.3.1	<code>failed_checks</code>	231
13.95	<code>cl_syoscb_queue_hash< HASH_DIGEST_WIDTH ></code> Class Template Reference	232
13.95.1	Detailed Description	234
13.95.2	Member Function Documentation	234
13.95.2.1	<code>add_item()</code>	235
13.95.2.2	<code>delete_item()</code>	235
13.95.2.3	<code>delete_iterator()</code>	236
13.95.2.4	<code>empty()</code>	236
13.95.2.5	<code>get_item()</code>	237
13.95.2.6	<code>get_key_queue()</code>	237
13.95.2.7	<code>get_size()</code>	238

13.95.2.8 insert_item()	238
13.95.3 Member Data Documentation	239
13.95.3.1 key_queue	239
13.96cl_syoscb_queue_hash_md5 Class Reference	239
13.96.1 Detailed Description	240
13.96.2 Member Function Documentation	240
13.96.2.1 create_iterator()	240
13.96.2.2 get_locator()	241
13.97cl_syoscb_queue_iterator_base Class Reference	241
13.97.1 Detailed Description	243
13.97.2 Member Function Documentation	243
13.97.2.1 first()	243
13.97.2.2 get_item_proxy()	244
13.97.2.3 get_queue()	244
13.97.2.4 has_next()	244
13.97.2.5 has_previous()	245
13.97.2.6 last()	245
13.97.2.7 next()	245
13.97.2.8 next_index()	246
13.97.2.9 previous()	246
13.97.2.10previous_index()	246
13.97.2.11set_queue()	247
13.98cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH > Class Template Reference	247
13.98.1 Detailed Description	249
13.98.2 Member Function Documentation	250
13.98.2.1 first()	250
13.98.2.2 get_item_proxy()	250
13.98.2.3 has_next()	251
13.98.2.4 has_previous()	251
13.98.2.5 last()	251

13.98.2.6 next()	252
13.98.2.7 previous()	252
13.98.2.8 set_queue()	252
13.99 cl_syoscb_queue_iterator_hash_md5 Class Reference	253
13.99.1 Detailed Description	254
13.100 l_syoscb_queue_iterator_std Class Reference	254
13.100.1 Detailed Description	255
13.100.2 Member Function Documentation	256
13.100.2.1 first()	256
13.100.2.2 get_item_proxy()	256
13.100.2.3 has_next()	256
13.100.2.4 has_previous()	257
13.100.2.5 last()	257
13.100.2.6 next()	257
13.100.2.7 previous()	258
13.100.2.8 set_queue()	258
13.101 l_syoscb_queue_locator_base Class Reference	258
13.101.1 Detailed Description	259
13.101.2 Member Function Documentation	259
13.101.2.1 search()	260
13.102 l_syoscb_queue_locator_hash< HASH_DIGEST_WIDTH > Class Template Reference	260
13.102.1 Detailed Description	262
13.102.2 Member Function Documentation	263
13.102.2.1 search()	263
13.102.2.2 validate_match()	263
13.102.2.3 validate_no_match()	264
13.103 l_syoscb_queue_locator_hash_md5 Class Reference	264
13.103.1 Detailed Description	265
13.104 l_syoscb_queue_locator_std Class Reference	265
13.104.1 Detailed Description	267

13.104.2	Member Function Documentation	267
13.104.2.1	compare_items()	267
13.104.2.2	search()	267
13.105	l_syoscb_queue_std Class Reference	268
13.105.1	Detailed Description	270
13.105.2	Member Function Documentation	270
13.105.2.1	add_item()	270
13.105.2.2	create_iterator()	271
13.105.2.3	delete_item()	271
13.105.2.4	delete_iterator()	272
13.105.2.5	empty()	272
13.105.2.6	get_item()	272
13.105.2.7	get_locator()	273
13.105.2.8	get_size()	273
13.105.2.9	insert_item()	273
13.106	l_syoscb_string_library Class Reference	274
13.106.1	Detailed Description	275
13.106.2	Member Function Documentation	275
13.106.2.1	generate_cmp_table_body()	275
13.106.2.2	generate_cmp_table_footer()	275
13.106.2.3	generate_cmp_table_header()	276
13.106.2.4	merge_string_arrays()	276
13.106.2.5	pad_str()	277
13.106.2.6	scb_header_str()	277
13.106.2.7	scb_separator_str()	278
13.106.2.8	split_string()	279
13.106.2.9	sprint_item()	279
13.107	l_syoscb_subscriber Class Reference	280
13.107.1	Detailed Description	280
13.107.2	Member Function Documentation	280

13.107.2.1set_mutexed_add_item_enable()	281
13.108.1syoscbs< FIN > Class Template Reference	281
13.108.1Detailed Description	283
13.108.2Member Function Documentation	283
13.108.2.1build_phase() [1/2]	283
13.108.2.2build_phase() [2/2]	283
13.108.2.3get_filter_trfm() [1/2]	283
13.108.2.4get_filter_trfm() [2/2]	284
13.109.1syoscbs_base Class Reference	285
13.109.1Detailed Description	287
13.109.2Member Function Documentation	287
13.109.2.1build_phase()	287
13.109.2.2compare_control_all()	287
13.109.2.3compare_control_by_index()	288
13.109.2.4compare_control_by_name()	288
13.109.2.5connect_filter_and_subscriber()	288
13.109.2.6connect_filters()	289
13.109.2.7create_filter()	289
13.109.2.8create_filters()	290
13.109.2.9create_report()	290
13.109.2.10create_scb_stats()	291
13.109.2.11create_total_stats()	291
13.109.2.12do_print()	292
13.109.2.13flush_queues_by_index()	292
13.109.2.14flush_queues_by_name()	293
13.109.2.15get_filter_trfm_base()	293
13.109.2.16get_scb()	294
13.109.2.17get_scb_failed_checks()	294
13.109.2.18report_phase()	294
13.109.3Member Data Documentation	295

13.109.3.1fts	295
13.110.1syoscbs_cfg Class Reference	295
13.110.1Detailed Description	297
13.110.2Member Function Documentation	297
13.110.2.1get_cfg()	297
13.110.2.2get_queues()	298
13.110.2.3get_scb_end_greediness()	298
13.110.2.4get_scb_index_by_name()	299
13.110.2.5get_scb_names()	299
13.110.2.6get_scb_trigger_greediness()	300
13.110.2.7nit()	300
13.110.2.8s_scb_names_unique()	300
13.110.2.9set_cfg()	301
13.110.2.10set_compare_type()	301
13.110.2.11set_enable_scb_stats()	302
13.110.2.12set_no_scbs()	302
13.110.2.13set_producers()	302
13.110.2.14set_queue_type()	304
13.110.2.15set_queues()	304
13.110.2.16set_scb_end_greediness()	305
13.110.2.17set_scb_names()	305
13.110.2.18set_scb_trigger_greediness()	306
13.110.3Member Data Documentation	306
13.110.3.1disable_report	306
13.110.3.2print_cfg	306
13.111.1l_tb_cmp_a_d_seq_item< TIOBJ > Class Template Reference	307
13.111.1Detailed Description	308
13.112.1l_tb_cmp_a_f_seq_item< T > Class Template Reference	308
13.112.1Detailed Description	309
13.113.1l_tb_cmp_a_m_seq_item< TIOBJ > Class Template Reference	309

13.113. Detailed Description	310
13.114. <code>l_tb_cmp_b_d_seq_item< TIOBJ ></code> Class Template Reference	311
13.114. Detailed Description	312
13.115. <code>l_tb_cmp_b_f_seq_item< TIOBJ ></code> Class Template Reference	312
13.115. Detailed Description	313
13.116. <code>l_tb_cmp_b_m_seq_item< TIOBJ ></code> Class Template Reference	314
13.116. Detailed Description	315
13.117. <code>l_tb_cmp_seq_item_base< TIOBJ, MAX_ARRAY_SIZE ></code> Class Template Reference	315
13.117. Detailed Description	316
13.118. <code>pk_utils_uvm::filter_trfm< IN, OUT ></code> Class Template Reference	316
13.118. Detailed Description	317
13.118. Member Function Documentation	318
13.118.2.1 <code>evaluate()</code>	318
13.118.2.2 <code>transform()</code>	318
13.118.2.3 <code>write()</code>	318
13.119. <code>l_syoscb_queue_hash< HASH_DIGEST_WIDTH >::packed</code> Struct Reference	319
13.119. Detailed Description	319
13.120. <code>l_syoscb_queue_hash< HASH_DIGEST_WIDTH >::tp_item_digest</code> Struct Reference	319
13.120. Detailed Description	320
13.121. <code>pk_syoscb::uvm_xml_printer</code> Class Reference	320
13.121. Detailed Description	320
13.121. Member Function Documentation	320
13.121.2.1 <code>format_syoscb_item()</code>	320
13.122. <code>vm_xml_printer</code> Class Reference	321
13.122. Detailed Description	321
13.122. Member Function Documentation	322
13.122.2.1 <code>format_array()</code>	322
13.122.2.2 <code>format_object()</code>	322
13.122.2.3 <code>format_primitive()</code>	322
13.122.2.4 <code>format_syoscb_item()</code>	323
13.122.2.5 <code>ss_array()</code>	323
13.122.2.6 <code>ss_object()</code>	324
13.122.2.7 <code>ss_primitive()</code>	324

Chapter 1

Main Page

User and implementation documentation for the SyoSil UVM scoreboard.

This document contains all documentation for the SyoSil UVM scoreboard. It includes a high-level description of the scoreboard's features, as well as class and method documentation generated by Doxygen.

It is assumed that the reader is familiar with the scoreboard architecture described in the paper "Versatile UVM Scoreboarding" located in the **docs/papers** directory.

The document is divided into the following sections:

1. [Getting started](#)
2. [How to integrate the UVM scoreboard](#)
3. [General implementation notes](#)
4. [Queue implementation notes](#)
5. [Compare implementation notes](#)
6. [Debugging features](#)
7. [API Descriptions](#)
8. [Overview of included tests](#)
9. [Overview of configuration knobs](#)

Chapter 2

Getting started

This software package provides several examples beside the source code for the UVM scoreboard. Before starting to integrate the UVM scoreboard into your own code, it might be beneficial to look at the provided examples. An example testbench is placed in the **tb** directory and the tests are in the **tb/test** directory.

To run the examples you need to select a Vendor since the examples can be run with all of the three major System↔Verilog simulator vendors: Cadence, Siemens EDA, and Synopsys. See **README.txt** for a description of how to select the vendor.

Once the vendor has been selected the available Make targets for that vendor can be listed by typing: "make". Typically, you run the simulation with **make sim**.

In general you can type **make help** to get information about the Make options that are available.

Chapter 3

How to integrate the UVM scoreboard

The UVM scoreboard is easily integrated into your existing testbench environment. The following steps should be followed to start using the UVM scoreboard:

1. Compile the UVM scoreboard
2. Access the UVM scoreboard from your own code
3. Perform factory overrides
4. Instantiate the UVM scoreboard
5. Configure the UVM scoreboard
6. Add sequence items to the scoreboard
7. Use the scoreboard wrapper for multiple similar scoreboards

3.1 Compiling the UVM scoreboard

To get the UVM scoreboard compiled you need to add [src/pk_syoscb.sv](#) to your list of files that are compiled when compiling your testbench. How this is done is highly dependent on the verification environment since some environments compile everything into different libraries and some do not. Refer to your vendor's manual for further information on how to include packages.

3.2 Accessing the UVM scoreboard from your own code

Once the UVM scoreboard is compiled with the verification environment, it is accessible either by explicit scoping:

```
class myclass;
    pk_syoscb::cl_syoscb my_new_scb;
    ...
endclass
```

or by importing the complete package into your scope:

```
import pk_syoscb::*;

class myclass;
    cl_syoscb my_new_scb;
    ...
endclass
```

3.3 Factory overrides

Before instantiating the scoreboard, the desired queue type and compare algorithm need to be set in the scoreboard's configuration object. This is done by factory overrides since the queue type and compare algorithm can be changed on a per-test basis.

NOTE: This MUST be done before creating the scoreboard!

The queue type and compare algorithm should **not** be overwritten with a call to the UVM configuration database. Instead, the scoreboard configuration object should be used.

The factory overrides are done in the build phase of the `cl_syoscb`, depending on the value of the `cl_syoscb_cfg.queue_type` and `cl_syoscb_cfg.compare_type` configuration knobs. If no overwriting is performed, the test will fail, as the default queue and comparison types are set to `USER_DEFINED`, a placeholder value for user-defined queue types and comparison types.

The scoreboard comes with a number of built-in queue types and comparison algorithms (see [Queue implementation notes](#) and [Compare implementation notes](#)). The following queue implementations are available:

1. Standard SV queue (`cl_syoscb_queue_std`)
2. MD5 queue (`cl_syoscb_queue_hash_md5`)

and the following compare algorithms are available:

1. Out of Order (OOO, `cl_syoscb_compare_ooo`)
2. In Order (IO, `cl_syoscb_compare_io`).
3. In Order with 2 queues, high performance (IO_2HP, `cl_syoscb_compare_io_2hp`)
4. In Order by Producer (IOP, `cl_syoscb_compare_iop`)

Setting the queue topology is done with the method `set_queue_type` in `cl_syoscb_cfg`. For example, the following line shows how to select the MD5 queue topology for a scoreboard.

```
this.syoscb_cfg.set_queue_type(pk_syoscb::SYOSCB_QUEUE_MD5);
```

The following line shows an example of how to change the compare strategy. Here, OOO comparisons are enabled.

```
this.syoscb_cfg.set_compare_type(pk_syoscb::SYOSCB_COMPARE_OOO);
```

All of the enum values used for selecting queue type and compare algorithm can be found in [src/syoscb_common.svh](#).

3.4 Instantiating the UVM scoreboard

The UVM scoreboard itself needs to be instantiated along with the configuration object. The simplest way to do this is to add the UVM scoreboard and the configuration object to the UVM environment – note that the configuration object is passed to the scoreboard via the `uvm_config_db`.

```
import pk_syoscb::*;

class cl_scbtest_env extends uvm_env;

    cl_syoscb      syoscb;
    cl_syoscb_cfg  syoscb_cfg;

    `uvm_component_utils_begin(cl_scbtest_env)
        `uvm_field_object(syoscb, UVM_ALL_ON)
        `uvm_field_object(syoscb_cfg, UVM_ALL_ON)
    `uvm_component_utils_end

    ...

endclass: cl_scbtest_env

function void cl_scbtest_env::build_phase(uvm_phase phase);
    super.build_phase(phase);

    // Create the scoreboard configuration object
    this.syoscb_cfg = cl_syoscb_cfg::type_id::create("syoscb_cfg");

    // Pass the scoreboard configuration object to the config_db
    uvm_config_db #(cl_syoscb_cfg)::set(this, "syoscb", "cfg", this.syoscb_cfg);

    // Create the scoreboard
    this.syoscb = cl_syoscb::type_id::create("syoscb", this);

    ...

endfunction: build_phase
```

3.5 Configuring the UVM scoreboard

A default configuration is not created, so a configuration object must be constructed, configured and set in the UVM configuration database for each scoreboard instance to pick it up. One must create a separate scoreboard configuration object for each scoreboard instance. It cannot be reused! The following example shows a scoreboard with two queues, Q1 and Q2, with Q1 as the primary queue. Furthermore, one producer P1 is added to both queues:

```
function void cl_scbtest_env::build_phase(uvm_phase phase);
    super.build_phase(phase);

    // Create the scoreboard configuration object
    this.syoscb_cfg = cl_syoscb_cfg::type_id::create("syoscb_cfg");

    // Configure the scoreboard
    this.syoscb_cfg.set_queues({"Q1", "Q2"});
    void' (this.syoscb_cfg.set_primary_queue("Q1"));
    void' (this.syoscb_cfg.set_producer("P1", {"Q1", "Q2"}));

    // Pass the scoreboard configuration object to the config_db
    uvm_config_db #(cl_syoscb_cfg)::set(this, "syoscb", "cfg", this.syoscb_cfg);

    // Create the scoreboard
    this.syoscb = cl_syoscb::type_id::create("syoscb", this);

    ...

endfunction: build_phase
```

For more info about the configuration options, see [pConfiguration](#).

3.5.1 Full build phase

The full build phase of our example environment `cl_scbtest_env` is shown here for completeness:

```
function void cl_scbtest_env::build_phase(uvm_phase phase);
    super.build_phase(phase);

    // Use the MD5 queue implementation as scoreboard queue
    this.syoscb_cfg.set_queue_type(pk_syoscb::SYOSCB_QUEUE_MD5);

    // Set the compare strategy to be OOO
    this.syoscb_cfg.set_compare_type(pk_syoscb::SYOSCB_COMPARE_OOO);

    // Create the scoreboard configuration object
    this.syoscb_cfg = cl_syoscb_cfg::type_id::create("syoscb_cfg");

    // Configure the scoreboard
    this.syoscb_cfg.set_queues({"Q1", "Q2"});
    void' (this.syoscb_cfg.set_primary_queue("Q1"));
    void' (this.syoscb_cfg.set_producer("P1", {"Q1", "Q2"}));

    // Pass the scoreboard configuration object to the config_db
    uvm_config_db #(cl_syoscb_cfg)::set(this, "syoscb", "cfg", this.syoscb_cfg);

    // Create the scoreboard
    this.syoscb = cl_syoscb::type_id::create("syoscb", this);

    ...
endfunction: build_phase
```

3.6 Add sequence items to the scoreboard

3.6.1 Function based API hook up

The function based API is very easy to use once you have done the configuration and instantiation of the scoreboard as described above.

Whenever you need to add a UVM sequence item to a queue produced by a specified producer, simply invoke the `cl_syoscb::add_item()` method:

```
// *NOTE*: Assumes syoscb is handle to an instance of the scoreboard and
//          item1 is a handle to a UVM sequence item
...

// Insert UVM sequence item for queue: Q1, for producer: P1
syoscb.add_item("Q1", "P1", item1);
```

Invoking the `cl_syoscb::add_item()` method will wrap the UVM sequence item in a `cl_syoscb_item` object, add it to the correct queue and finally invoke the configured compare algorithm.

The UVM environment will typically contain a handle to the scoreboard as described above. This can then be utilized if UVM sequence item needs to be added from a test case:

```
class cl_scbtest_seq_item extends uvm_sequence_item;
    //-----
    // Randomizable variables
    //-----
    rand int unsigned int_a;

    //-----
    // UVM Macros
    //-----
    `uvm_object_utils_begin(cl_scbtest_seq_item)
        `uvm_field_int(int_a, UVM_ALL_ON)
    `uvm_object_utils_end
```

```

//-----
// Constructor
//-----
function cl_scbtest_seq_item::new (string name = "cl_scbtest_seq_item");
    super.new(name);
endfunction
endclass: cl_scbtest_seq_item

class cl_scbtest_test extends uvm_test;
//-----
// Non randomizable variables
//-----
cl_scbtest_env scbtest_env;

//-----
// UVM Macros
//-----
`uvm_component_utils(cl_scbtest_test)

//-----
// Constructor
//-----
function new(string name = "cl_scbtest_test", uvm_component parent = null);
    super.new(name, parent);
endfunction: new

//-----
// UVM Phase methods
//-----
function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    scbtest_env = cl_scbtest_env::type_id::create("scbtest_env", this);
endfunction: build_phase

task run_phase(uvm_phase phase);
    super.run_phase(phase);
    begin
        cl_scbtest_seq_item item1;
        item1 = cl_scbtest_seq_item::type_id::create("item1");
        item1.int_a = 'h3a;
        scbtest_env.syoscb.add_item("Q1", "P1", item1);
    end
    begin
        cl_scbtest_seq_item item1;
        item1 = cl_scbtest_seq_item::type_id::create("item1");
        item1.int_a = 'h3a;
        scbtest_env.syoscb.add_item("Q2", "P1", item1);
    end
endtask: run_phase
endclass: cl_scbtest_test

```

3.6.2 TLM based API hook up

The TLM API is even easier to use than the function based API. The scoreboard provides a generic UVM subscribers for each producer on each queue. This subscriber can be connected to anything which has a UVM analysis port (e.g. a UVM monitor). Typically, the UVM agents inside the UVM environment contain one or more monitors with UVM analysis ports which should be connected to the scoreboard. The following example shows two agents, each of which has a monitor. The monitors are connected to Q1 and Q2 in the scoreboard, acting as producer P1:

```

import pk_syoscb::*;

class cl_scbtest_env extends uvm_env;

    cl_syoscb      syoscb;
    cl_syoscb_cfg syoscb_cfg;
    myagent        agent1;
    myagent        agent2;

    ...

    function void build_phase(uvm_phase phase);

        ...

        // Configure and create the scoreboard
        // Create and configure the agents

        ...
    endfunction
endclass

```

```

endfunction: build_phase

...

function void connect_phase(uvm_phase phase);
    super.connect_phase(phase);

    begin
        cl_syoscb_subscriber subscriber;

        // Get the subscriber for Producer: P1 for queue: Q1 and connect it
        // to the UVM monitor producing transactions for this queue
        subscriber = this.syoscb.get_subscriber("Q1", "P1");
        this.agent1.mon.<analysis port>.connect(subscriber.analysis_export);

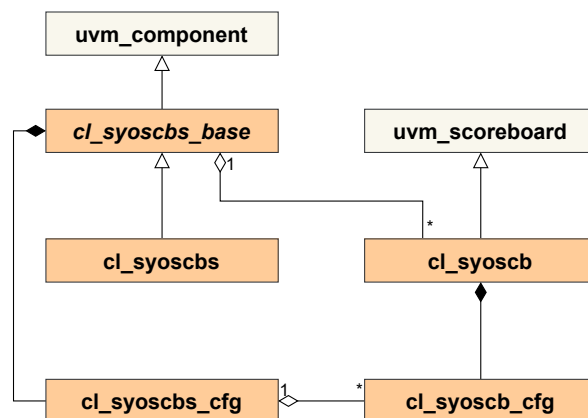
        // Get the subscriber for Producer: P1 for queue: Q2 and connect it
        // to the UVM monitor producing transactions for this queue
        subscriber = this.syoscb.get_subscriber("Q2", "P1");
        this.agent1.mon.<analysis port>.connect(subscriber.analysis_export);
    end
endfunction: connect_phase
endclass: cl_scbtest_env

```

3.7 Multiple SCB instances & filter transforms

The SyoSil UVM scoreboard also comes with a scoreboard wrapper, [cl_syoscb_base](#), which can be used to instantiate several scoreboards with similar configurations.

A configuration wrapper, [cl_syoscb_cfg](#), is used to configure the scoreboard wrapper. The configuration wrapper contains N configuration objects, one for each wrapped scoreboard. The wrapped scoreboards may have different queue / producer names and numbers of queues/producers, or they may be the same. See the figure below for a UML diagram of the relationship between individual scoreboards and their configurations, and the scoreboard wrapper and its configuration object.



3.7.1 Filter transforms

Since UVM analysis ports are parameterized with the types of items they will accept, and the SyoSil scoreboard's [cl_syoscb_subscriber](#) expects input items to be of type `uvm_sequence_item`, a transformation must be used to upcast sequence items to this datatype before they are inserted. When using a single scoreboard, this transformation can easily be instantiated manually, or items can be upcast in a monitor before being written to the attached subscriber.

In the case where 100's or 1000's of scoreboards are used, manually instantiating and connecting all of these transforms can become tedious. Instead, the scoreboard wrapper offers **filter transforms** to automate the process. When creating the scoreboard wrapper [cl_syoscb_base](#), it must be parameterized with the type of sequence items that

will be input. It then automatically instantiates a transformation object for each subscriber, and connects its output to the input of the subscriber. Now, instead of retrieving the subscribers for each queue/producer, a filter transform for each scoreboard's queue/producer combination should be retrieved.

The default filter transform `pk_utils_uvm::filter_trfm`, included in `lib/pk_utils_uvm.sv`, simply upcasts its input to a `uvm_sequence_item` before passing it on to the scoreboard. If more complex transforms are required, you can extend `cl_syoscbs_base` and implement `cl_syoscbs_base::create_filter` to suit your needs.

The class `cl_syoscbs_base` serves as the base class for the scoreboard wrapper, and a default implementation is included in `cl_syoscbs`. The default implementation should be enough for most applications

In the example below, a scoreboard wrapper with 10 scoreboards is created. Each scoreboard has two queues, DUT and REF, each of which has two producers, P1 and P2. In the environment's build phase, the scoreboard wrapper and config object are created. After initializing the configuration object, it is passed to the scoreboard wrapper with the UVM configuration database. In the environment's connect phase, each DUT agent is connected to filter transforms associated with their respective scoreboard. The parameter `FIN` is the input type to the filter transforms. The filter transforms convert this type to a `uvm_sequence_item` which is passed into the scoreboard.

```
import pk_syoscbs::*;

class cl_scbs_env#(type FIN = my_seq_item) extends uvm_env;
  int          NUM_SCB = 10;
  cl_syoscbs#(FIN) syoscbs;
  cl_syoscbs_cfg syoscbs_cfg;
  dut_agent     dut_agents[NUM_SCB];
  ref_agent     ref_agents[NUM_SCB];
  string        producers[] = {"P1", "P2"};
  string        queues[] = {"DUT", "REF"};

  ...

  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    this.syoscbs_cfg = cl_syoscbs_cfg::type_id::create("syoscbs_cfg");
    //Create an scb wrapper named my_syoscbs with 10 scoreboards. They will be named "scb[i]",
    //and each will have queues named "DUT", "REF", and each queue will have producers "P1" and "P2"
    this.syoscbs_cfg.init("syoscbs", NUM_SCB, "scb", queues, producers);

    uvm_config_db #(cl_syoscbs_cfg)::set(this, "", "syoscbs", this.syoscbs_cfg);

    this.syoscbs = cl_syoscbs#(FIN)::type_id::create("syoscbs");

    ... //Create and configure all dut agents and ref agents
  endfunction: build_phase

  function void connect_phase(uvm_phase phase);
    //Each DUT agent is connected to a separate scoreboard
    foreach(dut_agents[i]) begin
      pk_utils_uvm::filter_trfm#(FIN, uvm_sequence_item) filter_trfm_p1;
      pk_utils_uvm::filter_trfm#(FIN, uvm_sequence_item) filter_trfm_p2;

      //Get handles to the filter transforms connected to DUT queue for P1 and P2, scoreboard 'i'
      filter_trfm_p1 = this.syoscbs.get_filter_trfm("DUT", "P1", i);
      filter_trfm_p2 = this.syoscbs.get_filter_trfm("DUT", "P2", i);
      //Connect agents to filter transforms
      dut_agents[i].pl_anls_port.connect(filter_trfm_p1.analysis_export);
      dut_agents[i].p2_anls_port.connect(filter_trfm_p2.analysis_export);
    end
    // ... Perform the same procedure for reference model ports
  endfunction: connect_phase
```

The included testcases also include several tests using the scoreboard wrapper, which can be used as a starting point. See `cl_scbs_test_base` and `cl_tb_env_scbs`. See `tb/test/scbs/cl_scbs_test_base` and `tb/cl_tb_env_scbs`.

Chapter 4

General implementation notes

This chapter contains some background information on the architecture of the scoreboard.

4.1 General structure

Each scoreboard ([cl_syoscb](#)) consists of a number of queues (see [Queue implementation notes](#)). A scoreboard should have as many queues as there are models being tested. If e.g. the testbench compares a DUT to a single reference model, two queues, "DUT" and "REF" should be instantiated. Each queue is associated with a number of producers. If e.g. the inputs and outputs of the DUT/REF are being sampled, these producers may be named "IN" and "OUT".

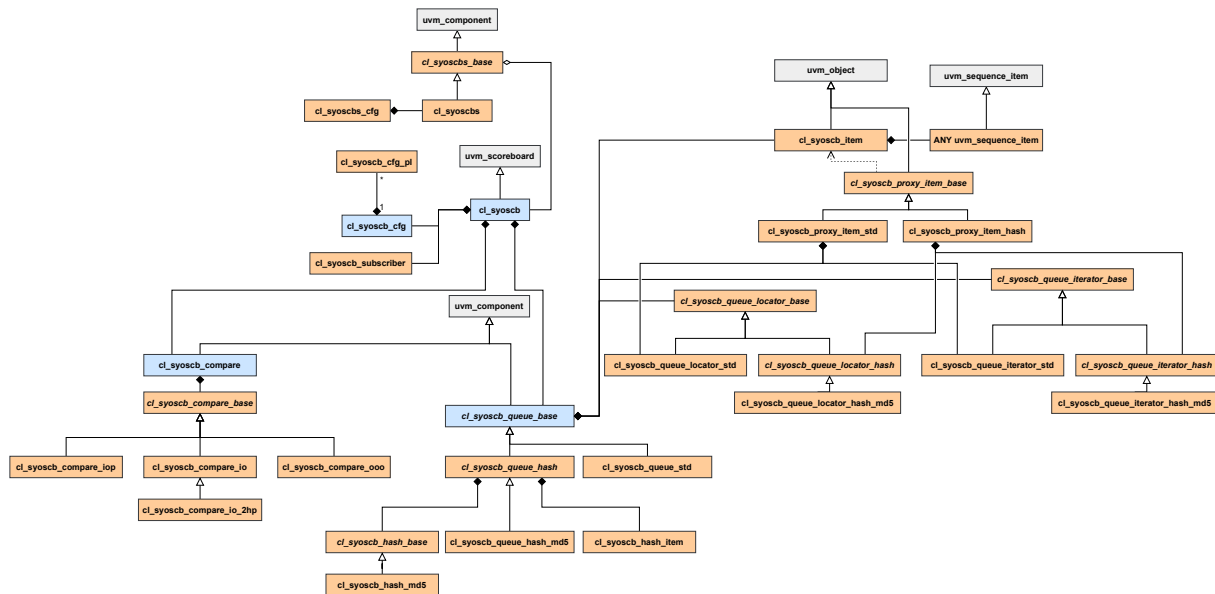
The scoreboard operates by using TLM connections or the function-based API to receive data sampled by a UVM monitor or a VC. The scoreboard has a subscriber for each producer on each queue. When a sequence item is written to a subscriber, the sequence item is wrapped in a [cl_syoscb_item](#) before it is added to the correct queue. The [cl_syoscb_item](#) is a wrapper item which includes some metadata such as which producer generated the item.

The queues in the scoreboard store sequence items until at least one sequence item is present in each queue. When this condition is met, a comparison is started, based on the chosen compare algorithm (see [Compare implementation notes](#)). Every time an item is inserted during simulation, the scoreboard checks if all queues contain an item, and starts a comparison if this is the case. It does so throughout the entire simulation. If any errors occur, the scoreboard will generate a UVM_ERROR. See [Debugging features](#) for more information on the debugging features that the scoreboard includes.

Note that when using the SyoSil UVM Scoreboard, the reference model and scoreboard are completely separated, which is not the case in many other UVM scoreboards. By separating scoreboard and reference model, we achieve separation of concerns, making both the reference model and scoreboard simpler to use and instantiate in other testbenches.

4.2 Class diagram

A UML class diagram of the SyoSCB is shown below. The PDF version of the manual also includes a PDF diagram, which allows you to zoom in further than the PNG file used in the HTML version of the manual.



Classes highlighted with blue make up the core 4 aspects of the scoreboard: configuration, queues, compare strategy and the scoreboard itself. Classes with italicized names are abstract base classes that should not be implemented. Grey classes are from the UVM class hierarchy, and have been included for visualization purposes only. They are not included in the scoreboard's source code.

4.3 General error handling

Whenever a method detects an error, two error handling concepts are used:

Getter methods and other methods with a return value will generally issue a UVM_INFO message at verbosity level UVM_DEBUG with some information about what went wrong, returning 1'b0 or `null` to signal an error. It is up to the calling method to respond to and handle the error, or escalate it to a UVM_ERROR / UVM_FATAL.

Setter methods and other methods without a return value will issue a UVM_ERROR or UVM_FATAL, as they have no other way of signalling errors. Again, it is the responsibility of the caller to potentially catch and handle these errors.

4.3.1 Error categories

There are several categories of errors used throughout the scoreboard. The following table lists some of them along with a possible cause for the error.

Error Category	Description
IMPL_ERROR	Implementation error. Something is really broken
QUEUE_ERROR	A queue related error, e.g. the queue could not be found
CFG_ERROR	Configuration error. Usually because the configuration object is missing
TYPE_ERROR	Type error. Typically issued when <code>\$cast()</code> fails
COMPARE_ERROR	Compare error. Issued, e.g. when the in order compare fails
SUBSCRIBER_ERROR	Subscriber error. Issued, e.g. when the call to <code>cl_syoscb::get_subscriber()</code> fails
ITERATOR_ERROR	Iterator error. Issued when an iterator cannot be found in <code>cl_syoscb_queue_hash</code>

4.4 Multiple queue references

Both the top level scoreboard class [cl_syoscb](#) and the configuration class [cl_syoscb_cfg](#) contain handles to all queues. The former uses an ordinary array which provides a fast way of looping over the queues and the latter an associative which makes it easy to find a queue using only its name. Using either reference is OK, use the approach best suited for a given operation.

4.5 Valid queue/compare type combinations

In the table below, an overview of all queue and compare type combinations is shown, as well as their possible limitations. See [Queue implementation notes](#) and [Compare implementation notes](#) for additional information on queue topologies and comparison strategies.

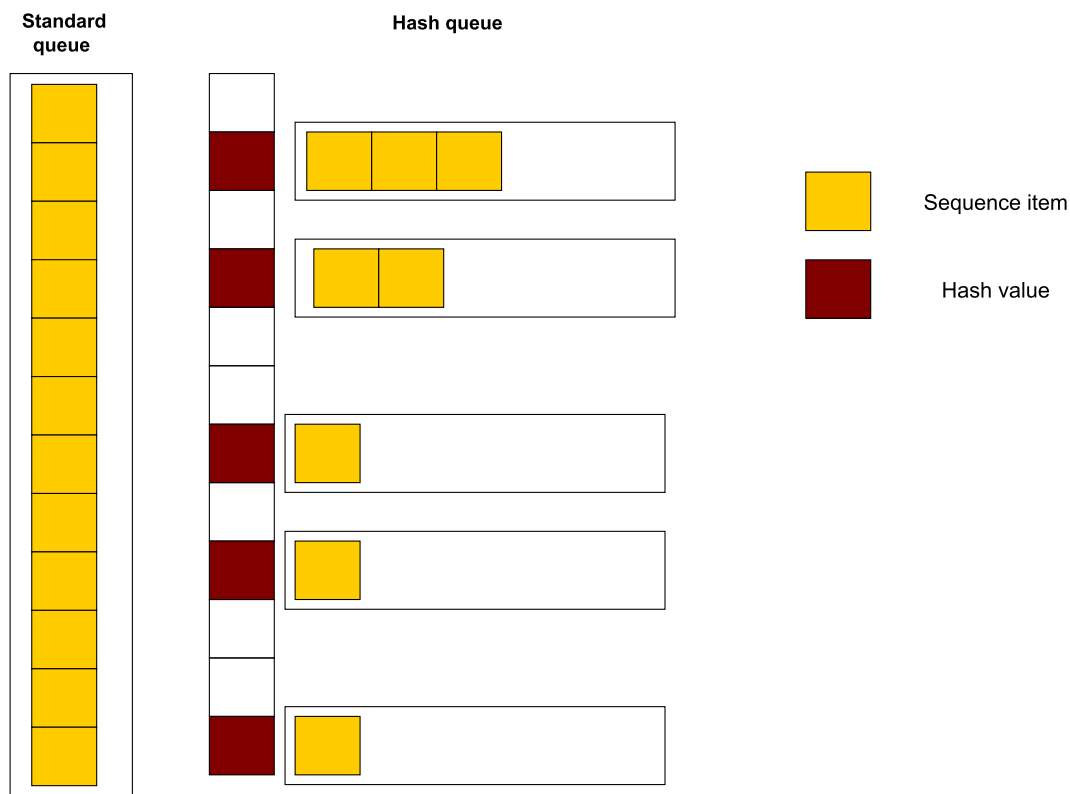
	STD Queue	MD5 Queue
IO Compare	Recommended	Works if cl_syoscb_cfg.ordered_next = 1. Not recommended
IOP Compare	Recommended	Works if cl_syoscb_cfg.ordered_next = 1. Not recommended
IO-2HP Compare	Recommended	Works if cl_syoscb_cfg.ordered_next = 1. Not recommended
OOO Compare	Works, slow on very large datasets (see Queue implementation notes)	Works, recommended approach for large datasets. Incurs a slight performance hit if cl_syoscb_cfg.ordered_next = 1

Chapter 5

Queue implementation notes

The queues are used to store the items that needs to be compared. It is possible to select the maximum number of items you can store before obtaining an error by modifying the value of [cl_syoscb_cfg.max_queue_size](#).

The scoreboard provides two different queue topologies. Standard queues, which are based on SV queues, and hash queues, which use an associative array to store items. In the following figure it is possible to see the format of the two different types of queues.



The hash queue shows that multiple sequence items may be associated with the same hash value. This is a rare occurrence, and should almost never happen. In the cases where it does happen, the scoreboard tracks all items with matching hash values, ensuring that no sequence items are lost in case of a hash collision.

The two types of queues are further described in the table below:

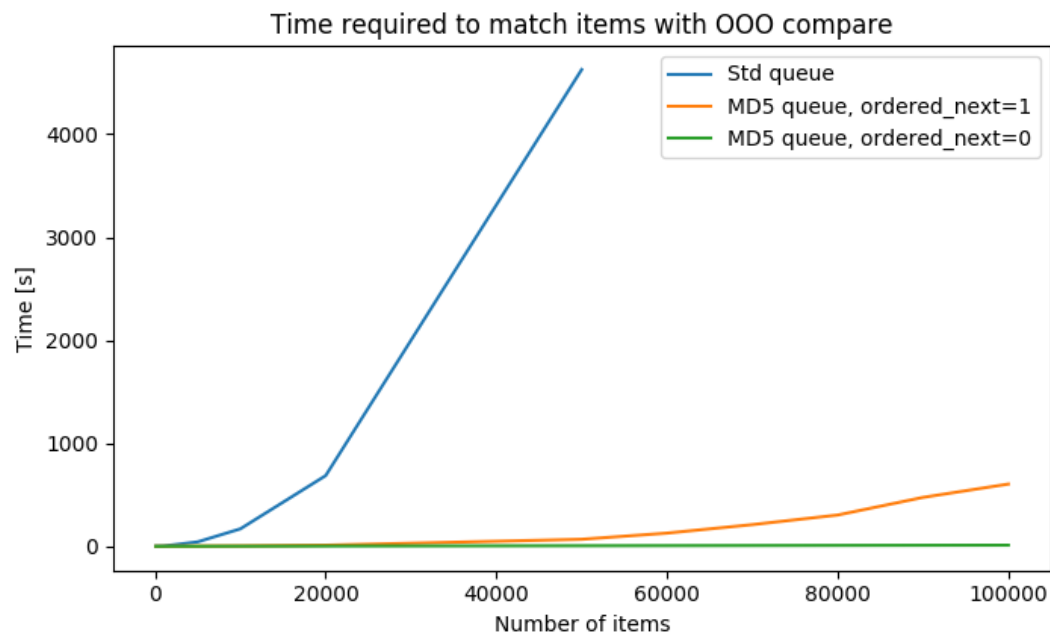
Queue type	Description
cl_syoscb_queue_std	SystemVerilog queue. New items are added at the end of the queue.
cl_syoscb_queue_hash	Associative array. Items are stored as key-value pairs, using the hash values of the packed item (using <code>object.pack()</code>) as the key. The value is a SV queue storing all items with that hash value. The queue is used to handle the very rare cases where two seq. items have the same hash value.

Hash queues do not by themselves preserve insertion order. To do so, such that hash queues can be iterated in the same order as standard queues, the configuration knob [cl_syoscb_cfg.ordered_next](#) must be enabled.

- Configuration knob set to 1: Guarantees the order of insertions by maintaining some metadata. This ensures that when using an iterator and traversing the queue with the iterator's `next` method, items are returned in the same order as they were inserted. The OOO compare with hashed queues take a minor performance hit when this is enabled.
- Configuration knob set to 0: Uses the SystemVerilog implementation of the `next` method for associative arrays. This does not guarantee the iteration order to be the order that items were inserted in. For OOO compares using hash queues, this is the option which makes the OOO compare perform at its maximum.

The advantage of using hash queue can be seen when using OOO comparisons. To check if an element is in a queue or not, all that is required is to check if the hash value is in the associative array. This operation runs in $O(1)$ time. Using the SystemVerilog queue, checking if an item is present runs in $O(n)$ time, where n is the number of items in the queue.

In the following graph, the time needed to finish the comparison using out of order compare is shown for the standard queue vs. the MD5 hash queue. It is obvious that hash queues are much better suited for large queues when using OOO comparisons than standard queues.



Chapter 6

Compare implementation notes

6.1 Available comparison algorithms

The compare procedure consists of finding the same element in all queues in the scoreboard. One queue is selected as the **primary queue**, and an attempt is then made to find the first element from the primary queue in all other queues. Whenever a match is found, these elements are removed from the queues.

The compare mechanism is triggered whenever an element is inserted into a queue, leading to all queues being non-empty (if one or more queues are empty after insertion, there cannot be a match and no comparison is performed). The figure below shows an OOO-compare being performed after an item is inserted into Q3.



The compare can be disabled after the first UVM_ERROR if the [cl_syoscb_cfg.disable_compare_after_error](#) configuration knob is set to 1'b1.

A UVM_ERROR is obtained on a number of occasions:

- When using In Order-based comparisons, a UVM_ERROR is issued if the first item in any of the secondary queues does not match the first item in the primary queue.

- When using Out of Order comparison, a UVM_ERROR is only issued if a queue's size reaches the limit set in `cl_syoscb_cfg.max_queue_size`. Compare errors are not issued, as the OOO comparison, by definition, cannot know whether a matching item may arrive at a later point in time.
- Independently of compare strategy, a UVM_ERROR may be issued if one or more of the queues are non-empty at the end of simulation. This depends on the value of `cl_syoscb_cfg.orphans_as_errors`

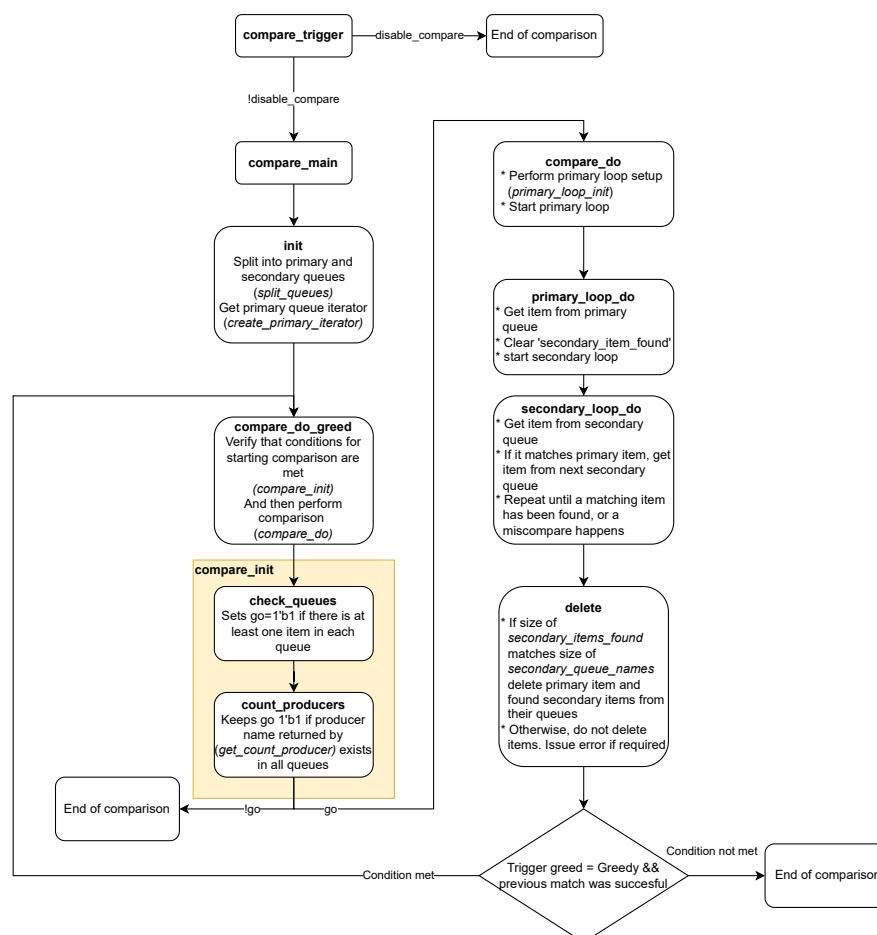
The table below outlines the differences between the available comparison strategies.

Compare method	Description
cl_syoscb_compare_ooo	Elements do not have to be in the same order. Standard queue: In order to find a match it has to loop through all queue items. MD5 queue: In order to find a match it only has to check if the hash value is in the associative array.
cl_syoscb_compare_io	Elements have to be in the same order.
cl_syoscb_compare_io_2hp	Only two queues in scoreboard. Elements have to be in the same order.
cl_syoscb_compare_iop	Elements from the same producer have to be in the same order.

6.2 Implementing custom compare algorithms

If you require a custom comparison behaviour which is not covered by the included compare algorithms, a new algorithm can be implemented by extending [cl_syoscb_compare_base](#) and implementing the necessary methods labeled with **Compare Strategy API**.

Included below is a flowchart of the general compare flow. This can be used as a starting point when implementing new compare algorithms.



Chapter 7

Debugging features

The SyoSil UVM Scoreboard has been designed to make debugging a failed test as simple as possible. This section describes some of the features which help to track down where errors occur.

7.1 Miscompare tables

When using the IO, IOP or IO_2HP comparison strategies, a miscompare table is generated when a comparison fails. The table contains a printout of the items which generated the miscompare, and may also contain a comparer report with specific information on which fields failed the comparison. By default, the comparer report is generated whenever an error occurs. It may be disabled via the [cl_syoscb_cfg.enable_comparer_report](#) knob.

When using Out of Order comparisons or a user defined compare strategy, comparer reports are disabled, as OOO compares naturally lead to a large number of "miscompares" when searching for a match.

An example of a miscompare table and comparer report is shown below: The upper portion of the miscompare table contains the two sequence items next to each other. The lower portion contains miscompare information retrieved from the `uvm_comparer` used to perform the comparison, indicating which fields caused the miscompare.

```

UVM_ERROR ./src/cl_syoscb_compare_io.svh(102) @ 0: reporter [COMPARE_ERROR]
#####
# [syoscb0]: cmp-io: Item from primary queue (Q1) not found in secondary queue (Q2) #
#####
-----
Name                Type                Size  Value                Name                Type                Size  Value
-----
P1-item-3881        cl_syoscb_item -      @3881                P1-item-3982        cl_syoscb_item -      @3982
  producer          string                2      P1                    producer          string                2      P1
  insertion_index    integral              64      'd3                  insertion_index    integral              64      'd3
  queue_index        integral              64      'd0                  queue_index        integral              64      'd0
  item               large_seq_item -      @3922                item               large_seq_item -      @3959
    int_a            integral              32      'hb9cc997a          int_a            integral              32      'hb0cca0e2
    int_b            integral              32      'h0                  int_b            integral              32      'h0
    int_arr          da(integral)          9      -                    int_arr          da(integral)          9      -
      [0]            integral              32      'h325ea203          [0]            integral              32      'hfd0e2295
      [1]            integral              32      'h6bfce43a          [1]            integral              32      'h706ab852
      [2]            integral              32      'h87cde414          [2]            integral              32      'h9cce61cb
      [3]            integral              32      'h2f81a427          [3]            integral              32      'hf3c23513
      [4]            integral              32      'hc6f99f4c          [4]            integral              32      'h69bb5ea7
      [5]            integral              32      'h4cb314e2          [5]            integral              32      'h5acc6b0b
      [6]            integral              32      'h84d81aab          [6]            integral              32      'h4227f3a3
      [7]            integral              32      'h12e31653          [7]            integral              32      'h5c79005
      [8]            integral              32      'h97bfa2c4          [8]            integral              32      'h1665fb34
-----
#####
Results from uvm_comparer::get_miscompares() [show_max=5]
-----
P1-item-3982.item.int_a: lhs = 'hb0cca0e2 : rhs = 'hb9cc997a
P1-item-3982.item.int_arr[0]: lhs = 'hfd0e2295 : rhs = 'h325ea203
P1-item-3982.item.int_arr[1]: lhs = 'h706ab852 : rhs = 'h6bfce43a
P1-item-3982.item.int_arr[2]: lhs = 'h9cce61cb : rhs = 'h87cde414
P1-item-3982.item.int_arr[3]: lhs = 'hf3c23513 : rhs = 'h2f81a427
#####

```

The maximum number of miscompares shown is controlled via the `show_max` knob in the `uvm_comparer` used for the comparison. To modify this value, use `cl_syoscb_comparer_config::set_show_max()`.

`uvm_comparers` are by default set to print miscompare information whenever a miscompare occurs. This is controlled via the `verbosity` knob of a `uvm_comparer`, which can be set with `cl_syoscb_comparer_config::set_verbosity()`. The default verbosity level used by `uvm_comparer`'s is `UVM_LOW`. Since the scoreboard incorporates its own comparer reports, the `uvm_comparer`'s reports have been muted. When a configuration object is initialized with `cl_syoscb_cfg::init()`, or the configuration's default comparer is first retrieved with `cl_syoscb_cfg::get_default_comparer()`, the default comparer's verbosity is set to `UVM_HIGH` to avoid cluttering the output to STDOUT.

Note that on UVM 1.1d, an error in the UVM source code has hard-coded the verbosity of some miscompare messages to `UVM_LOW`. These cannot be removed without modifying the UVM source used by the simulator. If possible, it is recommended to use another version of UVM where this problem does not exist.

7.2 Scoreboard dump

During simulation, the contents of the scoreboard's queues may be dumped to log files for postprocessing or inspection. Items may either be dumped to one file containing all transactions, or they may be dumped to separate files, one for each queue. Items may either be dumped using a standard `uvm_printer`, or they may be dumped using the `uvm_xml_printer`, which formats items into an XML-description which allows for easy postprocessing.

Scoreboard dumping is toggled with the config knob `cl_syoscb_cfg.full_scb_dump`. Other config knobs include `cl_syoscb_cfg.full_scb_dump_type` and `cl_syoscb_cfg.full_scb_dump_split`. See the entry on scoreboard dumping in [Overview of included tests](#) for further information.

7.3 Orphan dump

If items remain in any of the queues after simulation, these are seen as "orphans", which are generally treated as errors. Orphans may also be dumped to a log file once simulation finishes. This is toggled with the configuration knob [cl_syoscb_cfg.dump_orphans_to_files](#).

The same notes regarding the choice of printer apply as for the scoreboard dump.

7.4 XML printer

The SyoSil UVM Scoreboard comes with a printer that prints items as XML ([uvm_xml_printer](#)). This makes it simple to postprocess and transform the scoreboard or orphan dumps as desired.

Once a queue or scoreboard has been dumped with the printer, this file may be transformed into an HTML view of the items by using the make target

```
make generate_html XML_FILE=<filename>
```

Likewise, it may be transformed into the GraphML format by using the make target `generate_dot`

In the directory **lib/xml** the xsd file used to verify generated XML files, as well as XSLT-files for transforming to HTML and GraphML, are located.

Limitations

The current implementation of the XML printer does not support nested arrays, as these are not supported by UVM field macros.

Chapter 8

API Descriptions

The scoreboard presents several APIs that user-facing code can leverage. These APIs have been grouped based on their functionality, and are described in further detail in the sections below.

In general, all methods which are labeled with **API** are OK to call from user code. The only exception to this rule is methods labeled with **Compare Strategy API**. These methods should not be called from outside of a compare strategy, but they are labeled as a separate API to simplify custom compare strategy implementations. See also [Compare Strategy API](#) and the included figure of the comparison flowchart.

In the sections below, a brief overview of the APIs is presented. Selected methods from each API are listed here, but more exist. Follow the links to view each class' full list of methods. All methods labeled with **API** are part of the user-facing API, and have more in-depth explanations of their functionality, parameters and return values. On the class reference pages, all internal methods are also described in detail to aid in further extensions, but should not be called outside of the given class.

8.1 Configuration API

The configuration API is accessible in [cl_syoscb_cfg](#) and [cl_syoscbs_cfg](#). The methods of the configuration API are used to configure scoreboards and scoreboard wrappers. These methods are generally called in the UVM `build_phase`, setting configuration values before starting simulation.

Classes implementing this API

1. [cl_syoscb_cfg](#)
2. [cl_syoscbs_cfg](#)

Selected methods

See `pExampleConf` for an overview of all configuration knobs in [cl_syoscb_cfg](#). [cl_syoscbs_cfg](#) itself does not contain many config knobs, but includes methods for easily modifying all wrapped configuration objects.

8.2 Compare API

The compare API is used to start and control comparisons. The compare API is relatively small, whereas the [Compare Strategy API](#) contains the majority of the functions used to implement comparisons. The class [cl_syoscb_compare](#) inherits from `uvm_component` and is used to instantiate the desired compare strategy which inherits from [cl_syoscb_compare_base](#).

Classes implementing this API

1. [cl_syoscb_compare](#)
2. [cl_syoscb_compare_base](#)
3. [cl_syoscb_compare_io](#)
4. [cl_syoscb_compare_iop](#)
5. [cl_syoscb_compare_io_2hp](#)
6. [cl_syoscb_compare_ooo](#)

Selected methods

Name	Description
cl_syoscb_compare_base::compare_trigger	Initiates a comparison when an item has been inserted. If cl_syoscb_compare_base.disable_compare is set, the comparison is not performed
cl_syoscb_compare_base::compare_main	Initiates a comparison, bypassing the check of <code>disable_compare</code>
cl_syoscb_compare_base::compare_control	Disable or enable comparisons by calling this method

8.3 Queue API

The queue API is primarily used to insert items in queues. When an item has been added to the scoreboard via either the function-based API or a TLM connection, the scoreboard adds the item to a queue through the queue API. If other types of queues than the std. queues and hash queues are desired, these should also implement this API.

Classes implementing this API

1. [cl_syoscb_queue_base](#)
2. [cl_syoscb_queue_std](#)
3. [cl_syoscb_queue_hash](#)
4. [cl_syoscb_queue_hash_md5](#)

Selected methods

Name	Description
cl_syoscb_queue_base::add_item	Adds an item to the queue, placing it at the end of the queue.
cl_syoscb_queue_base::insert_item	Inserts an item at a specified index instead of placing it at the end of the queue.

Name	Description
cl_syoscb_queue_base::flush_queue	Flushes the queue, removing all items that previously occupied the queue
cl_syoscb_queue_base::create_iterator	Creates an iterator over this queue. See Iterator API for information on how iterators are used to traverse queues.
cl_syoscb_queue_base::get_item	Takes a proxy item as input, and uses this proxy item to return a specific item from the queue

8.4 Hash API

The hash API is used by hash algorithms to hash sequence items for use in hash queues.

Classes implementing this API

1. [cl_syoscb_hash_base](#)
2. [cl_syoscb_hash_md5](#)

Selected methods

Name	Description
cl_syoscb_hash_base::hash	Hashes a cl_syoscb_item using the specified hash algorithm, returning the bitstream representing that hash
cl_syoscb_hash_base::hash_str	Hashes a string using the specified hash algorithm
cl_syoscb_hash_base::do_hash	The underlying method which implements hashing of a bitstream. Can be used to hash items which are not strings or objects that extend cl_syoscb_item

8.5 Item API

The SyoSCB leverages multiple types of wrapper items to manage separation of concerns. The item API encompasses three different items' APIs, all of which are used to wrap sequence items when stored inside the scoreboard.

Classes implementing this API

1. [cl_syoscb_item](#)
 - Wraps a `uvm_sequence_item` generated by the DUT or a reference model with additional metadata used for comparisons.
2. [cl_syoscb_hash_item](#)
 - Used in hash queues to ensure that hash collisions, although unlikely, do not overwrite items in the underlying datastructure.
3. [cl_syoscb_proxy_item_base](#)
 - [cl_syoscb_proxy_item_std](#)
 - [cl_syoscb_proxy_item_hash](#)
 - Used with iterators and locators to separate the underlying queue's implementation and the act of iterating over it.

Selected methods

Name	Description
cl_syoscb_item::get_item	Gets the <code>uvm_sequence_item</code> wrapped by this scoreboard item
cl_syoscb_hash_item::get_item	Gets the first of possibly multiple items that have the same hash
cl_syoscb_proxy_item_base::get_item	Gets the item which this proxy item represents. The proxy item contains a handle to the queue, and the queue knows how to parse proxy items (see Queue API)

8.6 Iterator API

Iterators are used to iterate over queues in an implementation-agnostic manner. To iterate over all items in a queue, a loop of the following kind may be used:

```
if(iter.first()) begin //Reset the iterator, returns false if the queue is empty
    while(!iter.is_done()) begin
        //do something
        void' (iter.next());
    end
end
end
```

Classes implementing this API

1. [cl_syoscb_queue_iterator_base](#)
2. [cl_syoscb_queue_iterator_std](#)
3. [cl_syoscb_queue_iterator_hash](#)
4. [cl_syoscb_queue_iterator_hash_md5](#)

Selected methods

Name	Description
cl_syoscb_queue_iterator_base::next	Moves the iterator one item forward
cl_syoscb_queue_iterator_base::get_item_proxy	Gets a cl_syoscb_proxy_item_base representing the item that the iterator currently points to
cl_syoscb_queue_iterator_base::is_done	Checks whether the iterator has reached the end of the queue.

8.7 Locator API

Locators are used to find a specific item in a queue in an implementation-agnostic manner. Locators are primarily used for the OOO compare algorithm, where the item being searched for may exist anywhere in the queue.

Classes implementing this API

1. [cl_syoscb_queue_locator_base](#)
2. [cl_syoscb_queue_locator_std](#)
3. [cl_syoscb_queue_locator_hash](#)
4. [cl_syoscb_queue_locator_hash_md5](#)

Selected methods

Name	Description
cl_syoscb_queue_locator_base::search	Searches the underlying queue for an item which matches the argument.
cl_syoscb_queue_locator_base::get_queue	Gets a handle to the queue that this locator is operating on
cl_syoscb_queue_locator_base::set_queue	Sets the queue over which a locator should operate

8.8 Scoreboard API

The scoreboard API is used to insert items, either through the function-based API or through a TLM-based connection. Once inserted, the scoreboard contains a number of functions that can be used to generate statistics or dump items to log files.

Classes implementing this API

1. [cl_syoscb](#)

Selected methods

Name	Description
cl_syoscb::add_item	Adds an item to the scoreboard through the function based API. The item is inserted in a specific queue based on the arguments passed to the function.
cl_syoscb::get_subscriber	Gets a handle to a cl_syoscb_subscriber for a given queue/producer combination. Items written to this subscriber are added to the specified queue, tagged with the specified producer.
cl_syoscb::add_item_mutexed	Adds an item to the scoreboard, but acquires a mutex before doing so, ensuring that at most one insertion is ever taking place at once. Can only be used if cl_syoscb_cfg::mutexed_add_item_enable is set.
cl_syoscb::flush_queues	Flushes one or all queues of the scoreboard

8.9 Scoreboard Wrapper API

The scoreboard wrapper API is primarily used to easily manage and configure multiple identical scoreboards inside of a wrapper. The API is similar to the one presented for scoreboards, but may affect all scoreboards at the same time.

The wrapper API does not support adding items directly via the function based API. Instead, filter transforms are connected to agents as described in [Multiple SCB instances & filter transforms](#), or a handle to specific scoreboard may be extracted.

Classes implementing this API

1. [cl_syoscbs_base](#)
2. [cl_syoscbs](#)

Selected methods

Name	Description
cl_syoscbs::get_scb	Gets a handle to the scoreboard with index <i>i</i> .
cl_syoscbs::get_filter_trfm_base	Gets a filter transform component used to transform inputs to a specific queue from a specific producer.
cl_syoscb::flush_queues_all	Flushes all queues in all scoreboards at the same time

8.10 Compare Strategy API

The compare strategy API shall be used as a baseline in case custom comparisons must be implemented. By leveraging the compare strategy API, custom compare strategies should be interoperable with the compare strategies shipped with the SyoSil UVM Scoreboard.

Classes implementing this API

1. [cl_syoscb_compare_base](#)
2. [cl_syoscb_compare_io](#)
3. [cl_syoscb_compare_iop](#)
4. [cl_syoscb_compare_io_2hp](#)
5. [cl_syoscb_compare_ooo](#)

Selected methods

Name	Description
cl_syoscb_compare_base::primary_loop_do	A loop over the primary queue, selecting the item which should be found in all secondary queues
cl_syoscb_compare_base::secondary_loop_do	A loop over all secondary queues, attempting to find the same item as was selected from the primary queue. The manner of looping and which items are considered is defined by the compare strategy
cl_syoscb_compare_base::compare_do_greed	Initiates a comparison at the desired greed level. The scoreboard supports greedy comparisons (perform comparisons until a match is no longer found), or non-greedy comparisons (only perform 1 comparison whenever triggered, no matter if the comparison was successful or not)

Chapter 9

Overview of included tests

This section includes an overview of some of the included test cases, found in the **tb/test** directory. The list is not exhaustive, but should provide an indication of where to look for examples on how to use the available configuration knobs.

Feature	Relevant test cases
Demoting error verbosity	cl_scb_test_io_std_sbs_print cl_scb_test_ooo_std_dump_orphans
Full scoreboard dump (all transactions to one file)	cl_scb_test_io_std_dump
Split scoreboard dump (each queue in separate files)	cl_scb_test_io_std_dump_xml_split (in file cl_scb_test_io_std_dump_custom_printer)
Orphan dumping	cl_scb_test_ooo_std_dump_orphans cl_scb_test_ooo_std_dump_orphans_xml
Per-queue printer configuration	All tests in file cl_scb_test_io_std_dump_custom_printer
Per-queue comparer configuration	All tests in cl_scb_test_io_std_comparer_report
Using the function-based API	cl_scb_test_io_std_simple
Using the TLM based connections	cl_scb_test_ooo_std_tlm
Printing queue statistics while simulating	cl_scb_test_io_std_intermediate_dump
Mutexed add_item calls	cl_scb_test_io_std_tlm_mutexed
Using hash (MD5) queues	cl_scb_test_ooo_md5_simple
Using multiple queue types in one test	cl_scb_test_ooo_io_std_simple
Using the max_search_window configuration knob	cl_scb_test_ooo_std_max_search_window
Using filter transforms for transforming seq. items, using custom filters not derived from pk_utils_uvm::filter_trfm	cl_scb_test_ooo_std_tlm_filter_trfm cl_scbs_test_io_custom_filter_trfm
Using multiple scoreboards in a test	cl_scbs_test_io_std_base cl_scbs_test_io_std_cc
Dumping orphans/scoreboard to XML	cl_scb_test_ooo_std_dump_orphans_xml cl_scb_test_io_std_dump_xml_split

XML files generated with the XML printer can be converted into HTML files for easy viewing with the [generate_html](#) make target.

Chapter 10

Overview of configuration knobs

10.1 Included configuration knobs

The following table includes a reference of all configuration knobs in [cl_syoscb_cfg](#), as well as a short description of what each knob does. Click the link to the knob to view a more detailed explanation, as well as possible limitations.

In general, all getters and setters are named `get_<knob>` and `set_<knob>`. Depending on whether it is a global knob, a queue-specific knob or a queue/producer specific knob, the getter/setter may have multiple arguments. See the list of methods for [cl_syoscb_cfg](#) for additional details.

Configuration knob	Description
cl_syoscb_cfg.scb_name	The name of the scoreboard which this configuration is attached to.
cl_syoscb_cfg.queue_type	Type of queue used in the scoreboard. Defaults to <code>pk_syoscb::SYOSCB_QUEUE_USER_DEFINED</code> and must be changed if a custom queue topology is not used. All queue topologies defined in src/syoscb_common.svh are valid values.
cl_syoscb_cfg.compare_type	Type of compare algorithm to be used in the scoreboard. Defaults to <code>pk_syoscb::SYOSCB_COMPARE_USER_DEFINED</code> and must be changed if a custom compare strategy is not used. All compare strategies defined in src/syoscb_common.svh are valid values.
cl_syoscb_cfg.trigger_greediness	Greed level used when triggering comparisons during simulation. Defaults to <code>pk_syoscb::SYOSCB_COMPARE_NOT_GREEDY</code> . All greed levels defined in src/syoscb_common.svh are valid values .
cl_syoscb_cfg.end_greediness	Greed level used when triggering comparisons in the U↔VM cleanup phase. Defaults to <code>pk_syoscb::SYOSCB_COMPARE_GREEDY</code> . All greed levels defined in src/syoscb_common.svh are valid values .
cl_syoscb_cfg.enable_no_insert_check	Raise an error if any queue has no insertions at the end of simulation. Defaults to being enabled (1'b1).
cl_syoscb_cfg.disable_clone	Controls whether sequence items added to the scoreboard are cloned to disallow future modification. Clones are enabled by default (1'b0).

Configuration knob	Description
cl_syoscb_cfg.disable_compare_after_error	Controls whether comparisons should be disabled after a UVM_ERROR has been triggered. Defaults to still comparing items (1'b0).
cl_syoscb_cfg.max_queue_size	Per-queue knob controlling the maximum number of elements that can be in a queue before an error is raised. By default there is no limit on the number of elements in each queue (0).
cl_syoscb_cfg.print_orphans_as_errors	Controls whether orphans found in the UVM cleanup phase are reported as UVM_ERROR or UVM_INFO. Defaults to treating them as UVM_INFO (1'b0).
cl_syoscb_cfg.max_print_orphans	Control the maximum number of orphans that will be printed at end of simulation. Defaults to printing all orphans (0).
cl_syoscb_cfg.dump_orphans_to_files	Controls whether to dump orphans into log files at end of simulation. Defaults to not dumping orphans into log files (1'b0).
cl_syoscb_cfg.disable_report	Controls whether to disable the post-simulation report generated in the UVM report_phase. The report is enabled by default (1'b0).
cl_syoscb_cfg.enable_queue_stats	Per-queue knob enabling or disabling the printing of queue statistics per producer for each queue in simulation reports. Disabled by default (0, queues statistics do not include producer statistics).
cl_syoscb_cfg.full_scb_dump	Controls whether all transactions into the SCB should be dumped to a log file. Disabled by default (1'b0).
cl_syoscb_cfg.full_scb_dump_split	Controls whether transactions in a SCB dump should be written to the same file or individual files for each queue. Default to writing all transactions in the same file.
cl_syoscb_cfg.full_scb_max_queue_size	The number of elements that can be in a queue before dumping to file starts. Defaults to dumping every value when it is added (higher thresholds reduce the frequency of file I/O operations).
cl_syoscb_cfg.full_scb_dump_type	The type of file that the scoreboard should be dumped to. Can be either <code>pk_syoscb : : TEXT</code> (.txt file) or <code>pk_syoscb : : XML</code> (.XML file supporting XML transforms).
cl_syoscb_cfg.orphan_dump_type	The type of file that orphans should be dumped to. Supports the same knobs as cl_syoscb_cfg.full_scb_dump_type .
cl_syoscb_cfg.full_scb_dump_file_name	Prefix to be used in filenames when dumping scoreboard contents. Defaults to "full_scb_dump".
cl_syoscb_cfg.orphan_dump_file_name	Prefix to be used in filenames when dumping orphans. Defaults to "orphan_dump".
cl_syoscb_cfg.ordered_next	For hash-based queues, ensures that the iteration order over a queue is the same as the insertion order. Is enabled by default (1'b1), though this incurs a slight performance hit (see Queue implementation notes).

Configuration knob	Description
cl_syoscb_cfg.hash_compare_check	Toggles whether a safety check should be enabled when using hash queues. Can be used to check contents of matching items, or to ensure that no matches actually occur. All values of <code>t_hash_compare_check</code> defined in src/syoscb_common.svh are valid values.
cl_syoscb_cfg.print_cfg	Whether to print the scoreboard's configuration to STDOUT once the scoreboard has been built in the UVM build phase. Disabled by default (1'b0).
cl_syoscb_cfg.enable_comparer_report	Configuration knob for each queue/producer combination, controlling whether the specific fields that prompted a miscompare should be printed or not. If no queue/producer specific value has been set, uses cl_syoscb_cfg.default_enable_comparer_report .
cl_syoscb_cfg.default_enable_comparer_report	Default value for cl_syoscb_cfg.enable_comparer_report to use when printing miscompares if none is set for a specific queue/producer combination. Defaults to being enabled (1'b1).
cl_syoscb_cfg.comparers	Configuration knob for each queue/producer combination, allowing the use of a specific comparer. The primary item is used to select the comparer. If no specific comparer has been set for queue/producer combination, the default comparer is used instead.
cl_syoscb_cfg.default_comparer	The default comparer used when no specific queue/producer specific comparer has been set.
cl_syoscb_cfg.printer_verbosity	Per queue/producer configurable value that indirectly controls the number of elements in an array that a printer will output. If 1, all elements of arrays are printed. If 0, only prints the elements configured by the printer's begin/end elements flag.
cl_syoscb_cfg.default_printer_verbosity	The default printer verbosity bit used if none has been set for a queue/producer combination. Defaults to 1'b0 (use values of begin/end_elements flag in the printer).
cl_syoscb_cfg.printers	Per queue/producer configurable printer to be used when printing sequence items. If no specific printer has been set, uses the default printer instead.
cl_syoscb_cfg.default_printer	The default printer used when no specific printer has been set for a queue/producer combination. Defaults to being a <code>uvm_default_printer</code> .
cl_syoscb_cfg.max_search_window	Per-queue knob controlling the max number of items to check in that queue when performing OOO compare on STD queues, as well as IOP search for matching secondary items. If N=0, everything is searched, if N>0, only checks the first N items in each queue.
cl_syoscb_cfg.mutexed_add_item_enable	If toggled, a mutex must be acquired before items can be inserted into the queue. Note that all simulators that the SCB has been tested on use preemptive scheduling, removing the need for a mutex. This can be enabled for certainty that no insertions will overlap. Defaults to being disabled (1'b0).

Configuration knob	Description
cl_syoscb_cfg.queue_stat_interval	Per-queue knob that allows printing queue statistics (insertions, matches, flushed and remaining items) after every N insertions into that queue.
cl_syoscb_cfg.scb_stat_interval	Allows for printing overall SCB statistics to stdout after every N insertions into the scoreboard. SCB stats are the same figures as for queue stats, but are for the entire SCB.

10.2 Issues not resolved with config knobs

Some configuration issues which are not resolved through [cl_syoscb_cfg](#) are listed here:

1. Making the scoreboard stop after N compare errors:

- Since a compare error is issued as a UVM_ERROR, simply use the +UVM_MAX_QUIT_COUNT plusarg of UVM to control this.

Chapter 11

Hierarchical Index

11.1 Class Hierarchy

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

cl_scb_test_base	49
cl_scb_test_copy_cfg	54
cl_scb_test_double_scb	55
cl_scb_test_ooo_io_md5_simple	82
cl_scb_test_ooo_io_std_simple	83
cl_scb_test_rnd	93
cl_scb_test_benchmark	49
cl_scb_test_cmp_base< ATYPE, suffix >	50
cl_scb_test_cmp_base< ATYPE >	50
cl_scb_test_cmp_io< ATYPE, suffix >	51
cl_scb_test_cmp_ooo< ATYPE, suffix >	52
cl_scb_test_io_md5_disable_compare	59
cl_scb_test_io_md5_dump_orphans	59
cl_scb_test_io_std_comparer_printer	61
cl_scb_test_io_std_comparer_report	61
cl_scb_test_io_std_disable_compare	62
cl_scb_test_io_std_dump	62
cl_scb_test_io_std_dump_default	63
cl_scb_test_io_std_dump_mixed	65
cl_scb_test_io_std_dump_simple	66
cl_scb_test_io_std_dump_xml_join	67
cl_scb_test_io_std_dump_xml_split	68
cl_scb_test_io_std_dump_max_size	64
cl_scb_test_io_std_dump_max_size_less	64
cl_scb_test_io_std_insert_item	69
cl_scb_test_io_std_insert_item_md5	69
cl_scb_test_io_std_sbs_print	71
cl_scb_test_io_2hp_std_sbs_print	57
cl_scb_test_iop_std_sbs_print	75
cl_scb_test_io_std_simple	72
cl_scb_test_io_2hp_std_simple	58
cl_scb_test_io_2hp_md5_simple	56
cl_scb_test_io_md5_simple	60

cl_scb_test_io_std_intermediate_dump	70
cl_scb_test_io_std_simple_muxed	72
cl_scb_test_io_std_simple_real	73
cl_scb_test_io_std_tlm_gp_test	73
cl_scb_test_io_std_tlm_muxed	74
cl_scb_test_iop_md5_simple	74
cl_scb_test_iop_std_msw	74
cl_scb_test_iterator_correctness	76
cl_scb_test_iterator_unit_tests	76
cl_scb_test_iterator_unit_tests_md5	80
cl_scb_test_md5	81
cl_scb_test_md5_hash_collisions	81
cl_scb_test_ooo_heavy_base	81
cl_scb_test_ooo_md5_heavy	85
cl_scb_test_ooo_std_heavy	89
cl_scb_test_ooo_md5_duplets	84
cl_scb_test_ooo_md5_gp	84
cl_scb_test_ooo_md5_simple	85
cl_scb_test_ooo_md5_tlm	86
cl_scb_test_ooo_md5_validate	86
cl_scb_test_ooo_std_dump_orphans	87
cl_scb_test_ooo_std_dump_orphans_xml	88
cl_scb_test_ooo_std_dump_orphans_abort	87
cl_scb_test_ooo_std_gp	89
cl_scb_test_ooo_std_max_search_window	90
cl_scb_test_ooo_std_primary_multiple	90
cl_scb_test_ooo_std_simple	91
cl_scb_test_ooo_std_tlm	91
cl_scb_test_ooo_std_tlm_filter_trfm	92
cl_scb_test_ooo_std_trigger_greed	92
cl_scb_test_queue_find_vs_search	93
cl_scb_test_uvm_xml_printer	94
cl_scb_test_uvm_xml_printer_break	95
cl_scbs_test_base< FIN, MON, FT >	96
cl_scbs_test_io_std_base	98
cl_scbs_test_io_std_cc	100
cl_scbs_test_ooo_std_base	101
cl_scbs_test_ooo_std_flush	102
cl_scbs_test_base< cl_tb_seq_item, cl_tb_tlm_monitor< cl_tb_seq_item >, my_custom_filter_trfm< cl← _tb_seq_item, uvm_sequence_item > >	96
cl_scbs_test_io_custom_filter_trfm	98
cl_scbs_test_base< cl_tb_seq_item_par< 8 >, cl_tb_tlm_monitor_param< cl_tb_seq_item_par< 8 > > >	96
cl_scbs_test_filter_trfm_param	97
cl_syoscb	104
cl_syoscb_cfg	117
cl_syoscb_cfg_pl	151
cl_syoscb_compare	152
cl_syoscb_compare_base	154
cl_syoscb_compare_io	165
cl_syoscb_compare_io_2hp	169
cl_syoscb_compare_io_2hp	169
cl_syoscb_compare_io	165
cl_syoscb_compare_iop	172
cl_syoscb_compare_iop	172
cl_syoscb_compare_ooo	177

cl_syoscb_compare_ooo	177
cl_syoscb_comparer_config	180
cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >	184
cl_syoscb_hash_base< HASH_DIGEST_WIDTH >	191
cl_syoscb_hash_base< pk_syoscb::MD5_HASH_DIGEST_WIDTH >	191
cl_syoscb_hash_md5	198
cl_syoscb_hash_md5	198
cl_syoscb_hash_item	195
cl_syoscb_hash_packer	200
cl_syoscb_md5_packer	204
cl_syoscb_md5_packer	204
cl_syoscb_item	201
cl_syoscb_printer_config	205
cl_syoscb_proxy_item_base	210
cl_syoscb_proxy_item_hash< HASH_DIGEST_WIDTH >	212
cl_syoscb_proxy_item_hash< HASH_DIGEST_WIDTH >	212
cl_syoscb_proxy_item_std	214
cl_syoscb_proxy_item_std	214
cl_syoscb_queue_base	215
cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >	232
cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >	232
cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >	232
cl_syoscb_queue_hash_md5	239
cl_syoscb_queue_hash_md5	239
cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >	232
cl_syoscb_queue_std	268
cl_syoscb_queue_std	268
cl_syoscb_queue_iterator_base	241
cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH >	247
cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH >	247
cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >	247
cl_syoscb_queue_iterator_hash_md5	253
cl_syoscb_queue_iterator_hash_md5	253
cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >	247
cl_syoscb_queue_iterator_std	254
cl_syoscb_queue_iterator_std	254
cl_syoscb_queue_locator_base	258
cl_syoscb_queue_locator_hash< HASH_DIGEST_WIDTH >	260
cl_syoscb_queue_locator_hash< HASH_DIGEST_WIDTH >	260
cl_syoscb_queue_locator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >	260
cl_syoscb_queue_locator_hash_md5	264
cl_syoscb_queue_locator_hash_md5	264
cl_syoscb_queue_locator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >	260
cl_syoscb_queue_locator_std	265
cl_syoscb_queue_locator_std	265
cl_syoscb_string_library	274
cl_syoscb_subscriber	280
cl_syoscb_base	285
cl_syoscb< FIN >	281
cl_syoscb< FIN >	281
cl_syoscb_cfg	295
cl_tb_cmp_seq_item_base< TIOBJ, MAX_ARRAY_SIZE >	315
cl_tb_cmp_seq_item_base< T >	315
cl_tb_cmp_a_f_seq_item< T >	308
cl_tb_cmp_seq_item_base< TIOBJ >	315

cl_tb_cmp_a_d_seq_item< TIOBJ >	307
cl_tb_cmp_a_m_seq_item< TIOBJ >	309
cl_tb_cmp_b_d_seq_item< TIOBJ >	311
cl_tb_cmp_b_f_seq_item< TIOBJ >	312
cl_tb_cmp_b_m_seq_item< TIOBJ >	314
pk_utils_uvm::filter_trfm< IN, OUT >	316
pk_utils_uvm::filter_trfm< cl_tb_seq_item >	316
cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >::packed	319
cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >::tp_item_digest	319
pk_syoscb::uvm_xml_printer	320
uvm_xml_printer	321

Chapter 12

Class Index

12.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

cl_scb_test_base	Base class for all SCB tests	49
cl_scb_test_benchmark	Benchmark to compare performance of STD and Hash queues when executing OOO compare	49
cl_scb_test_cmp_base< ATYPE, suffix >	Base class for field macro/manual do_compare comparison tests	50
cl_scb_test_cmp_io< ATYPE, suffix >	Base class for specializations of cl_scb_test_cmp_base using IO compare	51
cl_scb_test_cmp_ooo< ATYPE, suffix >	Base class for specializations of cl_scb_test_cmp_base using OOO compare	52
cl_scb_test_copy_cfg	This test is used to ensure that copying data from one cl_syoscb_cfg object to the next correctly moves over all information	54
cl_scb_test_double_scb	Base class for all SCB tests usings two scoreboards	55
cl_scb_test_io_2hp_md5_simple	IO-2HP test using MD5 queues	56
cl_scb_test_io_2hp_std_sbs_print	Tests side-by-side error prints when using io-2hp comparison	57
cl_scb_test_io_2hp_std_simple	Simple IO-2HP compare test using the function based API	58
cl_scb_test_io_md5_disable_compare	Tests the ability to flush queues and disable compare during runtime for md5 hashed queues	59
cl_scb_test_io_md5_dump_orphans	Uses io-compare and MD5 queues showing orphan dumps & shadow queues / scb dump	59
cl_scb_test_io_md5_simple	Simple IO compare test using the function based API and md5 queues	60
cl_scb_test_io_std_comparer_printer	Tests uvm_comparer and uvm_printer related features	61
cl_scb_test_io_std_comparer_report	Illustrates how to use queue/producer-specific comparers instead of a default comparer	61
cl_scb_test_io_std_disable_compare	Tests the ability to flush queues and disable compare during runtime for std queues	62
cl_scb_test_io_std_dump	SCB dump test using the function based API	62

cl_scb_test_io_std_dump_default	Sets the default printer override. Since no specific printers are set, all queues rely on the default printer	63
cl_scb_test_io_std_dump_max_size	SCB dump test using the function based API	64
cl_scb_test_io_std_dump_max_size_less	Shows that SCB dumping still works when <code>full_scb_max_queue_size</code> > the actual number of transactions	64
cl_scb_test_io_std_dump_mixed	Sets the default printer override as well as specific printer overrides	65
cl_scb_test_io_std_dump_simple	Sets custom printer overrides for Q1/P1 and Q2/P2. The remaining queues will use the default printer	66
cl_scb_test_io_std_dump_xml_join	Uses XML/join printing to generate a single XML file once the test is finished	67
cl_scb_test_io_std_dump_xml_split	Uses XML/split printing to generate multiple XML files once the test is finished	68
cl_scb_test_io_std_insert_item	IO test that validates the behavior of cl_syoscb_queue_base::insert_item	69
cl_scb_test_io_std_insert_item_md5	IO test that validates the behavior of cl_syoscb_queue_base::insert_item when using MD5 queues	69
cl_scb_test_io_std_intermediate_dump	Tests the intermediate queue stat printout, see cl_syoscb_cfg::queue_stat_interval	70
cl_scb_test_io_std_sbs_print	Shows a number of different ways that the side-by-side miscompare table can be used	71
cl_scb_test_io_std_simple	Simple IO compare test using the function based API	72
cl_scb_test_io_std_simple_mutexed	Simple IO compare test using the function based API and mutexed <code>add_item</code> calls	72
cl_scb_test_io_std_simple_real	Simple IO compare test on real values using the function based API	73
cl_scb_test_io_std_tlm_gp_test	IO comparison test to ensure that the SYOSIL TLM GP comparison workaround works as expected	73
cl_scb_test_io_std_tlm_mutexed	Simple IO test using the TLM ssetup and mutexed <code>add_item</code>	74
cl_scb_test_iop_md5_simple	Simple IOP compare test using the function based API and MD5 queues	74
cl_scb_test_iop_std_msw	This test ensures that IOP compares correctly search through the primary queue, comparing not only the first item but also subsequent items for matches	74
cl_scb_test_iop_std_sbs_print	A functional copy of the test seen in cl_scb_test_io_std_sbs_print , shows that miscompare table work for IOP compare	75
cl_scb_test_iterator_correctness	Test to ensure that multiple iterators on the same queue won't deadlock and are performing correctly	76
cl_scb_test_iterator_unit_tests	Test containing a series of unit tests to ensure that all iterators conform to spec	76
cl_scb_test_iterator_unit_tests_md5	Test that md5-iterators using cl_syoscb_cfg::ordered_next conform to spec	80
cl_scb_test_md5	Test which verifies that the md5 hash implementation works correctly	81
cl_scb_test_md5_hash_collisions	Test to verify that comparisons still work correctly on hash items with multiple entries where hash collisions may have occurred	81

cl_scb_test_ooo_heavy_base	Heavy OOO compare test using the function based API	81
cl_scb_test_ooo_io_md5_simple	Simple test with two SCBs with different compares, both using MD5 queues	82
cl_scb_test_ooo_io_std_simple	Simple test with two SCBs with different compares and standard queues	83
cl_scb_test_ooo_md5_duplets	Duplets OOO compare test using the function based API	84
cl_scb_test_ooo_md5_gp	Simple OOO compare test for TLM generic payload using the function based API	84
cl_scb_test_ooo_md5_heavy	Heavy OOO compare test using the function based API and MD5 queues	85
cl_scb_test_ooo_md5_simple	Simple OOO compare test using the function based API and MD5 queues	85
cl_scb_test_ooo_md5_tlm	Simple OOO compare test using the TLM based API and MD5 queues	86
cl_scb_test_ooo_md5_validate	Test to ensure that config knob cl_syoscb_cfg::hash_compare_check correctly controls MD5 validation behavior	86
cl_scb_test_ooo_std_dump_orphans	This test uses the dump_orphans_to_files configuration knob	87
cl_scb_test_ooo_std_dump_orphans_abort	Tests queue and orphan dumping when an error occurs mid-simulation This test fails on purpose, and is therefore not included in the regression tests	87
cl_scb_test_ooo_std_dump_orphans_xml	Dumping orphans to files using XML printout	88
cl_scb_test_ooo_std_gp	Simple OOO compare test for TLM generic payload using the function based API	89
cl_scb_test_ooo_std_heavy	Heavy OOO compare test using the function based API and a standard queue	89
cl_scb_test_ooo_std_max_search_window	Simple OOO compare test using the function based API and the max_search_window knob to control OOO compare searches	90
cl_scb_test_ooo_std_primary_multiple	OOO compare test for ensuring that multiple items in the primary queue are checked against the secondary queue	90
cl_scb_test_ooo_std_simple	Simple OOO compare test using the function based API	91
cl_scb_test_ooo_std_tlm	Simple OOO compare test using the TLM based API	91
cl_scb_test_ooo_std_tlm_filter_trfm	Simple OOO compare test using the TLM based API and filter transforms	92
cl_scb_test_ooo_std_trigger_greed	OOO Compare test for validating that OOO compares respect the current greed level	92
cl_scb_test_queue_find_vs_search	A test comparing the performance of using iterators vs using .find_first on a SV queue	93
cl_scb_test_rnd	Random test to hit all the coverage holes which are not covered by tests derived from cl_scb_test_double_scb	93
cl_scb_test_uvm_xml_printer	A test which can be used to generate an XML printout for verifying the uvm_xml_printer	94
cl_scb_test_uvm_xml_printer_break	Tests whether the uvm_xml_printer correctly outputs a warning when it is used on a non-cl _← syoscb_item sequence item	95
cl_scbs_test_base< FIN, MON, FT >	Base class for all SCBs tests	96
cl_scbs_test_filter_trfm_param	SCBs test using a parameterized sequence item and filter transforms	97

cl_scbs_test_io_custom_filter_trfm	SCBs test using a filter transform not inherited from <code>pk_uvm_utils::filter_trfm</code> , to show that all types of filter transforms work	98
cl_scbs_test_io_std_base	Simple IO compare with standard queues using TLM based API	98
cl_scbs_test_io_std_cc	Simple IO compare with STD queue test. Testing the cl_syoscbs class	100
cl_scbs_test_ooo_std_base	Simple OOO compare with STD queue test. Testing the cl_syoscbs class	101
cl_scbs_test_ooo_std_flush	Simple OOO compare with STD queue test which inserts additional random items, requiring a flush at the end to pass the test	102
cl_syoscb	Top level class implementing the root of the SyoSil UVM scoreboard	104
cl_syoscb_cfg	Configuration class for the SyoSil UVM scoreboard	117
cl_syoscb_cfg_pl	Utility class for capturing the queue names associated with a producer	151
cl_syoscb_compare	Component which instantiates the chosen comparison algorithm	152
cl_syoscb_compare_base	Base class for all compare algorithms	154
cl_syoscb_compare_io	Implementation of the in-order comparison algorithm for N queues	165
cl_syoscb_compare_io_2hp	Implementation of the 2-queue, high speed in-order comparison algorithm	169
cl_syoscb_compare_iop	Class which implements the in order by producer compare algorithm	172
cl_syoscb_compare_ooo	Class which implements the out of order compare algorithm	177
cl_syoscb_comparer_config	Utility class used to perform manipulations of <code>uvm_comparer</code> objects	180
cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >	A wrapper around an associative array, used for storing hash queues	184
cl_syoscb_hash_base< HASH_DIGEST_WIDTH >	Class which defines the base concept of a hash algorithm	191
cl_syoscb_hash_item	A utility class used to wrap cl_syoscb_item objects when using hash queues	195
cl_syoscb_hash_md5	MD5 hash algorithm implementation	198
cl_syoscb_hash_packer	A base class for packers which should be used with hash algorithms in the scoreboard	200
cl_syoscb_item	The UVM scoreboard item which wraps <code>uvm_sequence_item</code>	201
cl_syoscb_md5_packer	An implementation of a <code>uvm_packer</code> which returns bitstreams that are ready for md5 packing	204
cl_syoscb_printer_config	Utility class used to perform manipulations of <code>uvm_printer</code> objects	205
cl_syoscb_proxy_item_base	Base class for all proxy items	210
cl_syoscb_proxy_item_hash< HASH_DIGEST_WIDTH >	Proxy item implementation for hash queues	212
cl_syoscb_proxy_item_std	Proxy item implementation for standard queues	214
cl_syoscb_queue_base	Class which represents the base concept of a queue	215
cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >	Class which represents the base concept of a hash queue	232

cl_syoscb_queue_hash_md5	
MD5 implementation of a hash queue which optimizes the OOO compare	239
cl_syoscb_queue_iterator_base	
Queue iterator base class defining the iterator API used for iterating over queues	241
cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH >	
Queue iterator class defining the iterator API used for iterating hash queues	247
cl_syoscb_queue_iterator_hash_md5	
Queue iterator class defining the iterator API used for iterating md5 hash queues	253
cl_syoscb_queue_iterator_std	
Queue iterator class for iterating over std queues	254
cl_syoscb_queue_locator_base	
Locator base class defining the locator API used for searching in queues	258
cl_syoscb_queue_locator_hash< HASH_DIGEST_WIDTH >	
Locator class for searching over generic hash queues	260
cl_syoscb_queue_locator_hash_md5	
Locator class for searching over hash queues using md5 as the hash algorithm	264
cl_syoscb_queue_locator_std	
Locator class for searching over std queues	265
cl_syoscb_queue_std	
Standard implementation of a queue	268
cl_syoscb_string_library	
A utility class holding a number of static methods for performing string manipulation	274
cl_syoscb_subscriber	
Generic subscriber for the scoreboard	280
cl_syoscbcs< FIN >	
Default implementation of a scoreboard wrapper	281
cl_syoscbcs_base	
Base class for a wrapper around multiple SyoSil Scoreboards	285
cl_syoscbcs_cfg	
Configuration object for the cl_syoscbcs_base scoreboard wrapper	295
cl_tb_cmp_a_d_seq_item< TIOBJ >	
An "a" type item which used a manual do_compare implementation instead of field macros	307
cl_tb_cmp_a_f_seq_item< T >	
An "a" type item which used a field macros instead of manually implementing do_compare	308
cl_tb_cmp_a_m_seq_item< TIOBJ >	
A "b" type item which used a mix of do_compare implementation and field macros	309
cl_tb_cmp_b_d_seq_item< TIOBJ >	
A "b" type item which used a manual do_compare implementation instead of field macros	311
cl_tb_cmp_b_f_seq_item< TIOBJ >	
A "b" type item which used a field macros instead of manually implementing do_compare	312
cl_tb_cmp_b_m_seq_item< TIOBJ >	
An "a" type item which used a mix of do_compare implementation and field macros	314
cl_tb_cmp_seq_item_base< TIOBJ, MAX_ARRAY_SIZE >	
A sequence item to be used in cmp-tests extending from cl_scb_test_cmp_base	315
pk_utils_uvm::filter_trfm< IN, OUT >	
Base class for a filter transformation	316
cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >::packed	
Typedef for struct representing whether an option with an iterator was valid	319
cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >::tp_item_digest	
Typedef for struct used to track items and their digests in the key queue	319
pk_syoscb::uvm_xml_printer	
An XML printer for cl_syoscb_items	320
uvm_xml_printer	
An XML printer for cl_syoscb_items	321

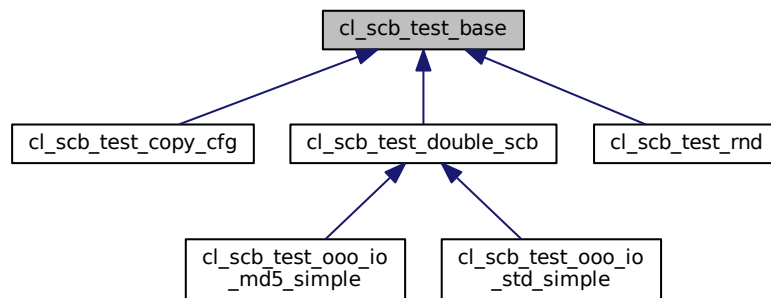
Chapter 13

Class Documentation

13.1 cl_scb_test_base Class Reference

Base class for all SCB tests.

Inheritance diagram for cl_scb_test_base:



13.1.1 Detailed Description

Base class for all SCB tests.

Definition at line 2 of file `cl_scb_test_base.svh`.

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

- `cl_scb_test_base.svh`

13.2 cl_scb_test_benchmark Class Reference

Benchmark to compare performance of STD and Hash queues when executing OOO compare.

Inherits `cl_scb_test_single_scb`.

Inherited by `cl_scb_test_benchmark_md5`, `cl_scb_test_benchmark_md5_on`, and `cl_scb_test_benchmark_std`.

13.2.1 Detailed Description

Benchmark to compare performance of STD and Hash queues when executing OOO compare.

Definition at line 3 of file `cl_scb_test_benchmark.svh`.

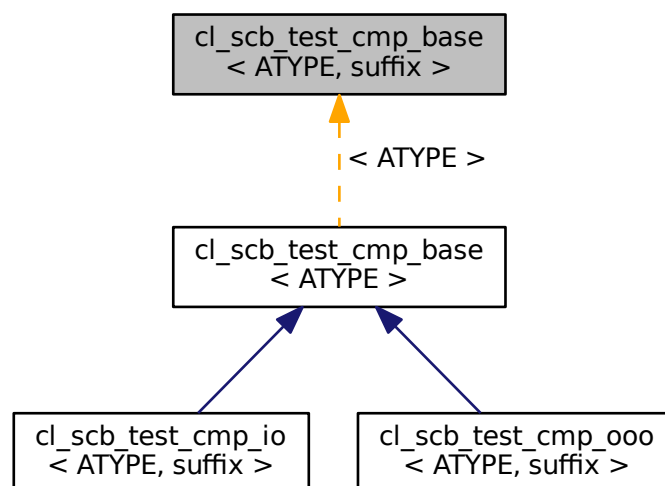
The documentation for this class was generated from the following file:

- `cl_scb_test_benchmark.svh`

13.3 `cl_scb_test_cmp_base< ATYPE, suffix >` Class Template Reference

Base class for field macro/manual `do_compare` comparison tests.

Inheritance diagram for `cl_scb_test_cmp_base< ATYPE, suffix >`:



13.3.1 Detailed Description

```
template<typename ATYPE = cl_tb_cmp_a_f_seq_item<cl_tb_cmp_b_f_seq_item<cl_tb_seq_item> >, string suffix = "">
class cl_scb_test_cmp_base< ATYPE, suffix >
```

Base class for field macro/manual `do_compare` comparison tests.

These tests serve to make sure that a mix of field macros and manual `do_compare` implementations evaluate correctly. Does this by using objects of type 'a', which has a handle to a type 'b' object, which has a handle to an endpoint object. This allows us to chain field macros/`do_compare`/mixed implementations

Parameters

<i>ATYPE</i>	Type of the top-level objects to instantiate
<i>suffix</i>	A suffix to add to the test name. The testname will be "cl_scb_test_cmp_<io/ooo>_<suffix>"

Definition at line 8 of file cl_scb_test_cmp_base.svh.

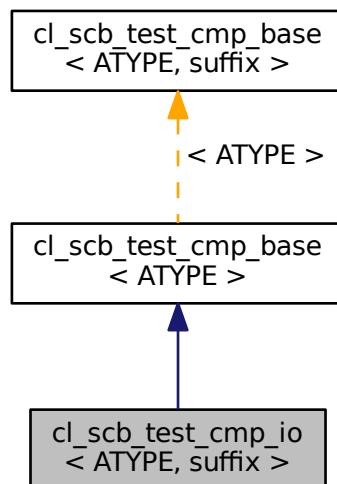
The documentation for this class was generated from the following file:

- cl_scb_test_cmp_base.svh

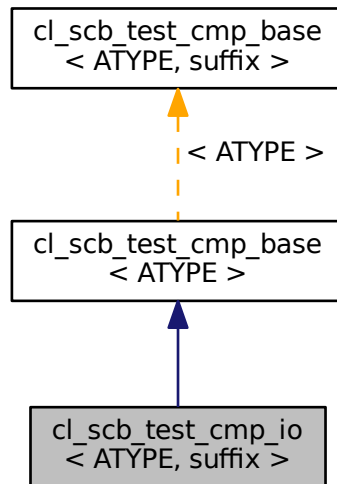
13.4 cl_scb_test_cmp_io< ATYPE, suffix > Class Template Reference

Base class for specializations of [cl_scb_test_cmp_base](#) using IO compare.

Inheritance diagram for cl_scb_test_cmp_io< ATYPE, suffix >:



Collaboration diagram for `cl_scb_test_cmp_io< ATYPE, suffix >`:



13.4.1 Detailed Description

```
template<typename ATYPE = cl_tb_cmp_a_f_seq_item<cl_tb_cmp_b_f_seq_item<cl_tb_seq_item> >, string suffix = "">
class cl_scb_test_cmp_io< ATYPE, suffix >
```

Base class for specializations of [cl_scb_test_cmp_base](#) using IO compare.

Implementations are provided below using typedefs

Parameters

<i>ATYPE</i>	Type of the top-level objects to instantiate
<i>suffix</i>	A string suffix to add to the nest name

Definition at line 6 of file `cl_scb_test_cmp_io.svh`.

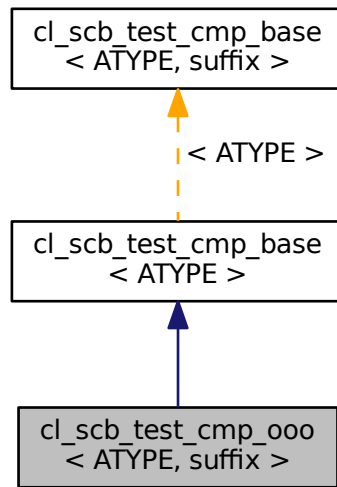
The documentation for this class was generated from the following file:

- `cl_scb_test_cmp_io.svh`

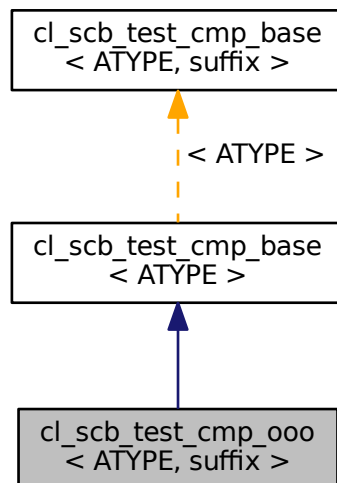
13.5 `cl_scb_test_cmp_ooo< ATYPE, suffix >` Class Template Reference

Base class for specializations of [cl_scb_test_cmp_base](#) using OOO compare.

Inheritance diagram for cl_scb_test_cmp_ooo< ATYPE, suffix >:



Collaboration diagram for cl_scb_test_cmp_ooo< ATYPE, suffix >:



13.5.1 Detailed Description

```
template<typename ATYPE = cl_tb_cmp_a_f_seq_item<cl_tb_cmp_b_f_seq_item<cl_tb_seq_item> >, string suffix = "">
class cl_scb_test_cmp_ooo< ATYPE, suffix >
```

Base class for specializations of [cl_scb_test_cmp_base](#) using OOO compare.

Implementations are provided below using typedefs

Parameters

<i>ATYPE</i>	Type of the top-level objects to instantiate
<i>suffix</i>	A string suffix to add to the nest name

Definition at line 6 of file `cl_scb_test_cmp_ooo.svh`.

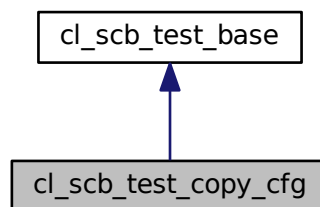
The documentation for this class was generated from the following file:

- `cl_scb_test_cmp_ooo.svh`

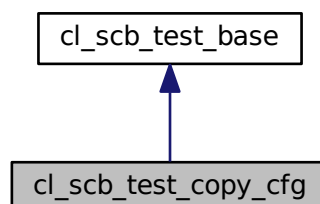
13.6 `cl_scb_test_copy_cfg` Class Reference

This test is used to ensure that copying data from one `cl_syoscb_cfg` object to the next correctly moves over all information.

Inheritance diagram for `cl_scb_test_copy_cfg`:



Collaboration diagram for `cl_scb_test_copy_cfg`:



13.6.1 Detailed Description

This test is used to ensure that copying data from one [cl_syoscb_cfg](#) object to the next correctly moves over all information.

Definition at line 2 of file `cl_scb_test_copy_cfg.svh`.

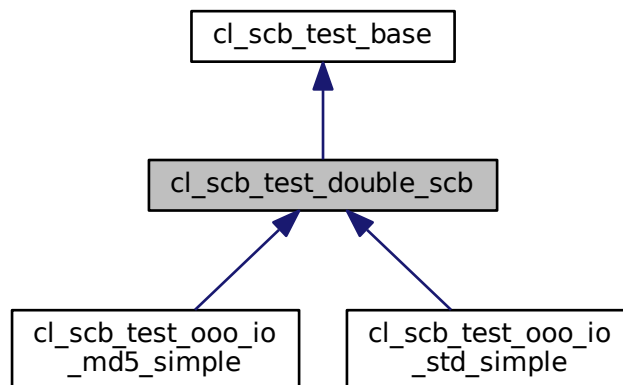
The documentation for this class was generated from the following file:

- `cl_scb_test_copy_cfg.svh`

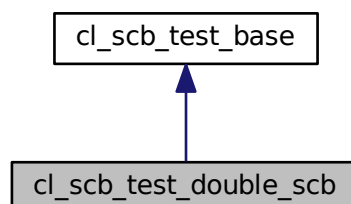
13.7 cl_scb_test_double_scb Class Reference

Base class for all SCB tests using two scoreboards.

Inheritance diagram for `cl_scb_test_double_scb`:



Collaboration diagram for `cl_scb_test_double_scb`:



13.7.1 Detailed Description

Base class for all SCB tests using two scoreboards.

Definition at line 3 of file `cl_scb_test_double_scb.svh`.

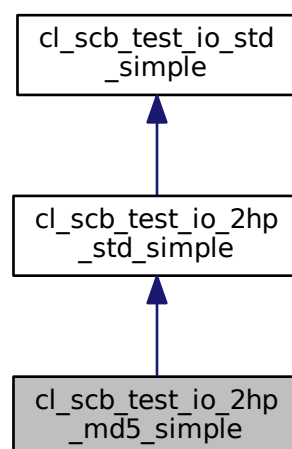
The documentation for this class was generated from the following file:

- `cl_scb_test_double_scb.svh`

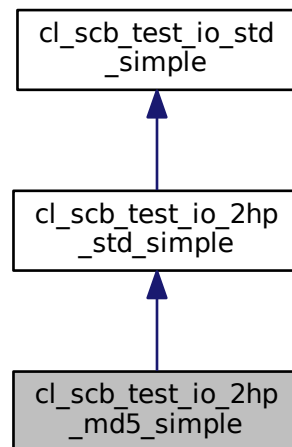
13.8 `cl_scb_test_io_2hp_md5_simple` Class Reference

IO-2HP test using MD5 queues.

Inheritance diagram for `cl_scb_test_io_2hp_md5_simple`:



Collaboration diagram for cl_scb_test_io_2hp_md5_simple:



13.8.1 Detailed Description

IO-2HP test using MD5 queues.

Definition at line 2 of file cl_scb_test_io_2hp_md5_simple.svh.

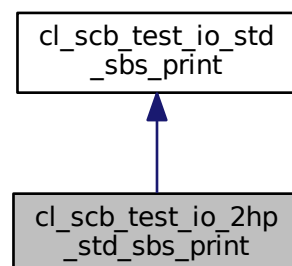
The documentation for this class was generated from the following file:

- cl_scb_test_io_2hp_md5_simple.svh

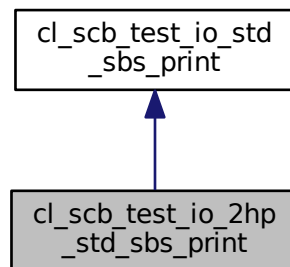
13.9 cl_scb_test_io_2hp_std_sbs_print Class Reference

Tests side-by-side error prints when using io-2hp comparison.

Inheritance diagram for cl_scb_test_io_2hp_std_sbs_print:



Collaboration diagram for `cl_scb_test_io_2hp_std_sbs_print`:



13.9.1 Detailed Description

Tests side-by-side error prints when using io-2hp comparison.

Definition at line 4 of file `cl_scb_test_io_2hp_std_sbs_print.svh`.

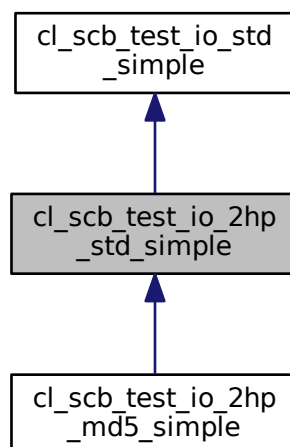
The documentation for this class was generated from the following file:

- `cl_scb_test_io_2hp_std_sbs_print.svh`

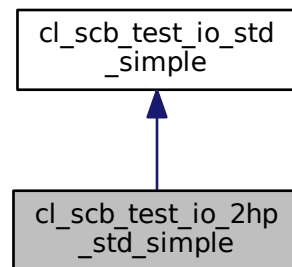
13.10 `cl_scb_test_io_2hp_std_simple` Class Reference

Simple IO-2HP compare test using the function based API.

Inheritance diagram for `cl_scb_test_io_2hp_std_simple`:



Collaboration diagram for cl_scb_test_io_2hp_std_simple:



13.10.1 Detailed Description

Simple IO-2HP compare test using the function based API.

Definition at line 2 of file `cl_scb_test_io_2hp_std_simple.svh`.

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

- `cl_scb_test_io_2hp_std_simple.svh`

13.11 cl_scb_test_io_md5_disable_compare Class Reference

Tests the ability to flush queues and disable compare during runtime for md5 hashed queues.

Inherits `cl_scb_test_single_scb`.

13.11.1 Detailed Description

Tests the ability to flush queues and disable compare during runtime for md5 hashed queues.

Definition at line 3 of file `cl_scb_test_io_md5_disable_compare.svh`.

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

- `cl_scb_test_io_md5_disable_compare.svh`

13.12 cl_scb_test_io_md5_dump_orphans Class Reference

Uses io-compare and MD5 queues showing orphan dumps & shadow queues / scb dump.

Inherits `cl_scb_test_single_scb`.

13.12.1 Detailed Description

Uses io-compare and MD5 queues showing orphan dumps & shadow queues / scb dump.

Definition at line 3 of file cl_scb_test_io_md5_dump_orphans.svh.

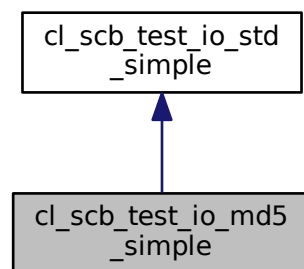
The documentation for this class was generated from the following file:

- cl_scb_test_io_md5_dump_orphans.svh

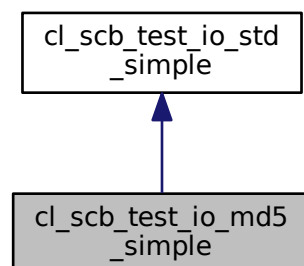
13.13 cl_scb_test_io_md5_simple Class Reference

Simple IO compare test using the function based API and md5 queues.

Inheritance diagram for cl_scb_test_io_md5_simple:



Collaboration diagram for cl_scb_test_io_md5_simple:



13.13.1 Detailed Description

Simple IO compare test using the function based API and md5 queues.

Definition at line 2 of file cl_scb_test_io_md5_simple.svh.

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

- cl_scb_test_io_md5_simple.svh

13.14 cl_scb_test_io_std_comparer_printer Class Reference

Tests uvm_comparer and uvm_printer related features.

Inherits cl_scb_test_single_scb.

13.14.1 Detailed Description

Tests uvm_comparer and uvm_printer related features.

Definition at line 2 of file cl_scb_test_io_std_comparer_printer.svh.

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

- cl_scb_test_io_std_comparer_printer.svh

13.15 cl_scb_test_io_std_comparer_report Class Reference

Illustrates how to use queue/producer-specific comparers instead of a default comparer.

Inherits cl_scb_test_single_scb.

13.15.1 Detailed Description

Illustrates how to use queue/producer-specific comparers instead of a default comparer.

Definition at line 3 of file cl_scb_test_io_std_comparer_report.svh.

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

- cl_scb_test_io_std_comparer_report.svh

13.16 cl_scb_test_io_std_disable_compare Class Reference

Tests the ability to flush queues and disable compare during runtime for std queues.

Inherits cl_scb_test_single_scb.

13.16.1 Detailed Description

Tests the ability to flush queues and disable compare during runtime for std queues.

Definition at line 3 of file cl_scb_test_io_std_disable_compare.svh.

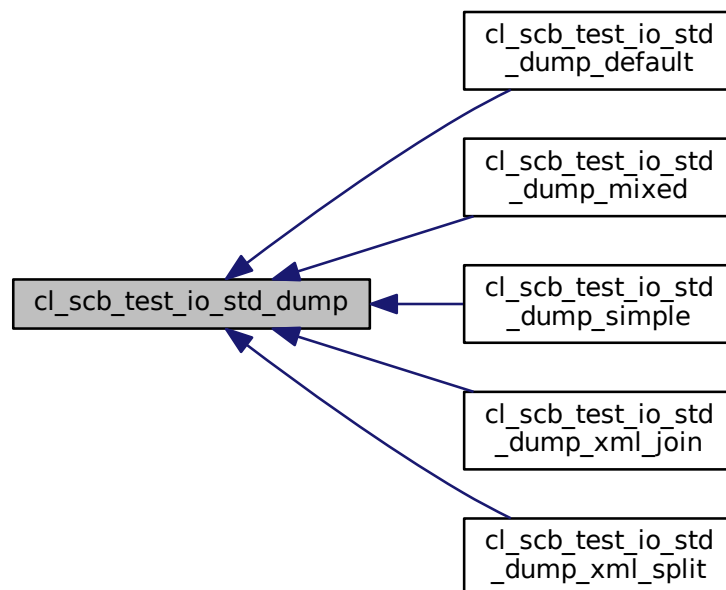
The documentation for this class was generated from the following file:

- cl_scb_test_io_std_disable_compare.svh

13.17 cl_scb_test_io_std_dump Class Reference

SCB dump test using the function based API.

Inheritance diagram for cl_scb_test_io_std_dump:



13.17.1 Detailed Description

SCB dump test using the function based API.

Definition at line 2 of file cl_scb_test_io_std_dump.svh.

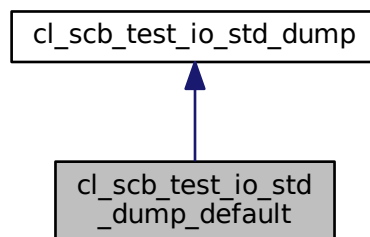
The documentation for this class was generated from the following file:

- cl_scb_test_io_std_dump.svh

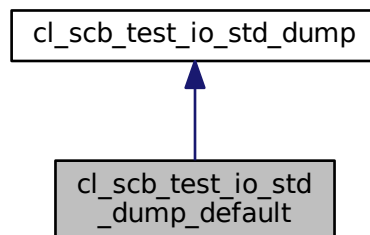
13.18 cl_scb_test_io_std_dump_default Class Reference

Sets the default printer override. Since no specific printers are set, all queues rely on the default printer.

Inheritance diagram for cl_scb_test_io_std_dump_default:



Collaboration diagram for cl_scb_test_io_std_dump_default:



13.18.1 Detailed Description

Sets the default printer override. Since no specific printers are set, all queues rely on the default printer.

Definition at line 43 of file `cl_scb_test_io_std_dump_custom_printer.svh`.

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

- `cl_scb_test_io_std_dump_custom_printer.svh`

13.19 `cl_scb_test_io_std_dump_max_size` Class Reference

SCB dump test using the function based API.

Inherits `cl_scb_test_single_scb`.

13.19.1 Detailed Description

SCB dump test using the function based API.

Definition at line 3 of file `cl_scb_test_io_std_dump_max_size.svh`.

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

- `cl_scb_test_io_std_dump_max_size.svh`

13.20 `cl_scb_test_io_std_dump_max_size_less` Class Reference

Shows that SCB dumping still works when `full_scb_max_queue_size >` the actual number of transactions.

Inherits `cl_scb_test_single_scb`.

13.20.1 Detailed Description

Shows that SCB dumping still works when `full_scb_max_queue_size >` the actual number of transactions.

Definition at line 2 of file `cl_scb_test_io_std_dump_max_size_less.svh`.

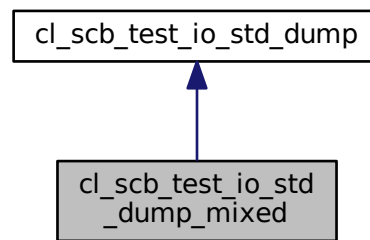
The documentation for this class was generated from the following file:

- `cl_scb_test_io_std_dump_max_size_less.svh`

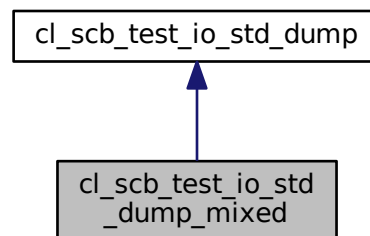
13.21 cl_scb_test_io_std_dump_mixed Class Reference

Sets the default printer override as well as specific printer overrides.

Inheritance diagram for cl_scb_test_io_std_dump_mixed:



Collaboration diagram for cl_scb_test_io_std_dump_mixed:



13.21.1 Detailed Description

Sets the default printer override as well as specific printer overrides.

Definition at line 81 of file cl_scb_test_io_std_dump_custom_printer.svh.

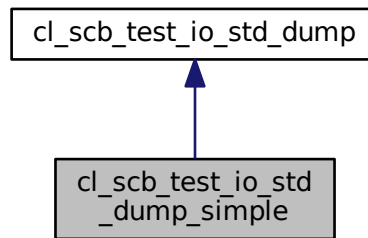
The documentation for this class was generated from the following file:

- cl_scb_test_io_std_dump_custom_printer.svh

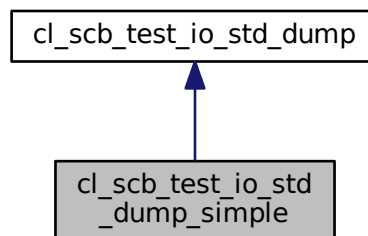
13.22 cl_scb_test_io_std_dump_simple Class Reference

Sets custom printer overrides for Q1/P1 and Q2/P2. The remaining queues will use the default printer.

Inheritance diagram for cl_scb_test_io_std_dump_simple:



Collaboration diagram for cl_scb_test_io_std_dump_simple:



13.22.1 Detailed Description

Sets custom printer overrides for Q1/P1 and Q2/P2. The remaining queues will use the default printer.

Definition at line 6 of file cl_scb_test_io_std_dump_custom_printer.svh.

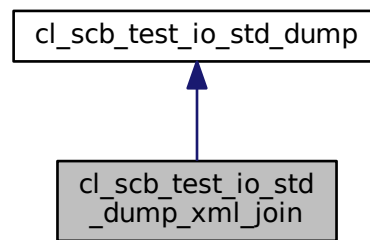
The documentation for this class was generated from the following file:

- cl_scb_test_io_std_dump_custom_printer.svh

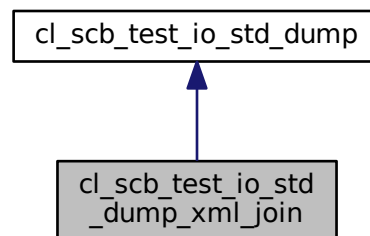
13.23 cl_scb_test_io_std_dump_xml_join Class Reference

Uses XML/join printing to generate a single XML file once the test is finished.

Inheritance diagram for cl_scb_test_io_std_dump_xml_join:



Collaboration diagram for cl_scb_test_io_std_dump_xml_join:



13.23.1 Detailed Description

Uses XML/join printing to generate a single XML file once the test is finished.

Definition at line 158 of file cl_scb_test_io_std_dump_custom_printer.svh.

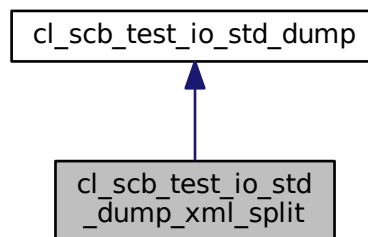
The documentation for this class was generated from the following file:

- cl_scb_test_io_std_dump_custom_printer.svh

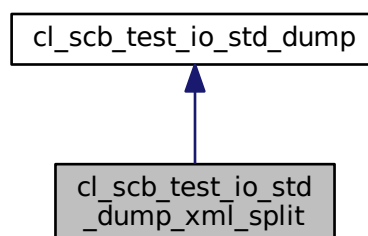
13.24 cl_scb_test_io_std_dump_xml_split Class Reference

Uses XML/split printing to generate multiple XML files once the test is finished.

Inheritance diagram for cl_scb_test_io_std_dump_xml_split:



Collaboration diagram for cl_scb_test_io_std_dump_xml_split:



13.24.1 Detailed Description

Uses XML/split printing to generate multiple XML files once the test is finished.

Definition at line 123 of file cl_scb_test_io_std_dump_custom_printer.svh.

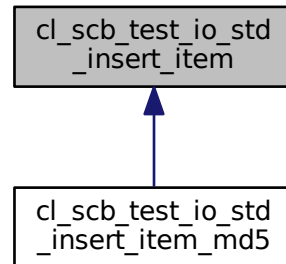
The documentation for this class was generated from the following file:

- cl_scb_test_io_std_dump_custom_printer.svh

13.25 cl_scb_test_io_std_insert_item Class Reference

IO test that validates the behavior of [cl_syoscb_queue_base::insert_item](#).

Inheritance diagram for cl_scb_test_io_std_insert_item:



13.25.1 Detailed Description

IO test that validates the behavior of [cl_syoscb_queue_base::insert_item](#).

Definition at line 2 of file cl_scb_test_io_std_insert_item.svh.

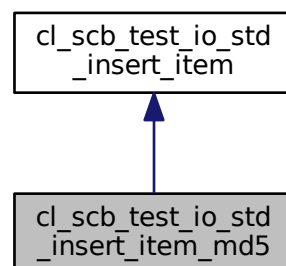
The documentation for this class was generated from the following file:

- cl_scb_test_io_std_insert_item.svh

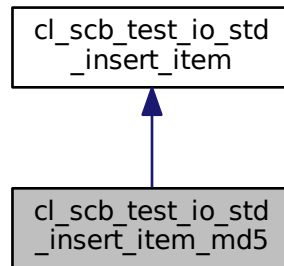
13.26 cl_scb_test_io_std_insert_item_md5 Class Reference

IO test that validates the behavior of [cl_syoscb_queue_base::insert_item](#) when using MD5 queues.

Inheritance diagram for cl_scb_test_io_std_insert_item_md5:



Collaboration diagram for `cl_scb_test_io_std_insert_item_md5`:



13.26.1 Detailed Description

IO test that validates the behavior of `cl_syoscb_queue_base::insert_item` when using MD5 queues.

Definition at line 3 of file `cl_scb_test_io_std_insert_item_md5.svh`.

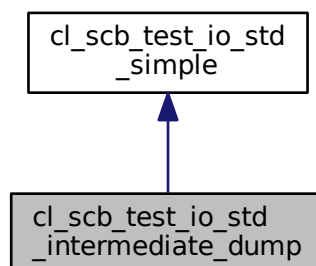
The documentation for this class was generated from the following file:

- `cl_scb_test_io_std_insert_item_md5.svh`

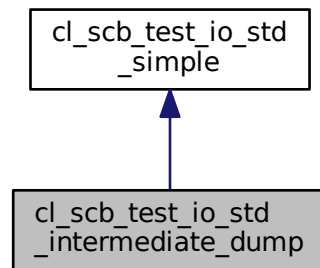
13.27 `cl_scb_test_io_std_intermediate_dump` Class Reference

Tests the intermediate queue stat printout, see `cl_syoscb_cfg::queue_stat_interval`.

Inheritance diagram for `cl_scb_test_io_std_intermediate_dump`:



Collaboration diagram for cl_scb_test_io_std_intermediate_dump:



13.27.1 Detailed Description

Tests the intermediate queue stat printout, see [cl_syoscb_cfg::queue_stat_interval](#).

Definition at line 3 of file `cl_scb_test_io_std_intermediate_dump.svh`.

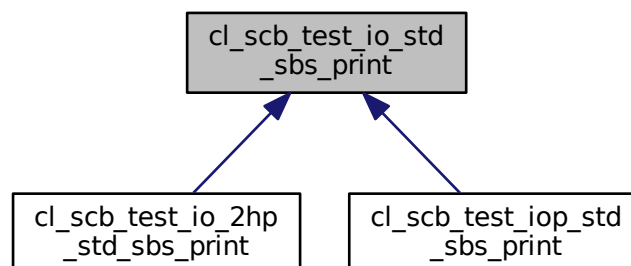
The documentation for this class was generated from the following file:

- `cl_scb_test_io_std_intermediate_dump.svh`

13.28 cl_scb_test_io_std_sbs_print Class Reference

Shows a number of different ways that the side-by-side miscompare table can be used.

Inheritance diagram for cl_scb_test_io_std_sbs_print:



13.28.1 Detailed Description

Shows a number of different ways that the side-by-side miscompare table can be used.

Definition at line 5 of file `cl_scb_test_io_std_sbs_print.svh`.

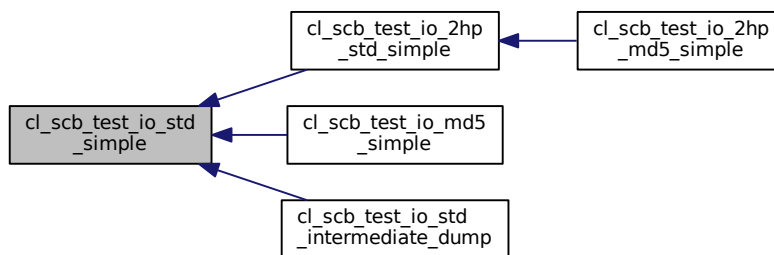
The documentation for this class was generated from the following file:

- `cl_scb_test_io_std_sbs_print.svh`

13.29 `cl_scb_test_io_std_simple` Class Reference

Simple IO compare test using the function based API.

Inheritance diagram for `cl_scb_test_io_std_simple`:



13.29.1 Detailed Description

Simple IO compare test using the function based API.

Definition at line 3 of file `cl_scb_test_io_std_simple.svh`.

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

- `cl_scb_test_io_std_simple.svh`

13.30 `cl_scb_test_io_std_simple_mutexed` Class Reference

Simple IO compare test using the function based API and mutexed `add_item` calls.

Inherits `cl_scb_test_single_scb`.

13.30.1 Detailed Description

Simple IO compare test using the function based API and mutexed add_item calls.

Definition at line 6 of file cl_scb_test_io_std_simple_mutexed.svh.

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

- cl_scb_test_io_std_simple_mutexed.svh

13.31 cl_scb_test_io_std_simple_real Class Reference

Simple IO compare test on real values using the function based API.

Inherits cl_scb_test_single_scb.

13.31.1 Detailed Description

Simple IO compare test on real values using the function based API.

Definition at line 2 of file cl_scb_test_io_std_simple_real.svh.

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

- cl_scb_test_io_std_simple_real.svh

13.32 cl_scb_test_io_std_tlm_gp_test Class Reference

IO comparison test to ensure that the SYOSIL TLM GP comparison workaround works as expected.

Inherits cl_scb_test_single_scb.

13.32.1 Detailed Description

IO comparison test to ensure that the SYOSIL TLM GP comparison workaround works as expected.

See [cl_syoscb_item](#) for a description as to why this workaround is necessary

Definition at line 3 of file cl_scb_test_io_std_tlm_gp_test.svh.

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

- cl_scb_test_io_std_tlm_gp_test.svh

13.33 cl_scb_test_io_std_tlm_mutexed Class Reference

Simple IO test using the TLM ssetup and mutexed add_item.

Inherits cl_scb_test_single_scb.

13.33.1 Detailed Description

Simple IO test using the TLM ssetup and mutexed add_item.

Definition at line 2 of file cl_scb_test_io_std_tlm_mutexed.svh.

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

- cl_scb_test_io_std_tlm_mutexed.svh

13.34 cl_scb_test_iop_md5_simple Class Reference

Simple IOP compare test using the function based API and MD5 queues.

Inherits cl_scb_test_iop_std_simple.

13.34.1 Detailed Description

Simple IOP compare test using the function based API and MD5 queues.

Definition at line 3 of file cl_scb_test_iop_md5_simple.svh.

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

- cl_scb_test_iop_md5_simple.svh

13.35 cl_scb_test_iop_std_msw Class Reference

This test ensures that IOP compares correctly search through the primary queue, comparing not only the first item but also subsequent items for matches.

Inherits cl_scb_test_single_scb.

13.35.1 Detailed Description

This test ensures that IOP compares correctly search through the primary queue, comparing not only the first item but also subsequent items for matches.

Definition at line 3 of file cl_scb_test_iop_std_msw.svh.

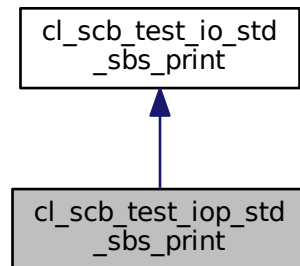
The documentation for this class was generated from the following file:

- cl_scb_test_iop_std_msw.svh

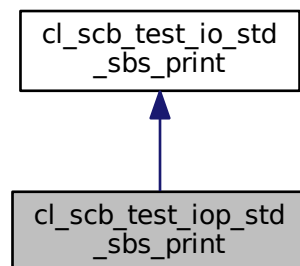
13.36 cl_scb_test_iop_std_sbs_print Class Reference

A functional copy of the test seen in [cl_scb_test_io_std_sbs_print](#), shows that miscompare table work for IOP compare.

Inheritance diagram for cl_scb_test_iop_std_sbs_print:



Collaboration diagram for cl_scb_test_iop_std_sbs_print:



13.36.1 Detailed Description

A functional copy of the test seen in [cl_scb_test_io_std_sbs_print](#), shows that miscompare table work for IOP compare.

Definition at line 2 of file cl_scb_test_iop_std_sbs_print.svh.

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

- cl_scb_test_iop_std_sbs_print.svh

13.37 cl_scb_test_iterator_correctness Class Reference

Test to ensure that multiple iterators on the same queue won't deadlock and are performing correctly.

Inherits cl_scb_test_single_scb.

13.37.1 Detailed Description

Test to ensure that multiple iterators on the same queue won't deadlock and are performing correctly.

At the end, validates that the correct items have been removed/inserted

Definition at line 28 of file cl_scb_test_iterator_correctness.svh.

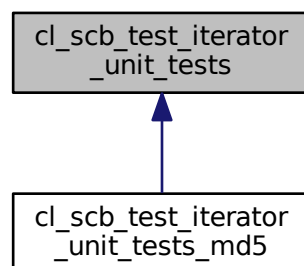
The documentation for this class was generated from the following file:

- cl_scb_test_iterator_correctness.svh

13.38 cl_scb_test_iterator_unit_tests Class Reference

Test containing a series of unit tests to ensure that all iterators conform to spec.

Inheritance diagram for cl_scb_test_iterator_unit_tests:



Public Member Functions

- task [check_next](#) ()
Checks whether the [cl_syoscb_queue_iterator_base::next](#) method correctly moves through the queue. When called, the idx should increment and it should return 1'b1.
- task [check_prev](#) ()
Checks whether the [cl_syoscb_queue_iterator_base::previous](#) method correctly moves through the queue.
- task [check_first](#) ()
Checks whether the [cl_syoscb_queue_iterator_base::first](#) method correctly moves through the queue.
- task [check_last](#) ()
Checks whether the [cl_syoscb_queue_iterator_base::last](#) method correctly moves through the queue.
- task [check_set_queue](#) ()
Checks whether the [cl_syoscb_queue_iterator_base::set_queue](#) method correctly sets the queue associated with an iterator.
- task [check_names](#) ()
Checks whether [cl_syoscb_queue_base::get_iterator](#) and [cl_syoscb_queue_base::create_iterator](#) correctly create and retrieve named iterators.
- task [check_flush](#) ()
When a queue is flushed, all associated iterators should be reset such that [has_next](#)/[has_previous](#) both return 0.

13.38.1 Detailed Description

Test containing a series of unit tests to ensure that all iterators conform to spec.

Definition at line 2 of file `cl_scb_test_iterator_unit_tests.svh`.

13.38.2 Member Function Documentation

13.38.2.1 [check_first\(\)](#)

```
task cl_scb_test_iterator_unit_tests::check_first ( )
```

Checks whether the [cl_syoscb_queue_iterator_base::first](#) method correctly moves through the queue.

When called, the idx should become 0 and it should return 1'b1. It should then also point to the first item in the queue. When called while already pointing to the first element of the queue, behavior should be the same. When called on an empty queue, should return 1'b0.

Definition at line 143 of file `cl_scb_test_iterator_unit_tests.svh`.

References [cl_syoscb_queue_base::create_iterator\(\)](#), [cl_syoscb_queue_base::delete_iterator\(\)](#), [cl_syoscb_queue_iterator_base::first\(\)](#), [cl_syoscb_queue_iterator_base::next\(\)](#), and [cl_syoscb_queue_iterator_base::previous_index\(\)](#).

13.38.2.2 `check_last()`

```
task cl_scb_test_iterator_unit_tests::check_last ( )
```

Checks whether the [cl_syoscb_queue_iterator_base::last](#) method correctly moves through the queue.

When called, the idx should become queue.size()-1 and it should return 1'b1. It should then also point to the final item in the queue. When called while already pointing to the final element of the queue, behavior should be the same. When called on an empty queue, should return 1'b0.

Definition at line 190 of file cl_scb_test_iterator_unit_tests.svh.

References [cl_syoscb_queue_base::create_iterator\(\)](#), [cl_syoscb_queue_base::delete_iterator\(\)](#), [cl_syoscb_queue_base::get_size\(\)](#), [cl_syoscb_queue_iterator_base::last\(\)](#), and [cl_syoscb_queue_iterator_base::next_index\(\)](#).

13.38.2.3 `check_names()`

```
task cl_scb_test_iterator_unit_tests::check_names ( )
```

Checks whether [cl_syoscb_queue_base::get_iterator](#) and [cl_syoscb_queue_base::create_iterator](#) correctly create and retrieve named iterators.

It should not be possible to create two iterators with the same name, and it should not be possible to retrieve an iterator if the name does not match any iterators.

Definition at line 292 of file cl_scb_test_iterator_unit_tests.svh.

References [cl_syoscb_queue_base::create_iterator\(\)](#), and [cl_syoscb_queue_base::delete_iterator\(\)](#).

13.38.2.4 `check_next()`

```
task cl_scb_test_iterator_unit_tests::check_next ( )
```

Checks whether the [cl_syoscb_queue_iterator_base::next](#) method correctly moves through the queue. When called, the idx should increment and it should return 1'b1.

It should then also point to the next item in the queue. When called while already pointing to the last element of the queue, it should generate an out-of-bounds message and return 1'b0.

Definition at line 55 of file cl_scb_test_iterator_unit_tests.svh.

References [cl_syoscb_queue_base::create_iterator\(\)](#), [cl_syoscb_queue_base::delete_iterator\(\)](#), [cl_syoscb_queue_base::get_size\(\)](#), [cl_syoscb_queue_iterator_base::has_next\(\)](#), [cl_syoscb_queue_iterator_base::next_index\(\)](#), and [cl_syoscb_queue_iterator_base::previous_index\(\)](#).

13.38.2.5 check_prev()

```
task cl_scb_test_iterator_unit_tests::check_prev ( )
```

Checks whether the [cl_syoscb_queue_iterator_base::previous](#) method correctly moves through the queue.

When called, the idx should decrement and it should return 1'b1. It should then also point to the previous item in the queue. When called while already pointing to the first element of the queue, it should generate an out-of-bounds message and return 1'b0. When called on an empty queue, it should return 1'b0.

Definition at line 96 of file cl_scb_test_iterator_unit_tests.svh.

References [cl_syoscb_queue_base::create_iterator\(\)](#), [cl_syoscb_queue_base::delete_iterator\(\)](#), [cl_syoscb_queue_iterator_base::has_next\(\)](#), [cl_syoscb_queue_iterator_base::has_previous\(\)](#), [cl_syoscb_queue_iterator_base::next\(\)](#), [cl_syoscb_queue_iterator_base::next_index\(\)](#), and [cl_syoscb_queue_iterator_base::previous_index\(\)](#).

13.38.2.6 check_set_queue()

```
task cl_scb_test_iterator_unit_tests::check_set_queue ( )
```

Checks whether the [cl_syoscb_queue_iterator_base::set_queue](#) method correctly sets the queue associated with an iterator.

When called with null as argument, should return 1'b0. When called and the iterator already has an owner associated, should raise a UVM_ERROR When called and the new owner is not of the right queue type, should raise a UVM_ERROR When called and the iterator does not have an owner associated, should return 1'b1 and set the queue as owner.

Definition at line 226 of file cl_scb_test_iterator_unit_tests.svh.

References [cl_syoscb_queue_base::create_iterator\(\)](#), [cl_syoscb_queue_base::delete_iterator\(\)](#), [cl_syoscb_queue_iterator_std::set_queue\(\)](#), [cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH >::set_queue\(\)](#), and [cl_syoscb_queue_iterator_base::set_queue\(\)](#).

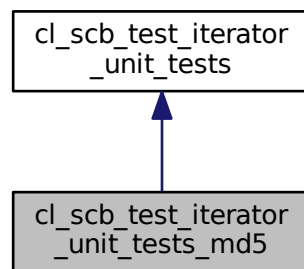
The documentation for this class was generated from the following file:

- cl_scb_test_iterator_unit_tests.svh

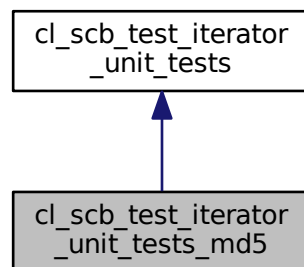
13.39 cl_scb_test_iterator_unit_tests_md5 Class Reference

Test that md5-iterators using [cl_syoscb_cfg::ordered_next](#) conform to spec.

Inheritance diagram for cl_scb_test_iterator_unit_tests_md5:



Collaboration diagram for cl_scb_test_iterator_unit_tests_md5:



Additional Inherited Members

13.39.1 Detailed Description

Test that md5-iterators using [cl_syoscb_cfg::ordered_next](#) conform to spec.

Definition at line 2 of file cl_scb_test_iterator_unit_tests_md5.svh.

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

- cl_scb_test_iterator_unit_tests_md5.svh

13.40 cl_scb_test_md5 Class Reference

Test which verifies that the md5 hash implementation works correctly.

Inherits cl_scb_test_single_scb.

13.40.1 Detailed Description

Test which verifies that the md5 hash implementation works correctly.

Definition at line 2 of file cl_scb_test_md5.svh.

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

- cl_scb_test_md5.svh

13.41 cl_scb_test_md5_hash_collisions Class Reference

Test to verify that comparisons still work correctly on hash items with multiple entries where hash collisions may have occurred.

Inherits cl_scb_test_single_scb.

13.41.1 Detailed Description

Test to verify that comparisons still work correctly on hash items with multiple entries where hash collisions may have occurred.

Definition at line 3 of file cl_scb_test_md5_hash_collisions.svh.

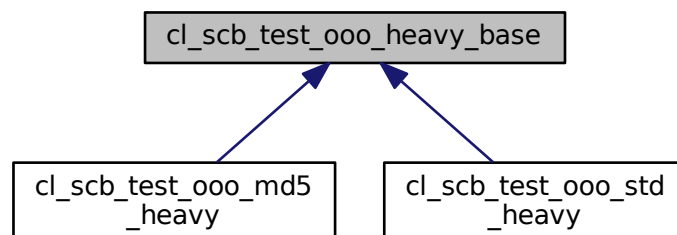
The documentation for this class was generated from the following file:

- cl_scb_test_md5_hash_collisions.svh

13.42 cl_scb_test_ooo_heavy_base Class Reference

Heavy OOO compare test using the function based API.

Inheritance diagram for cl_scb_test_ooo_heavy_base:



13.42.1 Detailed Description

Heavy OOO compare test using the function based API.

Definition at line 11 of file cl_scb_test_ooo_heavy_base.svh.

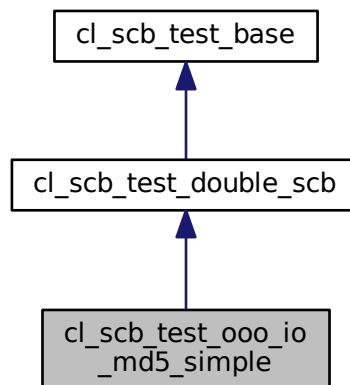
The documentation for this class was generated from the following file:

- cl_scb_test_ooo_heavy_base.svh

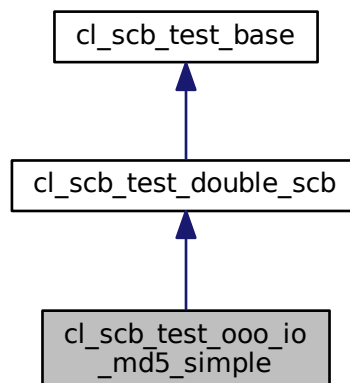
13.43 cl_scb_test_ooo_io_md5_simple Class Reference

Simple test with two SCBs with different compares, both using MD5 queues.

Inheritance diagram for cl_scb_test_ooo_io_md5_simple:



Collaboration diagram for cl_scb_test_ooo_io_md5_simple:



13.43.1 Detailed Description

Simple test with two SCBs with different compares, both using MD5 queues.

Definition at line 3 of file cl_scb_test_ooo_io_md5_simple.svh.

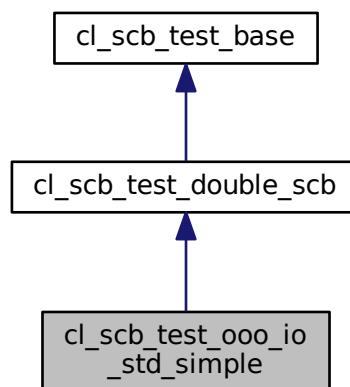
The documentation for this class was generated from the following file:

- cl_scb_test_ooo_io_md5_simple.svh

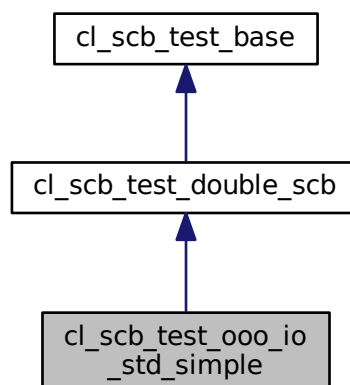
13.44 cl_scb_test_ooo_io_std_simple Class Reference

Simple test with two SCBs with different compares and standard queues.

Inheritance diagram for cl_scb_test_ooo_io_std_simple:



Collaboration diagram for cl_scb_test_ooo_io_std_simple:



13.44.1 Detailed Description

Simple test with two SCBs with different compares and standard queues.

Definition at line 3 of file `cl_scb_test_ooo_io_std_simple.svh`.

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

- `cl_scb_test_ooo_io_std_simple.svh`

13.45 `cl_scb_test_ooo_md5_duplets` Class Reference

Duplets OOO compare test using the function based API.

Inherits `cl_scb_test_single_scb`.

13.45.1 Detailed Description

Duplets OOO compare test using the function based API.

Definition at line 3 of file `cl_scb_test_ooo_md5_duplets.svh`.

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

- `cl_scb_test_ooo_md5_duplets.svh`

13.46 `cl_scb_test_ooo_md5_gp` Class Reference

Simple OOO compare test for TLM generic payload using the function based API.

Inherits `cl_scb_test_single_scb`.

13.46.1 Detailed Description

Simple OOO compare test for TLM generic payload using the function based API.

Definition at line 3 of file `cl_scb_test_ooo_md5_gp.svh`.

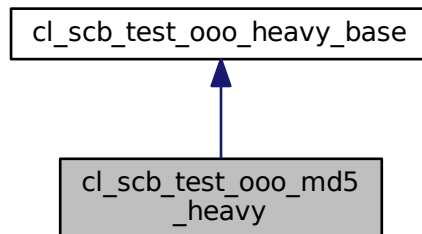
The documentation for this class was generated from the following file:

- `cl_scb_test_ooo_md5_gp.svh`

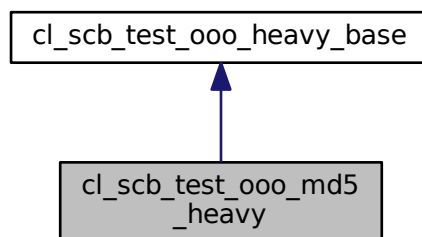
13.47 cl_scb_test_ooo_md5_heavy Class Reference

Heavy OOO compare test using the function based API and MD5 queues.

Inheritance diagram for cl_scb_test_ooo_md5_heavy:



Collaboration diagram for cl_scb_test_ooo_md5_heavy:



13.47.1 Detailed Description

Heavy OOO compare test using the function based API and MD5 queues.

Definition at line 2 of file cl_scb_test_ooo_md5_heavy.svh.

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

- cl_scb_test_ooo_md5_heavy.svh

13.48 cl_scb_test_ooo_md5_simple Class Reference

Simple OOO compare test using the function based API and MD5 queues.

Inherits cl_scb_test_single_scb.

13.48.1 Detailed Description

Simple OOO compare test using the function based API and MD5 queues.

Definition at line 3 of file `cl_scb_test_ooo_md5_simple.svh`.

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

- `cl_scb_test_ooo_md5_simple.svh`

13.49 `cl_scb_test_ooo_md5_tlm` Class Reference

Simple OOO compare test using the TLM based API and MD5 queues.

Inherits `cl_scb_test_single_scb`.

13.49.1 Detailed Description

Simple OOO compare test using the TLM based API and MD5 queues.

Definition at line 2 of file `cl_scb_test_ooo_md5_tlm.svh`.

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

- `cl_scb_test_ooo_md5_tlm.svh`

13.50 `cl_scb_test_ooo_md5_validate` Class Reference

Test to ensure that config knob `cl_syoscb_cfg::hash_compare_check` correctly controls MD5 validation behavior.

Inherits `cl_scb_test_single_scb`.

13.50.1 Detailed Description

Test to ensure that config knob `cl_syoscb_cfg::hash_compare_check` correctly controls MD5 validation behavior.

Validation will verify whether items with the same hash match, or whether no match found really is a no-match

Definition at line 4 of file `cl_scb_test_ooo_md5_validate.svh`.

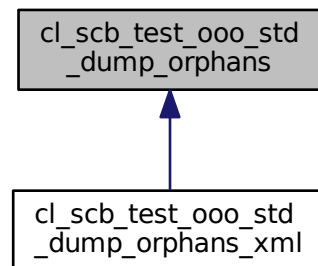
The documentation for this class was generated from the following file:

- `cl_scb_test_ooo_md5_validate.svh`

13.51 cl_scb_test_ooo_std_dump_orphans Class Reference

This test uses the dump_orphans_to_files configuration knob.

Inheritance diagram for cl_scb_test_ooo_std_dump_orphans:



13.51.1 Detailed Description

This test uses the dump_orphans_to_files configuration knob.

Definition at line 2 of file cl_scb_test_ooo_std_dump_orphans.svh.

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

- cl_scb_test_ooo_std_dump_orphans.svh

13.52 cl_scb_test_ooo_std_dump_orphans_abort Class Reference

Tests queue and orphan dumping when an error occurs mid-simulation This test fails on purpose, and is therefore not included in the regression tests.

Inherits cl_scb_test_single_scb.

13.52.1 Detailed Description

Tests queue and orphan dumping when an error occurs mid-simulation This test fails on purpose, and is therefore not included in the regression tests.

Definition at line 3 of file cl_scb_test_ooo_std_dump_orphans_abort.svh.

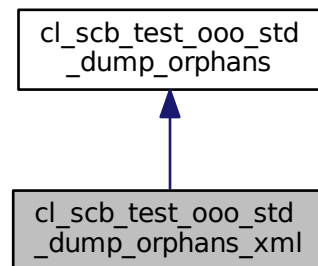
The documentation for this class was generated from the following file:

- cl_scb_test_ooo_std_dump_orphans_abort.svh

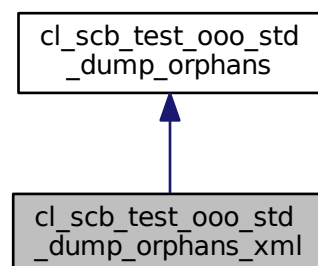
13.53 cl_scb_test_ooo_std_dump_orphans_xml Class Reference

Dumping orphans to files using XML printout.

Inheritance diagram for cl_scb_test_ooo_std_dump_orphans_xml:



Collaboration diagram for cl_scb_test_ooo_std_dump_orphans_xml:



13.53.1 Detailed Description

Dumping orphans to files using XML printout.

Definition at line 3 of file cl_scb_test_ooo_std_dump_orphans_xml.svh.

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

- cl_scb_test_ooo_std_dump_orphans_xml.svh

13.54 cl_scb_test_ooo_std_gp Class Reference

Simple OOO compare test for TLM generic payload using the function based API.

Inherits cl_scb_test_single_scb.

13.54.1 Detailed Description

Simple OOO compare test for TLM generic payload using the function based API.

Definition at line 3 of file cl_scb_test_ooo_std_gp.svh.

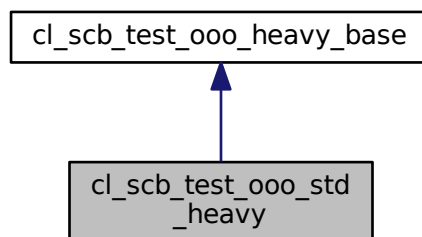
The documentation for this class was generated from the following file:

- cl_scb_test_ooo_std_gp.svh

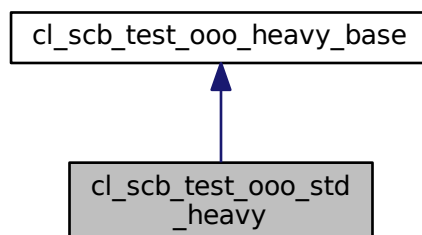
13.55 cl_scb_test_ooo_std_heavy Class Reference

Heavy OOO compare test using the function based API and a standard queue.

Inheritance diagram for cl_scb_test_ooo_std_heavy:



Collaboration diagram for cl_scb_test_ooo_std_heavy:



13.55.1 Detailed Description

Heavy OOO compare test using the function based API and a standard queue.

Definition at line 2 of file `cl_scb_test_ooo_std_heavy.svh`.

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

- `cl_scb_test_ooo_std_heavy.svh`

13.56 `cl_scb_test_ooo_std_max_search_window` Class Reference

Simple OOO compare test using the function based API and the `max_search_window` knob to control OOO compare searches.

Inherits `cl_scb_test_single_scb`.

13.56.1 Detailed Description

Simple OOO compare test using the function based API and the `max_search_window` knob to control OOO compare searches.

Definition at line 3 of file `cl_scb_test_ooo_std_max_search_window.svh`.

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

- `cl_scb_test_ooo_std_max_search_window.svh`

13.57 `cl_scb_test_ooo_std_primary_multiple` Class Reference

OOO compare test for ensuring that multiple items in the primary queue are checked against the secondary queue.

Inherits `cl_scb_test_single_scb`.

13.57.1 Detailed Description

OOO compare test for ensuring that multiple items in the primary queue are checked against the secondary queue.

If eg. `cl_syoscb_cfg::max_search_window = 5`, the first 5 items in the primary queue shall be compared against the first 5 elements in all secondary queues.

Definition at line 4 of file `cl_scb_test_ooo_std_primary_multiple.svh`.

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

- `cl_scb_test_ooo_std_primary_multiple.svh`

13.58 cl_scb_test_ooo_std_simple Class Reference

Simple OOO compare test using the function based API.

Inherits cl_scb_test_single_scb.

13.58.1 Detailed Description

Simple OOO compare test using the function based API.

Definition at line 3 of file cl_scb_test_ooo_std_simple.svh.

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

- cl_scb_test_ooo_std_simple.svh

13.59 cl_scb_test_ooo_std_tlm Class Reference

Simple OOO compare test using the TLM based API.

Inherits cl_scb_test_single_scb.

13.59.1 Detailed Description

Simple OOO compare test using the TLM based API.

Definition at line 2 of file cl_scb_test_ooo_std_tlm.svh.

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

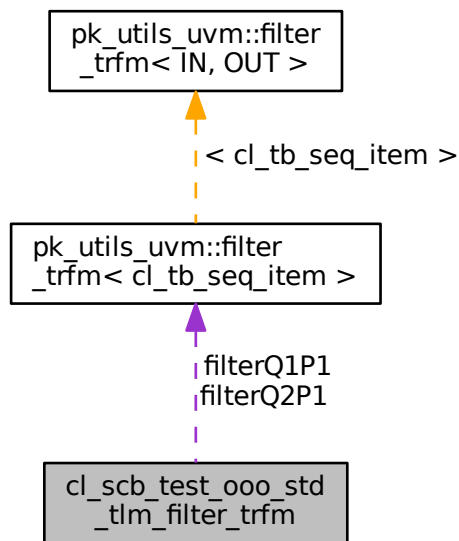
- cl_scb_test_ooo_std_tlm.svh

13.60 cl_scb_test_ooo_std_tlm_filter_trfm Class Reference

Simple OOO compare test using the TLM based API and filter transforms.

Inherits cl_scb_test_single_scb.

Collaboration diagram for cl_scb_test_ooo_std_tlm_filter_trfm:



13.60.1 Detailed Description

Simple OOO compare test using the TLM based API and filter transforms.

Definition at line 2 of file cl_scb_test_ooo_std_tlm_filter_trfm.svh.

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

- cl_scb_test_ooo_std_tlm_filter_trfm.svh

13.61 cl_scb_test_ooo_std_trigger_greed Class Reference

OOO Compare test for validating that OOO compares respect the current greed level.

Inherits cl_scb_test_single_scb.

13.61.1 Detailed Description

OOO Compare test for validating that OOO compares respect the current greed level.

Definition at line 2 of file cl_scb_test_ooo_std_trigger_greed.svh.

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

- cl_scb_test_ooo_std_trigger_greed.svh

13.62 cl_scb_test_queue_find_vs_search Class Reference

A test comparing the performance of using iterators vs using .find_first on a SV queue.

Inherits cl_scb_test_single_scb.

Public Member Functions

- bit [compare_items](#) (uvm_object primary_item, uvm_object sec_item)
Compares two sequence items using an implicit comparer Returns 1'b1 if the comparison is true, 1'b0 otherwise.

13.62.1 Detailed Description

A test comparing the performance of using iterators vs using .find_first on a SV queue.

Definition at line 19 of file cl_scb_test_queue_find_vs_search.svh.

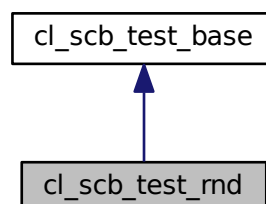
The documentation for this class was generated from the following file:

- cl_scb_test_queue_find_vs_search.svh

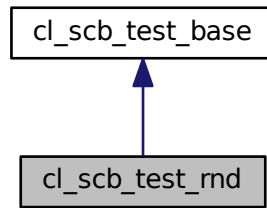
13.63 cl_scb_test_rnd Class Reference

Random test to hit all the coverage holes which are not covered by tests derived from [cl_scb_test_double_scb](#).

Inheritance diagram for cl_scb_test_rnd:



Collaboration diagram for `cl_scb_test_rnd`:



13.63.1 Detailed Description

Random test to hit all the coverage holes which are not covered by tests derived from [cl_scb_test_double_scb](#).

Definition at line 6 of file `cl_scb_test_rnd.svh`.

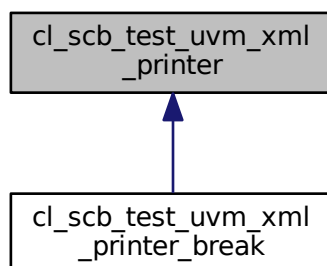
The documentation for this class was generated from the following file:

- `cl_scb_test_rnd.svh`

13.64 `cl_scb_test_uvm_xml_printer` Class Reference

A test which can be used to generate an XML printout for verifying the [uvm_xml_printer](#).

Inheritance diagram for `cl_scb_test_uvm_xml_printer`:



13.64.1 Detailed Description

A test which can be used to generate an XML printout for verifying the [uvm_xml_printer](#).

Definition at line 5 of file `cl_scb_test_uvm_xml_printer.svh`.

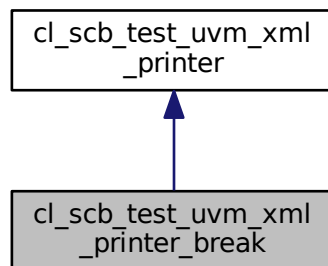
The documentation for this class was generated from the following file:

- `cl_scb_test_uvm_xml_printer.svh`

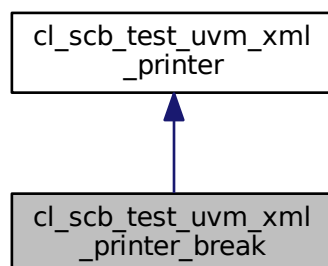
13.65 cl_scb_test_uvm_xml_printer_break Class Reference

Tests whether the [uvm_xml_printer](#) correctly outputs a warning when it is used on a non-cl_syoscb_item sequence item.

Inheritance diagram for `cl_scb_test_uvm_xml_printer_break`:



Collaboration diagram for `cl_scb_test_uvm_xml_printer_break`:



13.65.1 Detailed Description

Tests whether the [uvm_xml_printer](#) correctly outputs a warning when it is used on a non-cl_syoscb_item sequence item.

Definition at line 90 of file cl_scb_test_uvm_xml_printer.svh.

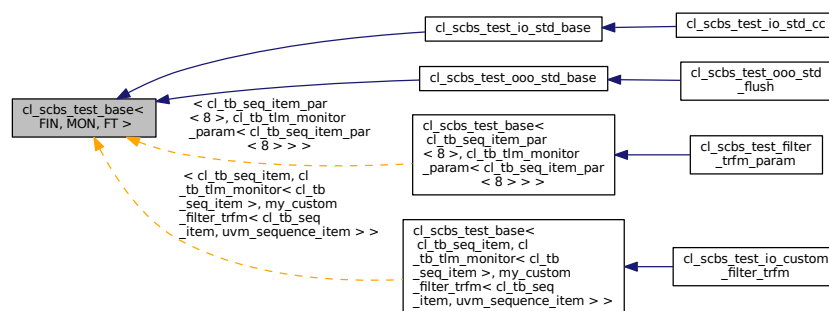
The documentation for this class was generated from the following file:

- cl_scb_test_uvm_xml_printer.svh

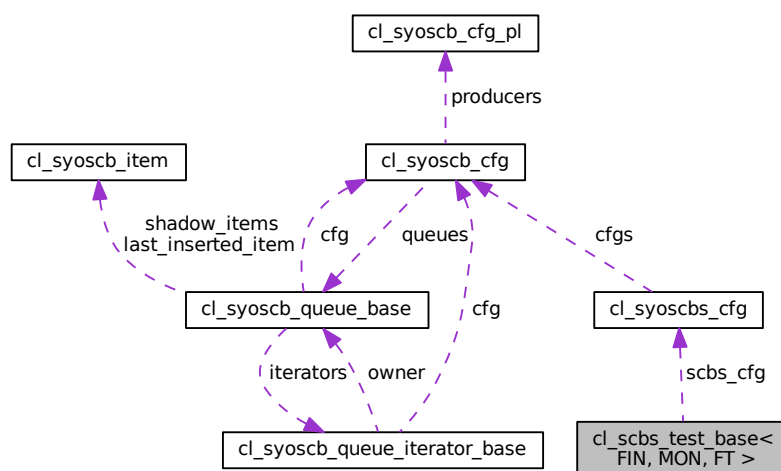
13.66 cl_scbs_test_base< FIN, MON, FT > Class Template Reference

Base class for all SCBs tests.

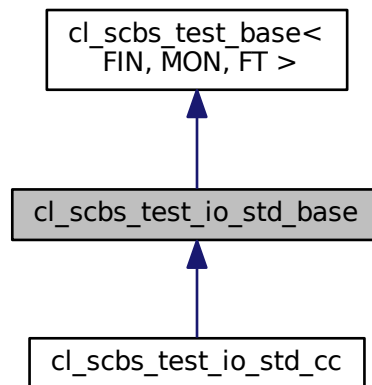
Inheritance diagram for cl_scbs_test_base< FIN, MON, FT >:



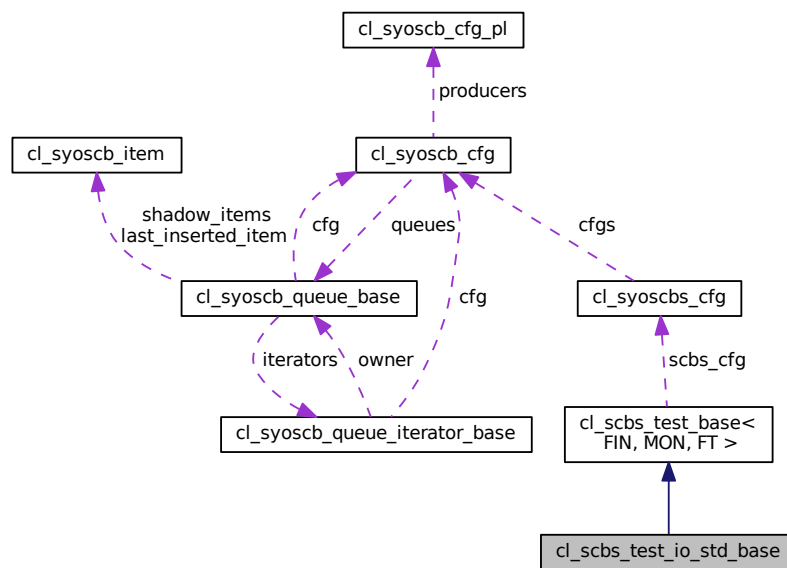
Collaboration diagram for cl_scbs_test_base< FIN, MON, FT >:



Inheritance diagram for cl_scbs_test_io_std_base:



Collaboration diagram for cl_scbs_test_io_std_base:



13.69.1 Detailed Description

Simple IO compare with standard queues using TLM based API.

Definition at line 2 of file `cl_scbs_test_io_std_base.svh`.

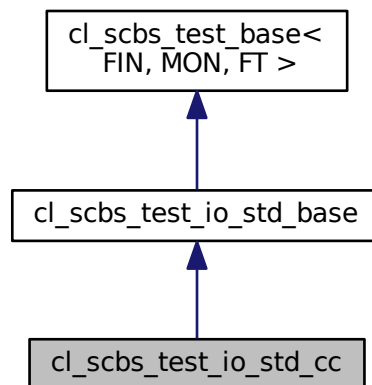
The documentation for this class was generated from the following file:

- `cl_scbs_test_io_std_base.svh`

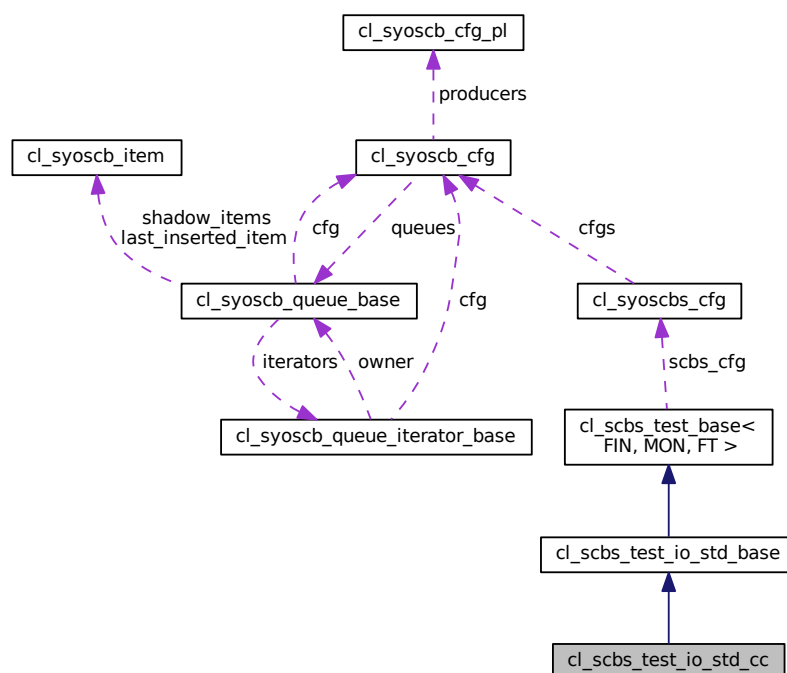
13.70 cl_scbs_test_io_std_cc Class Reference

Simple IO compare with STD queue test. Testing the [cl_syoscbs](#) class.

Inheritance diagram for `cl_scbs_test_io_std_cc`:



Collaboration diagram for `cl_scbs_test_io_std_cc`:



13.70.1 Detailed Description

Simple IO compare with STD queue test. Testing the [cl_syoscbs](#) class.

Definition at line 4 of file cl_scbs_test_io_std_cc.svh.

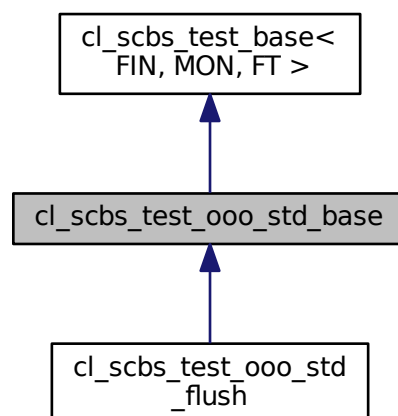
The documentation for this class was generated from the following file:

- cl_scbs_test_io_std_cc.svh

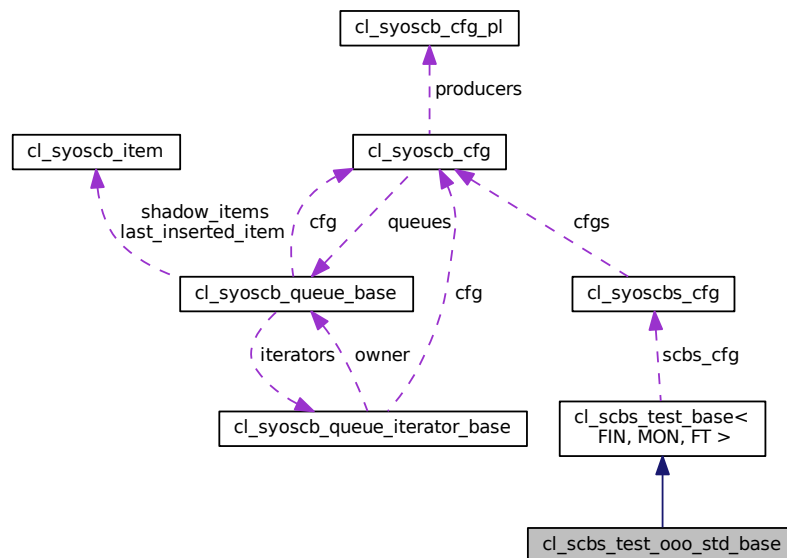
13.71 cl_scbs_test_ooo_std_base Class Reference

Simple OOO compare with STD queue test. Testing the [cl_syoscbs](#) class.

Inheritance diagram for cl_scbs_test_ooo_std_base:



Collaboration diagram for `cl_scbs_test_ooo_std_base`:



13.71.1 Detailed Description

Simple OOO compare with STD queue test. Testing the `cl_syoscbcs` class.

Definition at line 4 of file `cl_scbs_test_ooo_std_base.svh`.

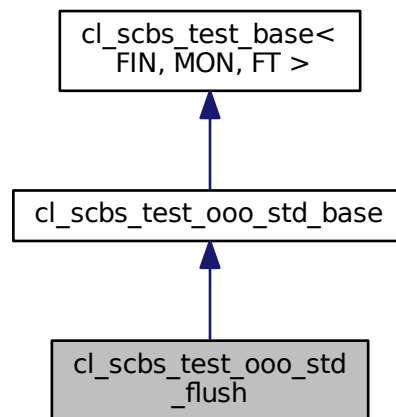
The documentation for this class was generated from the following file:

- `cl_scbs_test_ooo_std_base.svh`

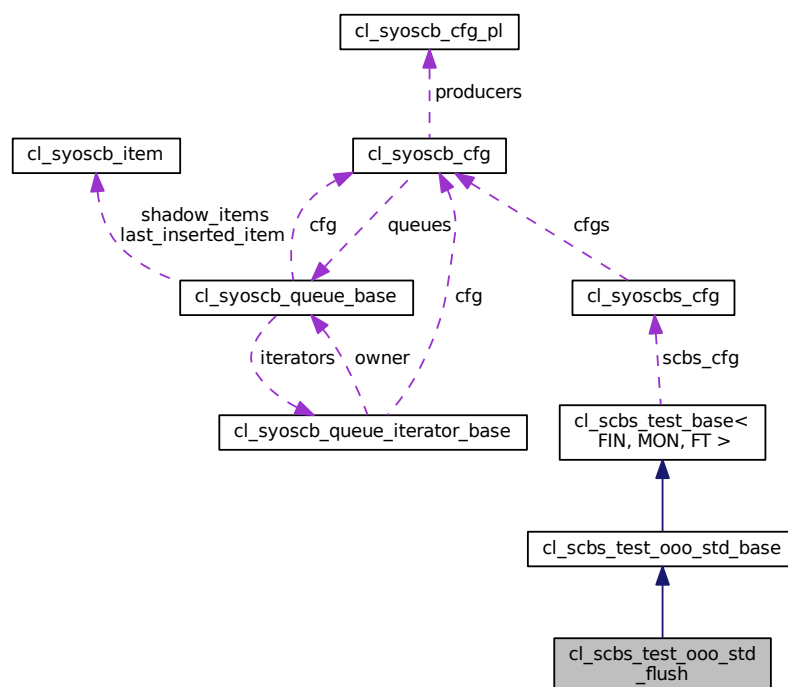
13.72 `cl_scbs_test_ooo_std_flush` Class Reference

Simple OOO compare with STD queue test which inserts additional random items, requiring a flush at the end to pass the test.

Inheritance diagram for cl_scbs_test_ooo_std_flush:



Collaboration diagram for cl_scbs_test_ooo_std_flush:



13.72.1 Detailed Description

Simple OOO compare with STD queue test which inserts additional random items, requiring a flush at the end to pass the test.

Definition at line 5 of file cl_scbs_test_ooo_std_flush.svh.

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

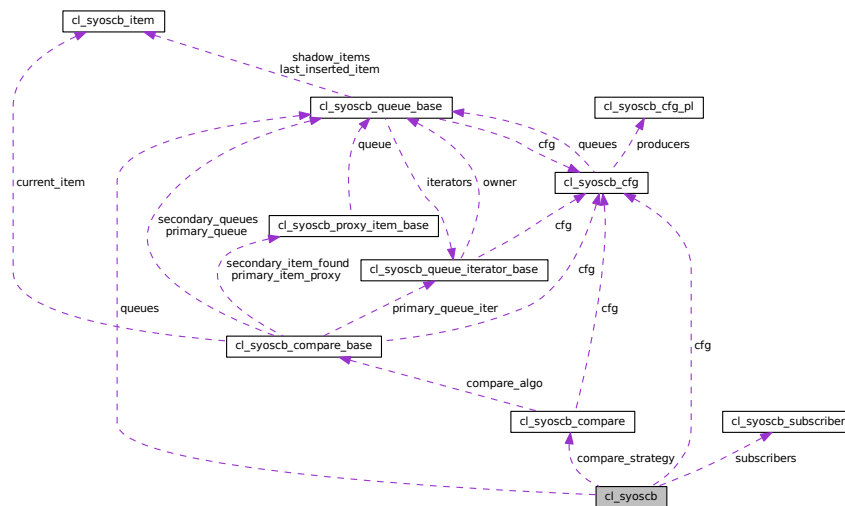
- cl_scbs_test_ooo_std_flush.svh

13.73 cl_syoscb Class Reference

Top level class implementing the root of the SyoSil UVM scoreboard.

Inherits uvm_scoreboard, and uvm_scoreboard.

Collaboration diagram for cl_syoscb:



Public Member Functions

- void [build_phase](#) (uvm_phase phase)
UVM build phase.
- void [end_of_elaboration_phase](#) (uvm_phase phase)
UVM end of elaboration phase.
- void [check_phase](#) (uvm_phase phase)
UVM check phase.
- void [report_phase](#) (uvm_phase phase)
UVM report phase. Prints the status of the scoreboard instance.
- void [final_phase](#) (uvm_phase phase)
UVM final phase. Prints in the file called dump.txt the information about the shadow queue of all the queues.
- virtual void [add_item](#) (string queue_name, string producer, uvm_sequence_item item)
Scoreboard API: Adds a *uvm_sequence_item* to a given queue for a given producer.
- virtual task [add_item_mutexed](#) (string queue_name, string producer, uvm_sequence_item item)
Scoreboard API: Add an item to the scoreboard, using a mutex to ensure than no more than one item is ever added to the SCB at the same time.

- virtual void [compare_trigger](#) (string queue_name="", [cl_syoscb_item](#) item=null)
Scoreboard API: Invokes the scoreboard's compare strategy
- virtual void [dump](#) ()
Scoreboard API: Dump items to files if [cl_syoscb_cfg::full_scb_dump](#) is enabled
- virtual void [flush_queues_all](#) ()
Scoreboard API: Shorthand for flushing all queues
- virtual void [flush_queues](#) (string queue_name="")
Scoreboard API: Flushes the contents of either all queues or a specific queue.
- virtual bit [empty_queues](#) (string queue_name="")
Returns whether all queues or a specific queue is empty or not:
- virtual bit [insert_queues](#) (string queue_name="")
Returns whether at least one element has been inserted in all queues or in a specific queue.
- virtual void [compare_control](#) (bit cc)
Scoreboard API: Toggles the scoreboard's comparison control.
- virtual string [create_total_stats](#) (int unsigned offset, int unsigned first_column_width)
Scoreboard API: Returns a table line summarising the insert/match/flush/orphan stats over all queues in the SCB.
- virtual string [create_report](#) (bit end_of_sim=0b1)
Scoreboard API: Creates a report containing information about this scoreboard.
- virtual int unsigned [get_total_cnt_add_items](#) ()
Scoreboard API: Returns the number of elements that have been inserted into the scoreboard
- virtual int unsigned [get_total_cnt_flushed_items](#) ()
Scoreboard API: Returns the number of elements that have been flushed out of the scoreboard
- virtual int unsigned [get_total_queue_size](#) ()
Scoreboard API: Returns the number of elements that the scoreboard currently contains
- virtual string [get_failed_checks](#) ()
Scoreboard API: Returns a string with information on which checks the scoreboard has failed (e.g.
- virtual [cl_syoscb_subscriber](#) [get_subscriber](#) (string queue_name, string producer)
Scoreboard API: Returns a UVM subscriber for a given combination of queue and producer.
- virtual [cl_syoscb_cfg](#) [get_cfg](#) ()
Gets the configuration for this scoreboard.
- virtual string [create_report_contents](#) (int unsigned offset, int unsigned first_column_width)
Scoreboard API: Returns a string with all queue's statistics, to be inserted into the final report generated by [create_report](#).
- virtual void [pre_abort](#) ()
UVM pre_abort hook.

Protected Member Functions

- virtual void [dump_txt](#) ()
Dumps the shadow queue into text files.
- virtual void [dump_xml](#) ()
Dump the shadow queue into XML files.
- virtual void [dump_split_txt](#) ()
Dumps the shadow queue into separate text files for each queue.
- virtual void [dump_join_txt](#) ()
Dumps the shadow queue into one combined text file called [scoreboard_name].
- virtual void [dump_split_xml](#) ()
Dumps the shadow queue into separate XML files for each queue.
- virtual void [dump_join_xml](#) ()
Dumps the shadow queue into one combined XML file called [scoreboard_name].

- virtual string [print_header](#) (string queue_name)
Gets a header string to print into a shadow queue dump file.
- virtual string [create_queues_stats](#) (int unsigned offset, int unsigned first_column_width)
Returns a table with per-queue statistics for all queues of the scoreboard.
- virtual string [get_queue_failed_checks](#) ()
Returns a string with information on which checks the different queues have failed (e.g.
- virtual void [override_queue_type](#) ()
Performs a factory override of the queue type to be used, based on the value of the [cl_syoscb_cfg::queue_type](#) cfg.
- virtual void [override_compare_type](#) ()
Performs a factory override of the compare type to be used, based on the value of this scoreboard's [cl_syoscb_cfg::compare_type](#).
- virtual void [config_validation](#) ()
Validates that the current scoreboard configuration is not invalid.
- virtual void [intermediate_queue_stat_dump](#) (string queue_name)
Prints the current queue statistics for a queue.

Private Attributes

- [cl_syoscb_cfg](#) cfg
Handle to the global UVM scoreboard configuration.
- [cl_syoscb_queue_base](#) queues []
Array holding handles to all queues.
- [cl_syoscb_compare](#) compare_strategy
Handle to the compare strategy.
- [cl_syoscb_subscriber](#) subscribers [string]
Associative array holding a uvm_subscriber for each queue.
- bit [header_dumped](#) [string]
Flag indicating if a scoreboard header has been dumped when dumping shadow queues.
- string [failed_checks](#) [string]
AA containing failed scoreboard check (e.g. no items inserted))
- semaphore [add_item_mutex](#)
Mutex to be used when calls to [add_item](#) should be mutexed.

13.73.1 Detailed Description

Top level class implementing the root of the SyoSil UVM scoreboard.

Definition at line 2 of file [cl_syoscb.svh](#).

13.73.2 Member Function Documentation

13.73.2.1 [add_item\(\)](#)

```
void cl_syoscb::add_item (
    string queue_name,
    string producer,
    uvm_sequence_item item ) [virtual]
```

Scoreboard API: Adds a `uvm_sequence_item` to a given queue for a given producer.

The method will check if the queue and producer exists before adding it to the queue.

Parameters

<i>queue_name</i>	The name of the queue the item should be added to
<i>producer</i>	The name of the producer that generated this item
<i>item</i>	The sequence item that should be added to the queue

Definition at line 237 of file cl_syoscb.svh.

References `cl_syoscb_queue_base::add_item()`, `cfg`, `compare_trigger()`, `create_report()`, `dump()`, `cl_syoscb_cfg::exist_producer()`, `cl_syoscb_cfg::exist_queue()`, `cl_syoscb_queue_base::get_cnt_add_item()`, `cl_syoscb_cfg::get_disable_clone()`, `cl_syoscb_cfg::get_full_scb_dump()`, `cl_syoscb_cfg::get_full_scb_max_queue_size()`, `cl_syoscb_queue_base::get_last_inserted_item()`, `cl_syoscb_cfg::get_max_queue_size()`, `cl_syoscb_cfg::get_queue()`, `cl_syoscb_cfg::get_queue_stat_interval()`, `cl_syoscb_cfg::get_scb_name()`, `cl_syoscb_cfg::get_scb_stat_interval()`, `cl_syoscb_queue_base::get_size()`, `get_total_cnt_add_items()`, `intermediate_queue_stat_dump()`, and `cl_syoscb_queue_base::shadow_items`.

Referenced by `add_item_mutexed()`, and `cl_syoscb_subscriber::write()`.

13.73.2.2 add_item_mutexed()

```
task cl_syoscb::add_item_mutexed (
    string queue_name,
    string producer,
    uvm_sequence_item item ) [virtual]
```

Scoreboard API: Add an item to the scoreboard, using a mutex to ensure than no more than one item is ever added to the SCB at the same time.

For additional details on adding items to the SCB, see [add_item](#)

Parameters

<i>queue_name</i>	The name of the queue the item should be added to
<i>producer</i>	The name of the producer that generated this item
<i>item</i>	The sequence item that should be added to the queue

Definition at line 314 of file cl_syoscb.svh.

References `add_item()`, `add_item_mutex`, `cfg`, and `cl_syoscb_cfg::get_mutexed_add_item_enable()`.

Referenced by `cl_syoscb_subscriber::write()`.

13.73.2.3 build_phase()

```
void cl_syoscb::build_phase (
    uvm_phase phase )
```

UVM build phase.

Gets the scoreboard configuration and forwards it to the child components (`cl_syoscb_queue` and [cl_syoscb_compare](#)). Additionally, it creates all of the queues defined in the configuration object. Finally, it also creates the compare strategy via a factory create call.

Definition at line 112 of file `cl_syoscb.svh`.

References `add_item_mutex`, `cfg`, `compare_strategy`, `cl_syoscb_cfg::get_mutexed_add_item_enable()`, `cl_syoscb_cfg::get_print_cfg()`, `cl_syoscb_cfg::get_producer()`, `cl_syoscb_cfg::get_producers()`, `cl_syoscb_cfg::get_scb_name()`, `cl_syoscb_cfg_pl::list`, `override_compare_type()`, `override_queue_type()`, `queues`, `cl_syoscb_subscriber::set_mutexed_add_item_enable()`, `cl_syoscb_subscriber::set_producer()`, `cl_syoscb_subscriber::set_queue_name()`, `cl_syoscb_cfg::set_scb_name()`, `cl_syoscb_cfg::size_queues()`, and `subscribers`.

13.73.2.4 check_phase()

```
void cl_syoscb::check_phase (
    uvm_phase phase )
```

UVM check phase.

Checks if the SCB is empty. If true and [cl_syoscb_cfg::enable_no_insert_check](#) is true, a UVM error is issued.

Definition at line 182 of file `cl_syoscb.svh`.

References `cfg`, `empty_queues()`, `failed_checks`, `cl_syoscb_cfg::get_enable_no_insert_check()`, `cl_syoscb_cfg::get_scb_name()`, and `insert_queues()`.

13.73.2.5 compare_control()

```
void cl_syoscb::compare_control (
    bit cc ) [virtual]
```

Scoreboard API: Toggles the scoreboard's comparison control.

Parameters

<code>cc</code>	Compare control bit. If 1, comparisons are enabled, if 0 they are disabled
-----------------	--

Definition at line 429 of file `cl_syoscb.svh`.

References `cl_syoscb_compare::compare_control()`, and `compare_strategy`.

13.73.2.6 config_validation()

```
void cl_syoscb::config_validation ( ) [protected], [virtual]
```

Validates that the current scoreboard configuration is not invalid.

If the configuration is invalid, raises a UVM_FATAL If the configuration is not recommended but still valid, prints a UVM_INFO message

Definition at line 515 of file cl_syoscb.svh.

References `cfg`, `cl_syoscb_cfg::get_compare_type()`, `cl_syoscb_cfg::get_ordered_next()`, `cl_syoscb_cfg::get_queue_type()`, `cl_syoscb_cfg::get_queues()`, and `cl_syoscb_cfg::get_scb_name()`.

Referenced by `end_of_elaboration_phase()`.

13.73.2.7 create_queues_stats()

```
string cl_syoscb::create_queues_stats (
    int unsigned offset,
    int unsigned first_column_width ) [protected], [virtual]
```

Returns a table with per-queue statistics for all queues of the scoreboard.

Parameters

<i>offset</i>	The x-offset to used when printing items in the first column of the table
<i>first_column_width</i>	The width of the first column of the table

Definition at line 716 of file cl_syoscb.svh.

References `cfg`, `cl_syoscb_queue_base::create_queue_report()`, `cl_syoscb_cfg::get_disable_report()`, `cl_syoscb_cfg::get_queue()`, `cl_syoscb_cfg::get_queues()`, and `cl_syoscb_string_library::scb_separator_str()`.

Referenced by `create_report_contents()`.

13.73.2.8 create_report()

```
string cl_syoscb::create_report (
    bit end_of_sim = 0b1 ) [virtual]
```

Scoreboard API: Creates a report containing information about this scoreboard.

The report contains information about the number of insertions, matches, flushed items and orphaned items.

Parameters

<i>end_of_sim</i>	A bit to indicate whether this is called at the } of simulation or not. This changes the name used to refer to items remaining in the queue when the function is called (orphans vs. remaining)
-------------------	---

Returns

That report

Definition at line 555 of file `cl_syoscb.svh`.

References `cfg`, `create_report_contents()`, `cl_syoscb_cfg::get_max_length_producer()`, `cl_syoscb_cfg::get_max_length_queue_name()`, `cl_syoscb_string_library::scb_header_str()`, and `cl_syoscb_string_library::scb_separator_str()`.

Referenced by `add_item()`, and `report_phase()`.

13.73.2.9 create_report_contents()

```
string cl_syoscb::create_report_contents (
    int unsigned offset,
    int unsigned first_column_width ) [virtual]
```

Scoreboard API: Returns a string with all queue's statistics, to be inserted into the final report generated by [create_report](#).

Parameters

<i>offset</i>	The x-offset to used when printing items in the first column of the table
<i>first_column_width</i>	The width of the first column of the table

Definition at line 613 of file `cl_syoscb.svh`.

References `cfg`, `create_queues_stats()`, `create_total_stats()`, and `cl_syoscb_cfg::get_disable_report()`.

Referenced by `create_report()`, and `cl_syoscb_base::create_scb_stats()`.

13.73.2.10 create_total_stats()

```
string cl_syoscb::create_total_stats (
    int unsigned offset,
    int unsigned first_column_width ) [virtual]
```

Scoreboard API: Returns a table line summarising the insert/match/flush/orphan stats over all queues in the SCB.

Parameters

<i>offset</i>	The x-offset to used when printing items in the first column of the table
<i>first_column_width</i>	The width of the first column of the table

Definition at line 581 of file `cl_syoscb.svh`.

References `cfg`, `cl_syoscb_cfg::get_disable_report()`, `cl_syoscb_cfg::get_scb_name()`, `get_total_cnt_add_items()`, `get_total_cnt_flushed_items()`, `get_total_queue_size()`, and `cl_syoscb_string_library::pad_str()`.

Referenced by `create_report_contents()`, and `cl_syoscb_base::create_scb_stats()`.

13.73.2.11 dump_join_txt()

```
void cl_syoscb::dump_join_txt ( ) [protected], [virtual]
```

Dumps the shadow queue into one combined text file called `[scoreboard_name]`.

`[full_scb_dump_file_name].txt`

Definition at line 815 of file `cl_syoscb.svh`.

References `cfg`, `cl_syoscb_queue_base::dump()`, `cl_syoscb_cfg::get_full_scb_dump_file_name()`, `cl_syoscb_cfg::get_queues()`, `cl_syoscb_string_library::pad_str()`, `print_header()`, and `queues`.

Referenced by `dump_txt()`.

13.73.2.12 dump_join_xml()

```
void cl_syoscb::dump_join_xml ( ) [protected], [virtual]
```

Dumps the shadow queue into one combined XML file called `[scoreboard_name]`.

`[full_scb_dump_file_name].xml`

Definition at line 891 of file `cl_syoscb.svh`.

References `cfg`, `cl_syoscb_queue_base::dump()`, `cl_syoscb_cfg::get_full_scb_dump_file_name()`, `cl_syoscb_cfg::get_queues()`, and `queues`.

Referenced by `dump_xml()`.

13.73.2.13 dump_split_txt()

```
void cl_syoscb::dump_split_txt ( ) [protected], [virtual]
```

Dumps the shadow queue into separate text files for each queue.

The text files are named `[scoreboard_name].[queue_name].[full_scb_dump_file_name].txt`

Definition at line 774 of file `cl_syoscb.svh`.

References `cfg`, `cl_syoscb_queue_base::dump()`, `cl_syoscb_cfg::get_full_scb_dump_file_name()`, `cl_syoscb_cfg::get_queues()`, `header_dumped`, `cl_syoscb_string_library::pad_str()`, `print_header()`, and `queues`.

Referenced by `dump_txt()`.

13.73.2.14 dump_split_xml()

```
void cl_syoscb::dump_split_xml ( ) [protected], [virtual]
```

Dumps the shadow queue into separate XML files for each queue.

The files are named [scoreboard_name].[queue_name].[full_scb_dump_file_name].xml

Definition at line 841 of file cl_syoscb.svh.

References `cfg`, `cl_syoscb_queue_base::dump()`, `cl_syoscb_cfg::get_full_scb_dump_file_name()`, `cl_syoscb_cfg::get_queues()`, `header_dumped`, `cl_syoscb_string_library::pad_str()`, `print_header()`, and `queues`.

Referenced by `dump_xml()`.

13.73.2.15 dump_txt()

```
void cl_syoscb::dump_txt ( ) [protected], [virtual]
```

Dumps the shadow queue into text files.

Will either dump shadow items into one or more files depending on [cl_syoscb_cfg::full_scb_dump_split](#)

Definition at line 754 of file cl_syoscb.svh.

References `cfg`, `dump_join_txt()`, `dump_split_txt()`, and `cl_syoscb_cfg::get_full_scb_dump_split()`.

Referenced by `dump()`.

13.73.2.16 dump_xml()

```
void cl_syoscb::dump_xml ( ) [protected], [virtual]
```

Dump the shadow queue into XML files.

Will either dump shadow items into one or more files depending on [cl_syoscb_cfg::full_scb_dump_split](#)

Definition at line 764 of file cl_syoscb.svh.

References `cfg`, `dump_join_xml()`, `dump_split_xml()`, and `cl_syoscb_cfg::get_full_scb_dump_split()`.

Referenced by `dump()`.

13.73.2.17 empty_queues()

```
bit cl_syoscb::empty_queues (
    string queue_name = "" ) [virtual]
```

Returns whether all queues or a specific queue is empty or not:

Parameters

<i>queue_name</i>	The queue that should be checked for emptiness. If "" is passed, checks all queues
-------------------	--

Returns

1 if the given queue (or all queues) are empty, 0 otherwise

Definition at line 372 of file cl_syoscb.svh.

References `cfg`, `cl_syoscb_queue_base::empty()`, `cl_syoscb_cfg::exist_queue()`, `cl_syoscb_cfg::get_queue()`, `cl_syoscb_cfg::get_scb_name()`, and `queues`.

Referenced by `check_phase()`.

13.73.2.18 end_of_elaboration_phase()

```
void cl_syoscb::end_of_elaboration_phase (
    uvm_phase phase )
```

UVM end of elaboration phase.

Validate the scb configuration before proceeding forward. Generate a UVM_FATAL for configuration combinations which are not allowed, or a warning if the combination has been internally evaluated as not recommended.

Definition at line 174 of file cl_syoscb.svh.

References `config_validation()`.

13.73.2.19 flush_queues()

```
void cl_syoscb::flush_queues (
    string queue_name = "" ) [virtual]
```

Scoreboard API: Flushes the contents of either all queues or a specific queue.

Parameters

<i>queue_name</i>	The name of the queue to flush. If "" is passed, flushes all queues
-------------------	---

Definition at line 349 of file cl_syoscb.svh.

References `cfg`, `cl_syoscb_cfg::exist_queue()`, `cl_syoscb_queue_base::flush_queue()`, `cl_syoscb_cfg::get_queue()`, and `queues`.

Referenced by `flush_queues_all()`.

13.73.2.20 get_failed_checks()

```
string cl_syoscb::get_failed_checks ( ) [virtual]
```

Scoreboard API: Returns a string with information on which checks the scoreboard has failed (e.g.

any queues non-empty, any queues with no insertions) This report also contains the per-queue information generated by [get_queue_failed_checks](#)

Definition at line 694 of file `cl_syoscb.svh`.

References `failed_checks`, and `get_queue_failed_checks()`.

Referenced by `cl_syoscb_base::get_scb_failed_checks()`.

13.73.2.21 get_queue_failed_checks()

```
string cl_syoscb::get_queue_failed_checks ( ) [protected], [virtual]
```

Returns a string with information on which checks the different queues have failed (e.g.

not empty at end of sim, no insertions). If they are not empty it also shows the number of orphans.

Definition at line 680 of file `cl_syoscb.svh`.

References `failed_checks`, `cl_syoscb_queue_base::get_failed_checks()`, and `queues`.

Referenced by `get_failed_checks()`, and `report_phase()`.

13.73.2.22 get_subscriber()

```
cl_syoscb_subscriber cl_syoscb::get_subscriber (
    string queue_name,
    string producer ) [virtual]
```

Scoreboard API: Returns a UVM subscriber for a given combination of queue and producer.

The returned UVM subscriber can then be connected to a UVM monitor or similar which produces transactions which should be scoreboarded.

Parameters

<i>queue_name</i>	The name of the queue that items should be added to
<i>producer</i>	The name of the producer that should add items to the queue

Returns

A handle to a uvm_subscriber that will insert items into the given queue with that producers name

Definition at line 741 of file cl_syoscb.svh.

References `cfg`, `cl_syoscb_cfg::get_scb_name()`, and `subscribers`.

13.73.2.23 insert_queues()

```
bit cl_syoscb::insert_queues (
    string queue_name = "" ) [virtual]
```

Returns whether at least one element has been inserted in all queues or in a specific queue.

Parameters

<i>queue_name</i>	The queue to check for insertions. If "" is passed, checks all queues
-------------------	---

Returns

1 if the given queue (or all queues) has had at least one insertion, 0 otherwise

Definition at line 400 of file cl_syoscb.svh.

References `cfg`, `cl_syoscb_cfg::exist_queue()`, `cl_syoscb_queue_base::get_cnt_add_item()`, `cl_syoscb_cfg::get_queue()`, `cl_syoscb_cfg::get_scb_name()`, and `queues`.

Referenced by `check_phase()`.

13.73.2.24 intermediate_queue_stat_dump()

```
void cl_syoscb::intermediate_queue_stat_dump (
    string queue_name ) [protected], [virtual]
```

Prints the current queue statistics for a queue.

This can be used to get queue statistics throughout simulation.

Parameters

<i>queue_name</i>	The name of the queue to dump statistics for.
-------------------	---

Definition at line 928 of file cl_syoscb.svh.

References `cfg`, `cl_syoscb_queue_base::create_queue_report()`, `cl_syoscb_cfg::exist_queue()`, `cl_syoscb_queue_base::get_cnt_add_item()`, `cl_syoscb_cfg::get_max_length_producer()`, `cl_syoscb_cfg::get_max_length`

`queue_name()`, `cl_syoscb_cfg::get_queue()`, `cl_syoscb_string_library::scb_header_str()`, and `cl_syoscb_string_library::scb_separator_str()`.

Referenced by `add_item()`.

13.73.2.25 `override_queue_type()`

```
void cl_syoscb::override_queue_type ( ) [protected], [virtual]
```

Performs a factory override of the queue type to be used, based on the value of the `cl_syoscb_cfg::queue_type` `cfg`.

knob. Once factory override has been performed, creates all queues in this scoreboard and forwards the configuration object to them

Definition at line 437 of file `cl_syoscb.svh`.

References `cfg`, `cl_syoscb_cfg::get_queue_type()`, `cl_syoscb_cfg::get_queues()`, `cl_syoscb_cfg::get_scb_name()`, `queues`, and `cl_syoscb_cfg::set_queue()`.

Referenced by `build_phase()`.

13.73.2.26 `pre_abort()`

```
void cl_syoscb::pre_abort ( ) [virtual]
```

UVM `pre_abort` hook.

Ensures that all shadow items are dumped if a `UVM_ERROR` is about to stop simulation

Definition at line 955 of file `cl_syoscb.svh`.

References `cfg`, `dump()`, and `cl_syoscb_cfg::get_full_scb_dump()`.

13.73.2.27 `print_header()`

```
string cl_syoscb::print_header (
    string queue_name ) [protected], [virtual]
```

Gets a header string to print into a shadow queue dump file.

Parameters

<code>queue_name</code>	The header for that queue
-------------------------	---------------------------

Definition at line 963 of file `cl_syoscb.svh`.

References `cfg`, and `cl_syoscb_cfg::get_full_scb_dump_type()`.

Referenced by `dump_join_txt()`, `dump_split_txt()`, and `dump_split_xml()`.

The documentation for this class was generated from the following files:

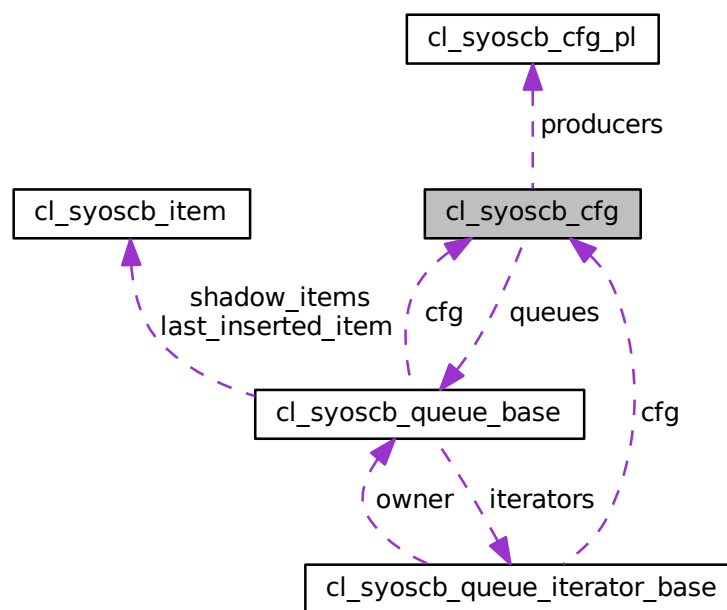
- `cl_syoscb.svh`
- `pk_syoscb.sv`

13.74 cl_syoscb_cfg Class Reference

Configuration class for the SyoSil UVM scoreboard.

Inherits `uvm_object`, and `uvm_object`.

Collaboration diagram for `cl_syoscb_cfg`:



Public Member Functions

- virtual void `init` (string `scb_name`, string `queues`[], string `producers`[])
Configuration API: Initializes the scoreboard's `cfg` with the given input parameters.
- virtual `cl_syoscb_queue_base` `get_queue` (string `queue_name`)
Configuration API: Returns a queue handle for the specified queue.
- virtual void `set_queue` (string `queue_name`, `cl_syoscb_queue_base` `queue`)
Configuration API: Sets the queue object for a given queue.
- virtual void `get_queues` (output string `queue_names`[])

- Configuration API:** Returns all queue names as a string list

 - virtual void [set_queues](#) (string queue_names[])
- Configuration API:** Will set the legal queues when provided with a list of queue names.

 - virtual bit [exist_queue](#) (string queue_name)
- Configuration API:** Checks if a queue with a given name exists.

 - virtual int unsigned [size_queues](#) ()
- Configuration API:** Returns the number of queues in this SCB

 - virtual [cl_syoscb_cfg_pl](#) [get_producer](#) (string producer)
- Configuration API:** Gets the producer object for a specified producer.

 - virtual bit [set_producer](#) (string producer, queue_names[])
- Configuration API:** Sets the given producer for the listed queues If any errors occur, information about this is printed as a UVM_DEBUG message If a list of queues has already been set for a given producer, overrides that list.

 - virtual bit [exist_producer](#) (string producer)
- Configuration API:** Checks if a given producer exists.

 - virtual void [get_producers](#) (output string producers[])
- Configuration API:** Returns the names of all producers

 - virtual string [get_primary_queue](#) ()
- Configuration API:** Gets the name of primary queue for this SCB.

 - virtual bit [set_primary_queue](#) (string primary_queue_name)
- Configuration API:** Sets the primary queue.

 - virtual void [set_queue_type](#) (t_scb_queue_type qt)
- Configuration API:** Set the value of the [queue_type](#) member variable

 - virtual t_scb_queue_type [get_queue_type](#) ()
- Configuration API:** Get the value of the [queue_type](#) member variable

 - virtual void [set_compare_type](#) (t_scb_compare_type ct)
- Configuration API:** Set the value of the [compare_type](#) member variable

 - virtual t_scb_compare_type [get_compare_type](#) ()
- Configuration API:** Get the value of the [compare_type](#) member variable

 - virtual void [set_trigger_greediness](#) (t_scb_compare_greed tg)
- Configuration API:** Set the value of the [trigger_greediness](#) member variable

 - virtual t_scb_compare_greed [get_trigger_greediness](#) ()
- Configuration API:** Get the value of the [trigger_greediness](#) member variable

 - virtual void [set_end_greediness](#) (t_scb_compare_greed eg)
- Configuration API:** Set the value of the [end_greediness](#) member variable

 - virtual t_scb_compare_greed [get_end_greediness](#) ()
- Configuration API:** Get the value of the [end_greediness](#) member variable

 - virtual void [set_disable_clone](#) (bit dc)
- Configuration API:** Set the value of the [disable_clone](#) member variable

 - virtual bit [get_disable_clone](#) ()
- Configuration API:** Get the value of the [disable_clone](#) member variable

 - virtual void [set_disable_compare_after_error](#) (bit dcae)
- Configuration API:** Set the value of the [disable_compare_after_error](#) member variable

 - virtual bit [get_disable_compare_after_error](#) ()
- Configuration API:** Get the value of the [disable_compare_after_error](#) member variable

 - virtual void [set_max_queue_size](#) (string queue_name, int unsigned mqs)
- Configuration API:** Set the maximum number of items allowed in a given queue.

 - virtual int unsigned [get_max_queue_size](#) (string queue_name)
- Configuration API:** Returns the maximum number of items allowed in a given queue.

 - virtual void [set_orphans_as_errors](#) (oae)
- Configuration API:** Set the value of the [print_orphans_as_errors](#) member variable

 - virtual bit [get_orphans_as_errors](#) ()

- Configuration API:** Get the value of the [print_orphans_as_errors](#) member variable

 - virtual void [set_max_print_orphans](#) (int mpo)

Configuration API: Set the value of the [max_print_orphans](#) member variable Not that if mpo < -1 throws a UVM↔_FATAL

 - virtual int [get_max_print_orphans](#) ()

Configuration API: Get the value of the [max_print_orphans](#) member variable

 - virtual void [set_disable_report](#) (bit dr)

Configuration API: Set the value of the [disable_report](#) member variable

 - virtual bit [get_disable_report](#) ()

Configuration API: Get the value of the [disable_report](#) member variable

 - virtual void [set_enable_queue_stats](#) (string queue_name, bit eqs)

Configuration API: Set the value of [enable_queue_stats](#) for a given queue If no queue exists with that name, throws a UVM_FATAL

 - virtual bit [get_enable_queue_stats](#) (string queue_name)

Configuration API: Get the value of [enable_queue_stats](#) for a given queue If no queue exists with that name, throws a UVM_FATAL

 - virtual string [get_scb_name](#) ()

Configuration API: Get the name of the SCB that this cfg is related to

 - virtual void [set_scb_name](#) (string scb_name)

Configuration API: Set the name of the SCB that this cfg is related to

 - virtual bit [get_ordered_next](#) ()

Configuration API: Get the value of the [ordered_next](#) member variable.

 - virtual void [set_ordered_next](#) (bit ordered_next)

Configuration API: Set the value of the [ordered_next](#) member variable.

 - virtual t_hash_compare_check [get_hash_compare_check](#) ()

Configuration API: Get the value of the [hash_compare_check](#) member variable

 - virtual void [set_hash_compare_check](#) (t_hash_compare_check hcc)

Configuration API: Set the value of the [hash_compare_check](#) member variable

 - virtual bit [get_print_cfg](#) ()

Configuration API: Get the value of the [print_cfg](#) member variable

 - virtual void [set_print_cfg](#) (bit pc)

Configuration API: Set the value of the [print_cfg](#) member variable

 - virtual bit [dynamic_primary_queue](#) ()

Configuration API: Checks whether this SCB uses a dynamic or static primary queue.

 - virtual void [set_full_scb_dump](#) (bit fsd)

Configuration API: Set the value of the [full_scb_dump](#) member variable

 - virtual bit [get_full_scb_dump](#) ()

Configuration API: Get the value of the [full_scb_dump](#) member variable

 - virtual void [set_enable_c2s_full_scb_dump](#) (bit ecfsd)

Configuration API: Get the value of the [enable_c2s_full_scb_dump](#) member variable

 - virtual bit [get_enable_c2s_full_scb_dump](#) ()

Configuration API: Set the value of the [enable_c2s_full_scb_dump](#) member variable

 - virtual void [set_full_scb_dump_type](#) (t_dump_type fsdt)

Configuration API: Set the value of the [full_scb_dump_type](#) member variable

 - virtual t_dump_type [get_full_scb_dump_type](#) ()

Configuration API: Get the value of the [full_scb_dump_type](#) member variable

 - virtual string [get_full_scb_dump_file_name](#) ()

Configuration API: Get the value of the [full_scb_dump_file_name](#) member variable

 - virtual void [set_full_scb_dump_file_name](#) (string full_scb_dump_file_name)

Configuration API: Set the value of the [full_scb_dump_file_name](#) member variable

 - virtual bit [set_full_scb_dump_split](#) (bit fsds)

Configuration API: Set the value of the `full_scb_dump_split` member variable Note that setting `full_scb_max_queue_size > 0` for any queue in the SCB will make it impossible to set `fsds=0`.

- virtual bit `get_full_scb_dump_split` ()

Configuration API: Get the value of the `full_scb_dump_split` member variable

- virtual void `set_full_scb_max_queue_size` (string queue_name, int unsigned fsmqs)

Configuration API: Set the value of the `full_scb_max_queue_size` member variable.

- virtual int unsigned `get_full_scb_max_queue_size` (string queue_name)

Configuration API: Get the value of the `full_scb_max_queue_size` member variable for a given queue.

- virtual int unsigned `get_max_length_queue_name` ()

Configuration API: Returns the length of the queue name with maximum length

- virtual int unsigned `get_max_length_producer` ()

Configuration API: Returns the length of the producer name with maximum length

- virtual void `set_enable_comparer_report` (bit ecr, string queue_names[], string producer_names[])

Configuration API: Enables or disables the comparer report for a number of comparers.

- virtual bit `get_enable_comparer_report` (string queue_name, string producer_name)

Configuration API: Returns the comparer report enable bit associated with a given queue/producer combination.

- virtual void `set_default_enable_comparer_report` (bit ecr)

Configuration API: Set the value of the `default_enable_comparer_report` member variable.

- virtual bit `get_default_enable_comparer_report` ()

Configuration API: Get the value of the `default_enable_comparer_report` member variable

- virtual void `set_comparer` (uvm_comparer comparer, string queue_names[], string producer_names[])

Configuration API: Sets the comparer to be used for a number of queues.

- virtual uvm_comparer `get_comparer` (string queue_name, string producer_name)

Configuration API: Returns the comparer associated with a given queue and producer.

- virtual void `set_default_comparer` (uvm_comparer comparer)

Configuration API: Set the value of the `default_comparer` member variable

- virtual uvm_comparer `get_default_comparer` ()

Configuration API: Get the value of the `default_comparer` member variable

- virtual void `set_printer_verbosity` (bit pv, string queue_names[], string producer_names[])

Configuration API: Sets the verbosity level to be used for a number of printers.

- virtual bit `get_printer_verbosity` (string queue_name, string producer_name)

Configuration API: Returns the verbosity bit associated with a given queue/producer combination.

- virtual void `set_default_printer_verbosity` (bit pv)

Configuration API: Set the value of the `default_printer_verbosity` member variable.

- virtual bit `get_default_printer_verbosity` ()

Configuration API: Get the value of the `default_printer_verbosity` member variable

- virtual void `set_printer` (uvm_printer printer, string queue_names[], string producer_names[])

Configuration API: Sets the given uvm_printer to be used for some queue/producer-combinations.

- virtual uvm_printer `get_printer` (string queue_name, string producer_name)

Configuration API: Returns the printer associated with a given producer/queue combination.

- virtual uvm_printer `get_default_printer` ()

Configuration API: Get the value of the `default_printer` member variable

- virtual void `set_default_printer` (uvm_printer printer)

Configuration API: Set the value of the `default_printer` member variable

- virtual void `set_enable_no_insert_check` (bit enic)

Configuration API: Set the value of the `enable_no_insert_check` member variable.

- virtual bit `get_enable_no_insert_check` ()

Configuration API: Gets the values of the `enable_no_insert_check` member variable

- virtual void `set_max_search_window` (int unsigned sw, string queue_names[])

Configuration API: Sets the maximum search window when performing OOO, IOP or user defined comparison operations.

- virtual int unsigned [get_max_search_window](#) (string queue_name)
Configuration API: Returns the value of [max_search_window](#) for a given queue.
- virtual bit [get_dump_orphans_to_files](#) ()
Configuration API: Gets the value of the [dump_orphans_to_files](#) member variable
- virtual void [set_dump_orphans_to_files](#) (bit dotf)
Configuration API: Sets the value of the [dump_orphans_to_files](#) member variable.
- virtual string [get_orphan_dump_file_name](#) ()
Configuration API: Gets the value of the [orphan_dump_file_name](#) member variable
- virtual void [set_orphan_dump_file_name](#) (string odfn)
Configuration API: Sets the value of the [orphan_dump_file_name](#) member variable
- virtual void [set_mutexed_add_item_enable](#) (bit maie)
Configuration API: Sets the value of the [mutexed_add_item_enable](#) member variable.
- virtual bit [get_mutexed_add_item_enable](#) ()
Configuration API: Gets the value of the [mutexed_add_item_enable](#) member variable.
- virtual void [set_queue_stat_interval](#) (string queue_name, int unsigned qsi)
Configuration API: Sets the value of the [queue_stat_interval](#) member variable for the given queue.
- virtual int unsigned [get_queue_stat_interval](#) (string queue_name)
Configuration API: Gets the value of the [queue_stat_interval](#) member variable for the given queue.
- virtual void [set_scb_stat_interval](#) (int unsigned ssi)
Configuration API: Sets the value of the [scb_stat_interval](#) member variable
- virtual int unsigned [get_scb_stat_interval](#) ()
Configuration API: Gets the value of the [scb_stat_interval](#) member variable
- virtual void [set_orphan_dump_type](#) (t_dump_type odt)
Configuration API: Sets the value of the [orphan_dump_type](#) member variable
- virtual t_dump_type [get_orphan_dump_type](#) ()
Configuration API: Get the value of the [orphan_dump_type](#) member variable

Public Attributes

- local t_scb_queue_type [queue_type](#) = pk_syoscb::SYOSCB_QUEUE_USER_DEFINED
Queue topology used in the SCB. Defaults to `pk_syoscb::SYOSCB_QUEUE_USER_DEFINED`.
- local t_scb_compare_type [compare_type](#) = pk_syoscb::SYOSCB_COMPARE_USER_DEFINED
Compare strategy used in the SCB. Defaults to `pk_syoscb::SYOSCB_COMPARE_IO`.

Private Attributes

- [cl_syoscb_queue_base](#) [queues](#) [string]
Associative array holding handles to each queue. Indexed by queue name.
- [cl_syoscb_cfg_pl](#) [producers](#) [string]
Associative array indexed by producer name.
- string [primary_queue](#)
Name of the primary queue used in this scoreboard.
- string [scb_name](#)
The name of the SCB. Default will be the instance name of the SCB component if the name is not set explicitly.
- t_scb_compare_greed [trigger_greediness](#) = pk_syoscb::SYOSCB_COMPARE_NOT_GREEDY
Defines the greed level for comparison operations.
- t_scb_compare_greed [end_greediness](#) = pk_syoscb::SYOSCB_COMPARE_GREEDY
See [trigger_greediness](#) for description.
- bit [enable_no_insert_check](#) = 0b1

- Enable/disable insertion checking on queues.*

 - bit `disable_clone` = 0b0

Controls whether calls to `cl_syoscb::add_item` will clone the given `uvm_sequence_item` or reuse the handle.
 - bit `disable_compare_after_error` = 0b0

Controls whether comparisons should be disabled after the first `UVM_ERROR` is raised.
 - int unsigned `max_queue_size` [string]

Maximum number of elements in each queue before an error is signaled.
 - bit `print_orphans_as_errors` = 0b0

Controls whether orphaned items in the queues should be treated as errors when printing at the end of simulation.
 - int `max_print_orphans` = 0

Select the maximum number of orphaned elements to print if any orphans are left in a queue after simulation.
 - bit `dump_orphans_to_files` = 0b0

Controls whether all orphaned items should be dumped to queue-specific files at the end of simulation.
 - bit `disable_report` = 0b0

Controls whether a report should be generated in the `report_phase`.
 - bit `enable_queue_stats` [string]

Enable/disable the printing of queue's statistics per producer.
 - bit `full_scb_dump` = 0b0

Controls whether all transactions going into the SCB should be dumped to a logfile.
 - bit `enable_c2s_full_scb_dump` = 0b0

Controls whether items in the full scoreboard dump should be dumped using `print()` or `convert2string()`.
 - bit `full_scb_dump_split` = 0b0

Controls whether SCB dumps (controlled by `full_scb_dump`) print all transactions in the same file, or if separate files are used for each queue.
 - int unsigned `full_scb_max_queue_size` [string]

Controls the number of elements that a queue in the SCB can receive before transaction dumping starts.
 - t_dump_type `full_scb_dump_type` = `pk_syoscb::TXT`

File format used when dumping SCB contents to a logfile.
 - t_dump_type `orphan_dump_type` = `pk_syoscb::TXT`

File format used when dumping orphans to logfiles.
 - string `full_scb_dump_file_name` = "full_scb_dump"

Base file name used when dumping SCB contents to a logfile.
 - string `orphan_dump_file_name` = "orphan_dump"

Base file name used when dumping orphans to a logfile.
 - bit `ordered_next` = 0b1

Controls whether a strict item ordering should be used in `assoc`.
 - t_hash_compare_check `hash_compare_check` = `pk_syoscb::SYOSCB_HASH_COMPARE_NO_VALIDATION`

Controls sanity check comparisons on hash queues.
 - bit `print_cfg` = 0b0

Controls whether the scoreboard's configuration values should be printed in the `cl_syoscb::build_phase()`.
 - bit `enable_comparer_report` [string][string]

Associative array holding the bit enabling the comparer report for a specific queue/producer combination.
 - bit `default_enable_comparer_report` = 0b1

The default comparer report toggle for a `uvm_comparer` that can be used when no other verbosity has been assigned to a queue's comparer.
 - uvm_comparer `comparers` [string][string]

Associative array holding handles to comparers used for a specific queue/producer combination.
 - uvm_comparer `default_comparer`

The default `uvm_comparer` that can be used when no other comparer has been assigned to a queue/producer.
 - bit `printer_verbosity` [string][string]

- Associative array holding the printer verbosity bit for a specific queue/producer combination.*

 - bit `default_printer_verbosity` = 0b0

Default printer verbosity bit.
- uvm_printer `printers` [string][string]

Associative array holding handles to printers used for a specific queue/producer combination.
- uvm_printer `default_printer`

The default printer used for all printing purposes if no specific printer has been associated with a queue.
- int unsigned `max_search_window` [string]

The maximum number of entries to iterate through in a queue when performing OOO compare.
- bit `mutexed_add_item_enable` = 0b0

Controls whether `cl_syoscb::add_item()` should be mutexed or not.
- int unsigned `queue_stat_interval` [string]

Defines an interval value N for each queue, such that the queue's statistics are printed on every N'th insertion.
- int unsigned `scb_stat_interval` = 0

Defines an interval value N, similar to `queue_stat_interval`, causing a printout of all queue stats in the SCB on every N'th insertion.

13.74.1 Detailed Description

Configuration class for the SyoSil UVM scoreboard.

Definition at line 2 of file `cl_syoscb_cfg.svh`.

13.74.2 Member Function Documentation

13.74.2.1 dynamic_primary_queue()

```
bit cl_syoscb_cfg::dynamic_primary_queue ( ) [virtual]
```

Configuration API: Checks whether this SCB uses a dynamic or static primary queue.

Returns

0b1 if the primary queue is dynamic, 10b0 if it is static

Definition at line 754 of file `cl_syoscb_cfg.svh`.

References `get_primary_queue()`.

Referenced by `cl_syoscb_compare_base::split_queues()`.

13.74.2.2 exist_producer()

```
bit cl_syoscb_cfg::exist_producer (
    string producer ) [virtual]
```

Configuration API: Checks if a given producer exists.

Parameters

<i>producer</i>	The name of the producer to check
-----------------	-----------------------------------

Returns

0b1 if that producer exists, 10b0 otherwise

Definition at line 517 of file `cl_syoscb_cfg.svh`.

References `cl_syoscb_cfg_pl::exists()`, and `producers`.

Referenced by `cl_syoscb::add_item()`, `get_comparer()`, `get_enable_comparer_report()`, `get_printer()`, `get_printer_verbosity()`, `get_producer()`, `set_comparer()`, `set_enable_comparer_report()`, `set_printer()`, and `set_printer_verbosity()`.

13.74.2.3 exist_queue()

```
bit cl_syoscb_cfg::exist_queue (
    string queue_name ) [virtual]
```

Configuration API: Checks if a queue with a given name exists.

Parameters

<i>queue_name</i>	The name of the queue to check
-------------------	--------------------------------

Returns

0b1 if a queue with that name exists, 10b0 if not

Definition at line 454 of file `cl_syoscb_cfg.svh`.

References `queues`.

Referenced by `cl_syoscb::add_item()`, `cl_syoscb::empty_queues()`, `cl_syoscb::flush_queues()`, `get_enable_comparer_report()`, `get_enable_queue_stats()`, `get_full_scb_max_queue_size()`, `get_max_queue_size()`, `get_max_search_window()`, `get_printer_verbosity()`, `get_queue()`, `get_queue_stat_interval()`, `cl_syoscb::insert_queues()`, `cl_syoscb::intermediate_queue_stat_dump()`, `set_enable_comparer_report()`, `set_enable_queue_stats()`, `set_full_scb_max_queue_size()`, `set_max_queue_size()`, `set_max_search_window()`, `set_primary_queue()`, `set_printer_verbosity()`, `set_producer()`, and `set_queue_stat_interval()`.

13.74.2.4 get_comparer()

```
uvm_comparer cl_syoscb_cfg::get_comparer (
    string queue_name,
    string producer_name ) [virtual]
```

Configuration API: Returns the comparer associated with a given queue and producer.

Parameters

<i>queue_name</i>	Name of the queue for which the comparer should be returned
<i>producer_name</i>	Name of the producer for which the associated queues comparer should be returned

Returns

The requested comparer. If no comparer has been set for a given queue/producer, returns null. Also returns null if either of the input parameters are invalid.

Definition at line 991 of file cl_syoscb_cfg.svh.

References `comparers`, `exist_producer()`, and `get_producer()`.

Referenced by `cl_syoscb_compare_io_2hp::primary_loop_do()`, `cl_syoscb_queue_locator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::search()`, `cl_syoscb_queue_locator_std::search()`, `cl_syoscb_compare_io::secondary_loop_do()`, `cl_syoscb_compare_iop::secondary_loop_do()`, `cl_syoscb_queue_locator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::validate_match()`, and `cl_syoscb_queue_locator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::validate_no_match()`.

13.74.2.5 get_enable_comparer_report()

```
bit cl_syoscb_cfg::get_enable_comparer_report (
    string queue_name,
    string producer_name ) [virtual]
```

Configuration API: Returns the comparer report enable bit associated with a given queue/producer combination.

If no bit has been set for a given queue's comparer by using [set_enable_comparer_report](#), returns [default_enable_comparer_report](#)

Parameters

<i>queue_name</i>	Name of the queue for which the designated comparers enable report bit should be returned
<i>producer_name</i>	Name of the producer for which the associated queues comparers enable report bit should be returned

Returns

The given queue/producer combinations comparer enable report bit, or the default value if none has been set for this specific queue/producer combination

Definition at line 928 of file cl_syoscb_cfg.svh.

References `default_enable_comparer_report`, `enable_comparer_report`, `exist_producer()`, and `exist_queue()`.

Referenced by `cl_syoscb_compare_base::generate_miscomp_table()`.

13.74.2.6 `get_enable_queue_stats()`

```
bit cl_syoscb_cfg::get_enable_queue_stats (
    string queue_name ) [virtual]
```

Configuration API: Get the value of [enable_queue_stats](#) for a given queue. If no queue exists with that name, throws a UVM_FATAL.

Parameters

<i>queue_name</i>	The name of the queue to get the value of enable_queue_stats for
-------------------	--

Definition at line 703 of file cl_syoscb_cfg.svh.

References [enable_queue_stats](#), and [exist_queue\(\)](#).

Referenced by [cl_syoscb_queue_base::create_queue_report\(\)](#).

13.74.2.7 `get_full_scb_max_queue_size()`

```
int unsigned cl_syoscb_cfg::get_full_scb_max_queue_size (
    string queue_name ) [virtual]
```

Configuration API: Get the value of the [full_scb_max_queue_size](#) member variable for a given queue.

If no queue exists with that name, prints a UVM_DEBUG message.

Returns

The value of full_scb_max_queue_size if the queue name is valid, 0 otherwise.

Definition at line 842 of file cl_syoscb_cfg.svh.

References [exist_queue\(\)](#), and [full_scb_max_queue_size](#).

Referenced by [cl_syoscb::add_item\(\)](#), and [set_full_scb_dump_split\(\)](#).

13.74.2.8 `get_max_queue_size()`

```
int unsigned cl_syoscb_cfg::get_max_queue_size (
    string queue_name ) [virtual]
```

Configuration API: Returns the maximum number of items allowed in a given queue.

If no queue exists with that name, throws a UVM_FATAL.

Parameters

<i>queue_name</i>	The name of the queue to get the maximum size for
-------------------	---

Definition at line 645 of file cl_syoscb_cfg.svh.

References exist_queue(), and max_queue_size.

Referenced by cl_syoscb::add_item().

13.74.2.9 get_max_search_window()

```
int unsigned cl_syoscb_cfg::get_max_search_window (
    string queue_name ) [virtual]
```

Configuration API: Returns the value of [max_search_window](#) for a given queue.

If an invalid queue name is passed, prints a UVM_DEBUG message

Parameters

<i>queue_name</i>	The queue for which to get the maximum search window.
-------------------	---

Returns

That queues max. search window. If no maximum search window has been set, returns 0

Definition at line 1211 of file cl_syoscb_cfg.svh.

References exist_queue(), and max_search_window.

Referenced by cl_syoscb_compare_ooo::primary_loop_do(), cl_syoscb_compare_iop::primary_loop_do(), cl_syoscb_queue_locator_std::search(), and cl_syoscb_compare_iop::secondary_loop_do().

13.74.2.10 get_primary_queue()

```
string cl_syoscb_cfg::get_primary_queue ( ) [virtual]
```

Configuration API: Gets the name of primary queue for this SCB.

The primary queue is used by the compare algorithms to select which queue to use as the primary one.

Returns

The name of the primary queue. If no primary queue has been set, returns an empty string

Definition at line 538 of file cl_syoscb_cfg.svh.

References primary_queue.

Referenced by dynamic_primary_queue(), and cl_syoscb_compare_base::get_primary_queue_name().

13.74.2.11 get_printer()

```
uvm_printer cl_syoscb_cfg::get_printer (
    string queue_name,
    string producer_name ) [virtual]
```

Configuration API: Returns the printer associated with a given producer/queue combination.

Parameters

<i>queue_name</i>	Name of the queue for which the printer should be returned
<i>producer_name</i>	Name of the producer for which the associated queues printer should be returned

Returns

That queue/producer combinations printer. If none has been set, or either argument is invalid, returns null.

Definition at line 1134 of file cl_syoscb_cfg.svh.

References exist_producer(), get_producer(), and printers.

Referenced by cl_syoscb_queue_base::dump(), and cl_syoscb_queue_base::dump_orphans_to_file().

13.74.2.12 get_printer_verbosity()

```
bit cl_syoscb_cfg::get_printer_verbosity (
    string queue_name,
    string producer_name ) [virtual]
```

Configuration API: Returns the verbosity bit associated with a given queue/producer combination.

Parameters

<i>queue_name</i>	Name of the queue for which the designated printers verbosity bit should be returned
<i>producer_name</i>	Name of the producer for which the associated queues printers verbosity bit should be returned

Returns

That queue/producer combinations printer verbosity bit. If none has been set, or either argument is invalid, returns [default_printer_verbosity](#).

Definition at line 1070 of file cl_syoscb_cfg.svh.

References default_printer_verbosity, exist_producer(), exist_queue(), and printer_verbosity.

13.74.2.13 get_producer()

```
cl_syoscb_cfg_pl cl_syoscb_cfg::get_producer (
    string producer ) [virtual]
```

Configuration API: Gets the producer object for a specified producer.

Parameters

<i>producer</i>	The name of the producer to get the producer object for
-----------------	---

Returns

The producer object for the requested producer, null if no producer has that name

Definition at line 467 of file cl_syoscb_cfg.svh.

References exist_producer(), and producers.

Referenced by cl_syoscb::build_phase(), get_comparer(), get_printer(), set_comparer(), set_enable_comparer_report(), set_printer(), and set_printer_verbosity().

13.74.2.14 get_producers()

```
void cl_syoscb_cfg::get_producers (
    output string producers[] ) [virtual]
```

Configuration API: Returns the names of all producers

Parameters

<i>producers</i>	Handle to dynamic string array in which producer names are returned. If the handle already points to an allocated array, that handle is overwritten.
------------------	--

Definition at line 524 of file cl_syoscb_cfg.svh.

References producers.

Referenced by cl_syoscb::build_phase(), cl_syoscb_queue_base::create_producer_stats(), cl_syoscb_queue_base::flush_queue(), set_comparer(), set_enable_comparer_report(), set_printer(), and set_printer_verbosity().

13.74.2.15 get_queue()

```
cl_syoscb_queue_base cl_syoscb_cfg::get_queue (
    string queue_name ) [virtual]
```

Configuration API: Returns a queue handle for the specified queue.

Parameters

<i>queue_name</i>	The name of the queue to get a handle for
-------------------	---

Returns

A handle to the requested queue, null if no queue with that name exists

Definition at line 398 of file `cl_syoscb_cfg.svh`.

References `exist_queue()`, and `queues`.

Referenced by `cl_syoscb::add_item()`, `cl_syoscb::create_queues_stats()`, `cl_syoscb_compare_base::dynamic_queue_split_do()`, `cl_syoscb::empty_queues()`, `cl_syoscb::flush_queues()`, `cl_syoscb_compare_base::get_queues_item_cnt()`, `cl_syoscb::get_total_cnt_add_items()`, `cl_syoscb::get_total_cnt_flushed_items()`, `cl_syoscb::get_total_queue_size()`, `cl_syoscb::insert_queues()`, `cl_syoscb::intermediate_queue_stat_dump()`, and `cl_syoscb_compare_base::static_queue_split_do()`.

13.74.2.16 get_queue_stat_interval()

```
int unsigned cl_syoscb_cfg::get_queue_stat_interval (
    string queue_name ) [virtual]
```

Configuration API: Gets the value of the [queue_stat_interval](#) member variable for the given queue.

If the given queue name does not match an existing queue, throws a UVM_FATAL. If no stat interval has been set for the given queue yet, returns 0

Parameters

<i>queue_name</i>	The name of the queue for which to get the value
-------------------	--

Definition at line 1283 of file `cl_syoscb_cfg.svh`.

References `exist_queue()`, and `queue_stat_interval`.

Referenced by `cl_syoscb::add_item()`.

13.74.2.17 get_queues()

```
void cl_syoscb_cfg::get_queues (
    output string queue_names[] ) [virtual]
```

Configuration API: Returns all queue names as a string list

Parameters

<i>queue_names</i>	A handle to a dynamic string array where all queue names will be returned. If this handle already points to an allocated array, that array will be lost.
--------------------	--

Definition at line 425 of file cl_syoscb_cfg.svh.

References `queues`.

Referenced by `cl_syoscb::config_validation()`, `cl_syoscb::create_queues_stats()`, `cl_syoscb::dump_join_txt()`, `cl_syoscb::dump_join_xml()`, `cl_syoscb::dump_split_txt()`, `cl_syoscb::dump_split_xml()`, `cl_syoscb_compare_base::dynamic_queue_split_do()`, `cl_syoscb::get_queues()`, `cl_syoscb_compare_base::get_queues_item_cnt()`, `cl_syoscb::get_total_cnt_add_items()`, `cl_syoscb::get_total_cnt_flushed_items()`, `cl_syoscb::get_total_queue_size()`, `cl_syoscb::override_queue_type()`, `set_comparer()`, `set_enable_comparer_report()`, `set_full_scb_dump_split()`, `set_max_search_window()`, `set_printer()`, `set_printer_verbosity()`, and `cl_syoscb_compare_base::static_queue_split_do()`.

13.74.2.18 `init()`

```
void cl_syoscb_cfg::init (
    string scb_name,
    string queues[],
    string producers[] ) [virtual]
```

Configuration API: Initializes the scoreboard's cfg with the given input parameters.

Parameters

<i>scb_name</i>	The name of the SCB that this cfg is related to
<i>queues</i>	Names of all queues used in this SCB
<i>producers</i>	Names of all producers used in this scb

Definition at line 375 of file cl_syoscb_cfg.svh.

References `get_default_comparer()`, `producers`, `queues`, `set_default_comparer()`, `set_producer()`, `set_queues()`, `set_scb_name()`, and `cl_syoscb_comparer_config::set_verbosity()`.

13.74.2.19 `set_comparer()`

```
void cl_syoscb_cfg::set_comparer (
    uvm_comparer comparer,
    string queue_names[],
    string producer_names[] ) [virtual]
```

Configuration API: Sets the comparer to be used for a number of queues.

If both "queue_names" and "producer_names" are empty, sets the given comparer for all queue/producer combinations. If an invalid/non-existent queue or producer name is passed, a DEBUG message is printed.

Parameters

<i>comparer</i>	The comparer to be used for the given queues and producers.
<i>queue_names</i>	Names of the queues for which the given comparer should be used.
<i>producer_names</i>	Names of the producers for which all associated queues should use the given comparer.

Definition at line 959 of file `cl_syoscb_cfg.svh`.

References `comparers`, `exist_producer()`, `cl_syoscb_cfg_pl::exists()`, `get_producer()`, `get_producers()`, and `get_queues()`.

13.74.2.20 `set_default_enable_comparer_report()`

```
void cl_syoscb_cfg::set_default_enable_comparer_report (
    bit ecr ) [virtual]
```

Configuration API: Set the value of the [default_enable_comparer_report](#) member variable.

See [enable_comparer_report](#) for legal values.

Definition at line 943 of file `cl_syoscb_cfg.svh`.

References `default_enable_comparer_report`.

13.74.2.21 `set_default_printer_verbosity()`

```
void cl_syoscb_cfg::set_default_printer_verbosity (
    bit pv ) [virtual]
```

Configuration API: Set the value of the [default_printer_verbosity](#) member variable.

See [printer_verbosity](#) for legal values

Definition at line 1085 of file `cl_syoscb_cfg.svh`.

References `default_printer_verbosity`.

13.74.2.22 `set_dump_orphans_to_files()`

```
void cl_syoscb_cfg::set_dump_orphans_to_files (
    bit dotf ) [virtual]
```

Configuration API: Sets the value of the [dump_orphans_to_files](#) member variable.

Note

If `dotf == 0b1` and [max_print_orphans](#) < 0, a `UVM_FATAL` is thrown as it does not make sense to dump orphans when no orphans are printed.

Definition at line 1235 of file `cl_syoscb_cfg.svh`.

References `dump_orphans_to_files`, and `max_print_orphans`.

13.74.2.23 set_enable_comparer_report()

```
void cl_syoscb_cfg::set_enable_comparer_report (
    bit ecr,
    string queue_names[],
    string producer_names[] ) [virtual]
```

Configuration API: Enables or disables the comparer report for a number of comparers.

If both "queue_names" and "producer_names" are empty, sets the comparer report enable bit for all queue/producer combinations. If an invalid/non-existent queue or producer name is passed, a DEBUG message is printed,

Parameters

<i>ecr</i>	The value of the comparer report enable/disable flag. See enable_comparer_report for value descriptions.
<i>queue_names</i>	Names of the queues for which the designated comparers should use this comparer report enable bit
<i>producer_names</i>	Names of the producers for which all associated queues comparers should use the given value

Definition at line 888 of file cl_syoscb_cfg.svh.

References [enable_comparer_report](#), [exist_producer\(\)](#), [exist_queue\(\)](#), [get_producer\(\)](#), [get_producers\(\)](#), [get_queues\(\)](#), and [cl_syoscb_cfg_pl::list](#).

13.74.2.24 set_enable_queue_stats()

```
void cl_syoscb_cfg::set_enable_queue_stats (
    string queue_name,
    bit eqs ) [virtual]
```

Configuration API: Set the value of [enable_queue_stats](#) for a given queue. If no queue exists with that name, throws a UVM_FATAL.

Parameters

<i>queue_name</i>	The name of the queue to set the value of enable_queue_stats for
<i>eqs</i>	The new value of enable_queue_stats for that queue

Definition at line 692 of file cl_syoscb_cfg.svh.

References [enable_queue_stats](#), and [exist_queue\(\)](#).

13.74.2.25 set_full_scb_dump_split()

```
bit cl_syoscb_cfg::set_full_scb_dump_split (
    bit fsds ) [virtual]
```

Configuration API: Set the value of the [full_scb_dump_split](#) member variable. Note that setting [full_scb_max_queue_size](#) > 0 for any queue in the SCB will make it impossible to set `fsds=0`.

A UVM_DEBUG message is printed if this happens.

Returns

0b1 if the value was successfully set, 10b0 otherwise

Definition at line 802 of file `cl_syoscb_cfg.svh`.

References [full_scb_dump_split](#), [get_full_scb_max_queue_size\(\)](#), and [get_queues\(\)](#).

13.74.2.26 `set_full_scb_max_queue_size()`

```
void cl_syoscb_cfg::set_full_scb_max_queue_size (
    string queue_name,
    int unsigned fsmqs ) [virtual]
```

Configuration API: Set the value of the [full_scb_max_queue_size](#) member variable.

[full_scb_dump_split](#) must be enabled before setting this value. If not, a UVM_DEBUG message is printed and the call fails. If no queue exists with that name, throws a UVM_FATAL.

Parameters

<i>queue_name</i>	The name of the queue for which to set the value of <code>full_scb_max_queue_size</code>
<i>fsmqs</i>	The new value of <code>full_scb_max_queue_size</code>

Definition at line 828 of file `cl_syoscb_cfg.svh`.

References [exist_queue\(\)](#), [full_scb_max_queue_size](#), and [get_full_scb_dump_split\(\)](#).

13.74.2.27 `set_max_queue_size()`

```
void cl_syoscb_cfg::set_max_queue_size (
    string queue_name,
    int unsigned mqs ) [virtual]
```

Configuration API: Set the maximum number of items allowed in a given queue.

Defaults to 0 (no maximum number of items). If no queue exists with that name, throws a UVM_FATAL.

Parameters

<i>queue_name</i>	The name of the queue to modify
<i>mqs</i>	The maximum number of items allowed in that queue

Definition at line 634 of file cl_syoscb_cfg.svh.

References exist_queue(), and max_queue_size.

13.74.2.28 set_max_search_window()

```
void cl_syoscb_cfg::set_max_search_window (
    int unsigned sw,
    string queue_names[] ) [virtual]
```

Configuration API: Sets the maximum search window when performing OOO, IOP or user defined comparison operations.

If the current comparison type is not SYOSCB_COMPARE_OOO, SYOSCB_COMPARE_IOP or SYOSCB_COMPARE_USER_DEFINED, a uvm_fatal is generated. All other comparison types expect matching items to be at the head of their respective queues, so these comparisons are the only place where the notion of a maximum search window makes sense. Will also throw a fatal if the given queue's type is not one of SYOSCB_QUEUE_STD or SYOSCB_QUEUE_USER_DEFINED. A maximum search window does not make sense when using MD5-queues, as all lookups are performed in O(1) time, independent of the number of elements in the queue.

Parameters

<i>sw</i>	The maximum search window the for given queues. If set to 0, all items in the given queues are searched
<i>queue_names</i>	Names of the queues to set the maximum search window for. If an empty array is given, the maximum search window for all queues is set to <i>sw</i> . If an invalid queue name is passed, a UVM_FATAL is raised.

Definition at line 1187 of file cl_syoscb_cfg.svh.

References compare_type, exist_queue(), get_queues(), max_search_window, and queue_type.

13.74.2.29 set_primary_queue()

```
bit cl_syoscb_cfg::set_primary_queue (
    string primary_queue_name ) [virtual]
```

Configuration API: Sets the primary queue.

The primary queue is used by the compare algorithms to select which queue to use as the primary one. If the given name does not match an existing queue's name, prints a UVM_DEBUG message.

Parameters

<i>primary_queue_name</i>	The name of the queue to make the primary queue
---------------------------	---

Returns

0b1 if the primary queue was successfully set, 10b0 if the input queue name does not match a valid queue name

Definition at line 547 of file `cl_syoscb_cfg.svh`.

References `exist_queue()`, and `primary_queue`.

13.74.2.30 set_printer()

```
void cl_syoscb_cfg::set_printer (
    uvm_printer printer,
    string queue_names[],
    string producer_names[] ) [virtual]
```

Configuration API: Sets the given `uvm_printer` to be used for some queue/producer-combinations.

If both "queue_names" and "producer_names" are empty, sets that printer to be used for all queue/producers. If an invalid/non-existent queue or producer name is passed, a DEBUG message is printed,

Parameters

<i>printer</i>	The printer to be used for the given queues and producers
<i>queue_names</i>	Names of the queues which should use the printer.
<i>producer_names</i>	Names of the producers for which all associated queues should use the given printer.

Definition at line 1101 of file `cl_syoscb_cfg.svh`.

References `exist_producer()`, `cl_syoscb_cfg_pl::exists()`, `get_producer()`, `get_producers()`, `get_queues()`, and `printers`.

13.74.2.31 set_printer_verbosity()

```
void cl_syoscb_cfg::set_printer_verbosity (
    bit pv,
    string queue_names[],
    string producer_names[] ) [virtual]
```

Configuration API: Sets the verbosity level to be used for a number of printers.

If both "queue_names" and "producer_names" are empty, sets the verbosity bit for all queue/producer combinations. If an invalid/non-existent queue or producer name is passed, a DEBUG message is printed,

Parameters

<i>pv</i>	The value of the verbosity bit to set. See printer_verbosity for value descriptions
<i>queue_names</i>	Names of the queues for which the designated printers should use this verbosity bit
<i>producer_names</i>	Names of the producers for which all associated queues printers should use the given
	verbosity bit.

Definition at line 1033 of file cl_syoscb_cfg.svh.

References `exist_producer()`, `exist_queue()`, `get_producer()`, `get_producers()`, `get_queues()`, `cl_syoscb_cfg_pl::list`, and `printer_verbosity`.

13.74.2.32 set_producer()

```
bit cl_syoscb_cfg::set_producer (
    string producer,
    queue_names [ ] ) [virtual]
```

Configuration API: Sets the given producer for the listed queues. If any errors occur, information about this is printed as a UVM_DEBUG message. If a list of queues has already been set for a given producer, overrides that list.

Parameters

<i>producer</i>	The name of the producer
<i>queue_names</i>	Array of queue names which the producer should be associated with

Returns

0b1 if everything works correctly, 10b0 if an error occurs

Definition at line 482 of file cl_syoscb_cfg.svh.

References `exist_queue()`, `producers`, and `cl_syoscb_cfg_pl::set_list()`.

Referenced by `init()`.

13.74.2.33 set_queue()

```
void cl_syoscb_cfg::set_queue (
    string queue_name,
    cl_syoscb_queue_base queue ) [virtual]
```

Configuration API: Sets the queue object for a given queue.

Also sets the values of `max_queue_size` and `enable_queue_stats` for the given queue to 0

Parameters

<i>queue_name</i>	The name of the queue
<i>queue</i>	The queue object to set the queue name to point to

Definition at line 412 of file cl_syoscb_cfg.svh.

References `enable_queue_stats`, `max_queue_size`, and `queues`.

Referenced by `cl_syoscb::override_queue_type()`, and `set_queues()`.

13.74.2.34 `set_queue_stat_interval()`

```
void cl_syoscb_cfg::set_queue_stat_interval (
    string queue_name,
    int unsigned qsi ) [virtual]
```

Configuration API: Sets the value of the `queue_stat_interval` member variable for the given queue.

If the given queue name does not match an existing queue, throws a `UVM_FATAL`.

Parameters

<i>queue_name</i>	The name of the queue for which to set the value
<i>qsi</i>	The new value of the queues statistic printout interval

Definition at line 1271 of file `cl_syoscb_cfg.svh`.

References `exist_queue()`, and `queue_stat_interval`.

13.74.2.35 `set_queues()`

```
void cl_syoscb_cfg::set_queues (
    string queue_names[] ) [virtual]
```

Configuration API: Will set the legal queues when provided with a list of queue names.

An example could be: `set_queues({"Q1", "Q2"})`

Parameters

<i>queue_names</i>	The legal queue names to use for this SCB.
--------------------	--

Note

Throws a `UVM_FATAL` if `queue_names` is empty

Definition at line 440 of file `cl_syoscb_cfg.svh`.

References `set_queue()`.

Referenced by `init()`.

13.74.2.36 set_scb_stat_interval()

```
void cl_syoscb_cfg::set_scb_stat_interval (
    int unsigned ssi ) [virtual]
```

Configuration API: Sets the value of the [scb_stat_interval](#) member variable

Parameters

<i>ssi</i>	The new value of the field
------------	----------------------------

Definition at line 1294 of file cl_syoscb_cfg.svh.

References [scb_stat_interval](#).

13.74.2.37 size_queues()

```
int unsigned cl_syoscb_cfg::size_queues ( ) [virtual]
```

Configuration API: Returns the number of queues in this SCB

Returns

That value

Definition at line 460 of file cl_syoscb_cfg.svh.

References [queues](#).

Referenced by [cl_syoscb::build_phase\(\)](#).

13.74.3 Member Data Documentation**13.74.3.1 comparers**

```
uvm_comparer cl_syoscb_cfg::comparers [private]
```

Associative array holding handles to comparers used for a specific queue/producer combination.

If no comparer has been set for a given queue/producer combination, the [default_comparer](#) is used.

Definition at line 167 of file cl_syoscb_cfg.svh.

Referenced by [get_comparer\(\)](#), and [set_comparer\(\)](#).

13.74.3.2 default_comparer

```
uvm_comparer cl_syoscb_cfg::default_comparer [private]
```

The default uvm_comparer that can be used when no other comparer has been assigned to a queue/producer.

By default, this comparer has a verbosity of UVM_LOW, causing miscompare information to be printed when performing OOO compares. To change this, use [cl_syoscb_comparer_config::set_verbosity](#) to change the verbosity level

Definition at line 172 of file cl_syoscb_cfg.svh.

Referenced by [get_default_comparer\(\)](#), and [set_default_comparer\(\)](#).

13.74.3.3 default_enable_comparer_report

```
bit cl_syoscb_cfg::default_enable_comparer_report = 0b1 [private]
```

The default comparer report toggle for a uvm_comparer that can be used when no other verbosity has been assigned to a queue's comparer.

Defaults to 1'b1 (comparer report is enabled) for IO, IOP and IO-2HP comparisons. Defaults to 1'b0 (comparer report is disabled) for OOO and User Defined comparisons. See [enable_comparer_report](#) for additional details.

Definition at line 163 of file cl_syoscb_cfg.svh.

Referenced by [get_default_enable_comparer_report\(\)](#), [get_enable_comparer_report\(\)](#), [set_compare_type\(\)](#), and [set_default_enable_comparer_report\(\)](#).

13.74.3.4 default_printer

```
uvm_printer cl_syoscb_cfg::default_printer [private]
```

The default printer used for all printing purposes if no specific printer has been associated with a queue.

Defaults to being a uvm_default_printer

Definition at line 191 of file cl_syoscb_cfg.svh.

Referenced by [get_default_printer\(\)](#), and [set_default_printer\(\)](#).

13.74.3.5 default_printer_verbosity

```
bit cl_syoscb_cfg::default_printer_verbosity = 0b0 [private]
```

Default printer verbosity bit.

Controls the number of array elements to output when printing a tx item. See field [printer_verbosity](#) for value descriptions. Defaults to 1'b0 (5 elements are printed at the head/tail of lists)

Definition at line 183 of file cl_syoscb_cfg.svh.

Referenced by [get_default_printer_verbosity\(\)](#), [get_printer_verbosity\(\)](#), and [set_default_printer_verbosity\(\)](#).

13.74.3.6 disable_clone

```
bit cl_syoscb_cfg::disable_clone = 0b0 [private]
```

Controls whether calls to [cl_syoscb::add_item](#) will clone the given uvm_sequence_item or reuse the handle.

Defaults to 1'b0

- 1'b0 => Calls to [cl_syoscb::add_item](#) will clone the uvm_sequence_item
- 1'b1 => Calls to [cl_syoscb::add_item](#) will not clone the uvm_sequence_item

Definition at line 44 of file cl_syoscb_cfg.svh.

Referenced by [get_disable_clone\(\)](#), and [set_disable_clone\(\)](#).

13.74.3.7 disable_compare_after_error

```
bit cl_syoscb_cfg::disable_compare_after_error = 0b0 [private]
```

Controls whether comparisons should be disabled after the first UVM_ERROR is raised.

Defaults to 1'b0.

- 1'b0 => Comparisons are not disabled after the first UVM_ERROR
- 1'b1 => Comparisons are disabled after the first UVM_ERROR

Definition at line 49 of file cl_syoscb_cfg.svh.

Referenced by [get_disable_compare_after_error\(\)](#), and [set_disable_compare_after_error\(\)](#).

13.74.3.8 disable_report

```
bit cl_syoscb_cfg::disable_report = 0b0 [private]
```

Controls whether a report should be generated in the report_phase.

Defaults to 1'b0 Used when e.g. this scb is wrapped by [cl_syoscb](#) wrapper.

- 1'b0 => Report is not disabled
- 1'b1 => Report is disabled

Definition at line 78 of file cl_syoscb_cfg.svh.

Referenced by `get_disable_report()`, and `set_disable_report()`.

13.74.3.9 dump_orphans_to_files

```
bit cl_syoscb_cfg::dump_orphans_to_files = 0b0 [private]
```

Controls whether all orphaned items should be dumped to queue-specific files at the end of simulation.

Defaults to 1'b0. If set, a number of files named `<scb_name>.<orphan_dump_file_name>.<queue_name>_orphan.log` are generated at the end of simulation. The number of orphans that are printed is controlled by the knob [max_print_orphans](#) The value of `<orphan_dump_file_name>` is set by [orphan_dump_file_name](#)

- 1'b0 => Orphans are not dumped to files at the end of simulation
- 1'b1 => Orphans are dumped to files at the end of simulation

Definition at line 72 of file cl_syoscb_cfg.svh.

Referenced by `get_dump_orphans_to_files()`, and `set_dump_orphans_to_files()`.

13.74.3.10 enable_c2s_full_scb_dump

```
bit cl_syoscb_cfg::enable_c2s_full_scb_dump = 0b0 [private]
```

Controls whether items in the full scoreboard dump should be dumped using `print()` or `convert2string()`.

Defaults to 1'b0 (using `object.print()`). If enabled and [full_scb_dump_type](#) is set to TXT, the `convert2string()`-implementation of the wrapped object is used when dumping. The output of `convert2string` must be one line, otherwise a UVM_WARNING is raised.

- 1'b0 => Items are dumped using their `.print()`-representation
- 1'b1 => Items are dumped using their `.convert2string()`-representation

Definition at line 97 of file cl_syoscb_cfg.svh.

Referenced by `get_enable_c2s_full_scb_dump()`, and `set_enable_c2s_full_scb_dump()`.

13.74.3.11 enable_comparer_report

```
bit cl_syoscb_cfg::enable_comparer_report [private]
```

Associative array holding the bit enabling the comparer report for a specific queue/producer combination.

The comparer report contains information on the specific fields where a miscompare happens. If no value has been set the value of [default_enable_comparer_report](#) is used.

- 1'b0 => Disable comparer report.
- 1'b1 => Enable comparer report.

Definition at line 157 of file cl_syoscb_cfg.svh.

Referenced by [get_enable_comparer_report\(\)](#), and [set_enable_comparer_report\(\)](#).

13.74.3.12 enable_no_insert_check

```
bit cl_syoscb_cfg::enable_no_insert_check = 0b1 [private]
```

Enable/disable insertion checking on queues.

Defaults to 1'b1.

- 1'b1 => Enables the check. If a queue has not had any insertions at the end of simulation, a UVM_ERROR is raised
- 1'b0 => Disables the insertion check. No error is raised if a queue did not have any insertions.

Definition at line 39 of file cl_syoscb_cfg.svh.

Referenced by [get_enable_no_insert_check\(\)](#), and [set_enable_no_insert_check\(\)](#).

13.74.3.13 enable_queue_stats

```
bit cl_syoscb_cfg::enable_queue_stats [private]
```

Enable/disable the printing of queue's statistics per producer.

Defaults to 1'b0. Indexed by queue name.

- 1'b0 => Queue's producer-specific stats are disabled
- 1'b1 => Queue's producer-specific stats are enabled

Definition at line 84 of file cl_syoscb_cfg.svh.

Referenced by [get_enable_queue_stats\(\)](#), [set_enable_queue_stats\(\)](#), and [set_queue\(\)](#).

13.74.3.14 end_greediness

```
t_scb_compare_greed cl_syoscb_cfg::end_greediness = pk_syoscb::SYOSCB_COMPARE_GREEDY [private]
```

See `trigger_greediness` for description.

Defaults to `pk_syoscb::SYOSCB_COMPARE_GREEDY` This greed level is used in the `cl_syoscb_compare::extract_phase()` to drain remaining matches if they exist.

Definition at line 34 of file `cl_syoscb_cfg.svh`.

Referenced by `get_end_greediness()`, and `set_end_greediness()`.

13.74.3.15 full_scb_dump

```
bit cl_syoscb_cfg::full_scb_dump = 0b0 [private]
```

Controls whether all transactions going into the SCB should be dumped to a logfile.

Defaults to 1'b0 (off).

- 1'b0 => Disables dumping all transactions to a logfile
- 1'b1 => Enables dumping all transactions to a logfile

Definition at line 89 of file `cl_syoscb_cfg.svh`.

Referenced by `get_full_scb_dump()`, and `set_full_scb_dump()`.

13.74.3.16 full_scb_dump_split

```
bit cl_syoscb_cfg::full_scb_dump_split = 0b0 [private]
```

Controls whether SCB dumps (controlled by `full_scb_dump`) print all transactions in the same file, or if separate files are used for each queue.

Defaults to 1'b0

- 1'b0 => Dump the transactions of all the queues into the same file.
- 1'b1 => Dump the transactions of each queue in separate file.

Definition at line 103 of file `cl_syoscb_cfg.svh`.

Referenced by `get_full_scb_dump_split()`, and `set_full_scb_dump_split()`.

13.74.3.17 full_scb_dump_type

```
t_dump_type cl_syoscb_cfg::full_scb_dump_type = pk_syoscb::TXT [private]
```

File format used when dumping SCB contents to a logfile.

Defaults to TXT. Valid values are pk_syoscb::TXT and pk_syoscb::XML.

Definition at line 111 of file cl_syoscb_cfg.svh.

Referenced by get_full_scb_dump_type(), and set_full_scb_dump_type().

13.74.3.18 full_scb_max_queue_size

```
int unsigned cl_syoscb_cfg::full_scb_max_queue_size [private]
```

Controls the number of elements that a queue in the SCB can receive before transaction dumping starts.

Defaults to 0 (items are logged every time they are added to the SCB)

Definition at line 107 of file cl_syoscb_cfg.svh.

Referenced by get_full_scb_max_queue_size(), and set_full_scb_max_queue_size().

13.74.3.19 hash_compare_check

```
t_hash_compare_check cl_syoscb_cfg::hash_compare_check = pk_syoscb::SYOSCB_HASH_COMPARE_NO_VALIDATION [private]
```

Controls sanity check comparisons on hash queues.

- NO_VALIDATION => Does not perform any additional validations after finding an item in the secondary queue which matches the digest of the primary item.
- VALIDATE_MATCH => If an item is found in the secondary queue, compares the fields of the primary item to those of the secondary to validate the match.
- VALIDATE_NO_MATCH => If a match is not found in the secondary queue, performs ordinary comparison of the primary item to all items in the secondary queue. This may incur a significant performance hit due to the many additional comparisons.
- VALIDATE_ALL => Performs validation when matches are found and when matches are not found. Only used for hash-based queue implementations. Defaults to SYOSCB_HASH_COMPARE_NO_VALIDATION.

Definition at line 144 of file cl_syoscb_cfg.svh.

Referenced by get_hash_compare_check(), and set_hash_compare_check().

13.74.3.20 max_print_orphans

```
int cl_syoscb_cfg::max_print_orphans = 0 [private]
```

Select the maximum number of orphaned elements to print if any orphans are left in a queue after simulation.

Defaults to 0 (print everything). If set to -1, no orphans are printed. If set to 0, all orphans are printed. If set to a positive value N, prints up to N orphans from each queue. See also [dump_orphans_to_files](#) for the ability to log orphans into a file

Definition at line 64 of file cl_syoscb_cfg.svh.

Referenced by `get_max_print_orphans()`, `set_dump_orphans_to_files()`, and `set_max_print_orphans()`.

13.74.3.21 max_queue_size

```
int unsigned cl_syoscb_cfg::max_queue_size [private]
```

Maximum number of elements in each queue before an error is signaled.

0 means no limit (default). Indexed by queue name.

Definition at line 53 of file cl_syoscb_cfg.svh.

Referenced by `get_max_queue_size()`, `set_max_queue_size()`, and `set_queue()`.

13.74.3.22 max_search_window

```
int unsigned cl_syoscb_cfg::max_search_window [private]
```

The maximum number of entries to iterate through in a queue when performing OOO compare.

If no matches are found within the search window, it is registered as no match occurring. If `max_search_window == 0`, all items in a given queue are searched. The maximum search window can be set on a per-queue basis using [set_max_search_window\(\)](#) Defaults to 0 (search everything)

Definition at line 198 of file cl_syoscb_cfg.svh.

Referenced by `get_max_search_window()`, and `set_max_search_window()`.

13.74.3.23 mutexed_add_item_enable

```
bit cl_syoscb_cfg::mutexed_add_item_enable = 0b0 [private]
```

Controls whether `cl_syoscb::add_item()` should be mutexed or not.

Defaults to 1'b0 (not mutexed). When enabled, whenever an item is added to the SCB, the mutex `cl_syoscb::add_item_mutex` must be acquired. This ensures that no other items are added while scanning for a match, preserving queue order when iterating.

- 1'b0 => Adding items is not mutexed
- 1'b1 => Adding items is mutexed

Definition at line 205 of file `cl_syoscb_cfg.svh`.

Referenced by `get_mutexed_add_item_enable()`, and `set_mutexed_add_item_enable()`.

13.74.3.24 ordered_next

```
bit cl_syoscb_cfg::ordered_next = 0b1 [private]
```

Controls whether a strict item ordering should be used in assoc.

arrays in hash-based queue. Defaults to 1'b1.

- 1'b0 => Use the SystemVerilog implementation of the `next()` function for associative arrays in the hash queue implementations. This does not guarantee the order to insertion order For OOO compares using the hash queues this is an option which makes the OOO compare perform at its maximum
- 1'b1 => Guarantee the order of insertions by maintaining some metadata. The OOO compare with hashed queues take a minor performance hit when this is enabled. Only valid for hash based queue implementations. Defaults to 1'b1 (guaranteed order of insertions)

Definition at line 132 of file `cl_syoscb_cfg.svh`.

Referenced by `get_ordered_next()`, and `set_ordered_next()`.

13.74.3.25 orphan_dump_type

```
t_dump_type cl_syoscb_cfg::orphan_dump_type = pk_syoscb::TXT [private]
```

File format used when dumping orphans to logfiles.

Default to TXT Valid values are `pk_syoscb::TXT` and `pk_syoscb::XML` If set to XML, orphan dump logfiles will use .xml extension instead of .log

Definition at line 116 of file `cl_syoscb_cfg.svh`.

Referenced by `get_orphan_dump_type()`, and `set_orphan_dump_type()`.

13.74.3.26 primary_queue

```
string cl_syoscb_cfg::primary_queue [private]
```

Name of the primary queue used in this scoreboard.

If set to an empty string, the primary queue is dynamically selected when performing comparisons (takes the shortest queue)

Definition at line 15 of file `cl_syoscb_cfg.svh`.

Referenced by `get_primary_queue()`, and `set_primary_queue()`.

13.74.3.27 print_cfg

```
bit cl_syoscb_cfg::print_cfg = 0b0 [private]
```

Controls whether the scoreboard's configuration values should be printed in the `cl_syoscb::build_phase()`.

Defaults to 1'b0 (disable).

- 1'b0 => Disable print of cfg configuration in `cl_syoscb::build_phase()`
- 1'b1 => Enable print of cfg configuration in `cl_syoscb::build_phase()`

Definition at line 150 of file `cl_syoscb_cfg.svh`.

Referenced by `get_print_cfg()`, and `set_print_cfg()`.

13.74.3.28 print_orphans_as_errors

```
bit cl_syoscb_cfg::print_orphans_as_errors = 0b0 [private]
```

Controls whether orphaned items in the queues should be treated as errors when printing at the end of simulation.

Defaults to 1'b0. -1'b0 => Orphans are printed with UVM_INFO -1'b1 => Orphans are printed as UVM_ERRORS

Definition at line 58 of file `cl_syoscb_cfg.svh`.

Referenced by `get_orphans_as_errors()`, and `set_orphans_as_errors()`.

13.74.3.29 printer_verbosity

```
bit cl_syoscb_cfg::printer_verbosity [private]
```

Associative array holding the printer verbosity bit for a specific queue/producer combination.

This verbosity bit controls the number of elements to be printed at the start/end of a list in a tx item. If no entry has been set for a specific queue/producer combination the value of [default_printer_verbosity](#) is used.

- 1'b0 => Number of elements at the head and at the end of a list is 5 (unless changed with `cl_syoscb_cfg::printer_config::set_printer_begin/end_elements`)
- 1'b1 => No maximum number of elements (the entire list contents are printed)

Definition at line 179 of file `cl_syoscb_cfg.svh`.

Referenced by `get_printer_verbosity()`, and `set_printer_verbosity()`.

13.74.3.30 printers

```
uvm_printer cl_syoscb_cfg::printers [private]
```

Associative array holding handles to printers used for a specific queue/producer combination.

If no printer has been set for a specific queue/producer combination, uses the printer set in [default_printer](#).

Definition at line 187 of file `cl_syoscb_cfg.svh`.

Referenced by `get_printer()`, and `set_printer()`.

13.74.3.31 producers

```
cl_syoscb_cfg_pl cl_syoscb_cfg::producers [private]
```

Associative array indexed by producer name.

Returns the list of queues which this producer is related to.

Definition at line 11 of file `cl_syoscb_cfg.svh`.

Referenced by `exist_producer()`, `get_max_length_producer()`, `get_producer()`, `get_producers()`, `init()`, and `set_producer()`.

13.74.3.32 queue_stat_interval

```
int unsigned cl_syoscb_cfg::queue_stat_interval [private]
```

Defines an interval value N for each queue, such that the queue's statistics are printed on every N'th insertion.

All entries default to 0 (stat printouts disabled)

- 0 => Printing stats are disabled for the given queue
- N>0 => The given queue's stats are printed every N insertions into the queue.

Definition at line 211 of file `cl_syoscb_cfg.svh`.

Referenced by `get_queue_stat_interval()`, and `set_queue_stat_interval()`.

13.74.3.33 scb_stat_interval

```
int unsigned cl_syoscb_cfg::scb_stat_interval = 0 [private]
```

Defines an interval value N, similar to [queue_stat_interval](#), causing a printout of all queue stats in the SCB on every N'th insertion.

Default value is 0 (stat printout disabled)

- 0 => Printing SCB stats is disabled
- N>0 => The SCB stats are printed after every N insertions into the SCB.

Definition at line 217 of file `cl_syoscb_cfg.svh`.

Referenced by `get_scb_stat_interval()`, and `set_scb_stat_interval()`.

13.74.3.34 trigger_greediness

```
t_scb_compare_greed cl_syoscb_cfg::trigger_greediness = pk_syoscb::SYOSCB_COMPARE_NOT_GREEDY [private]
```

Defines the greed level for comparison operations.

Defaults to `pk_syoscb::SYOSCB_COMPARE_NOT_GREEDY` The greed level controls whether a comparison trigger will attempt to drain the SCB by performing additional comparisons if the previous comparison was successful (greedy) or if only a single comparison is performed when triggered (not greedy)

Definition at line 30 of file `cl_syoscb_cfg.svh`.

Referenced by `get_trigger_greediness()`, and `set_trigger_greediness()`.

The documentation for this class was generated from the following files:

- `cl_syoscb_cfg.svh`
- `pk_syoscb.sv`

13.75 cl_syoscb_cfg_pl Class Reference

Utility class for capturing the queue names associated with a producer.

Inherits uvm_object, and uvm_object.

Public Member Functions

- virtual void [set_list](#) (string [list](#)[])
Sets the list of queue names associated with a producer.
- virtual bit [exists](#) (string queue)
Checks whether a given queue is connected to the producer that this object represents.

Public Attributes

- string [list](#) []
The list of queue names connected to the producer that this _pl represents.

13.75.1 Detailed Description

Utility class for capturing the queue names associated with a producer.

Definition at line 2 of file cl_syoscb_cfg_pl.svh.

13.75.2 Member Function Documentation

13.75.2.1 exists()

```
bit cl_syoscb_cfg_pl::exists (
    string queue ) [virtual]
```

Checks whether a given queue is connected to the producer that this object represents.

Parameters

<i>queue</i>	The name of the queue to check
--------------	--------------------------------

Definition at line 39 of file cl_syoscb_cfg_pl.svh.

References list.

Referenced by cl_syoscb_cfg::exist_producer(), cl_syoscb_cfg::set_comparer(), and cl_syoscb_cfg::set_printer().

The documentation for this class was generated from the following files:

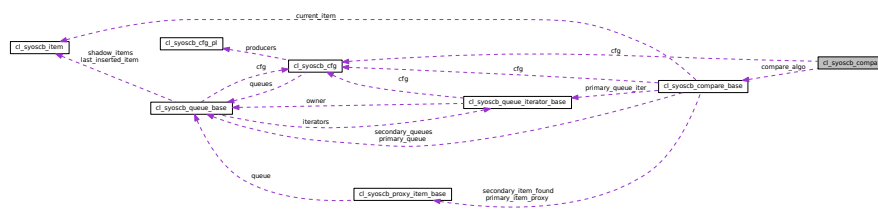
- `cl_syoscb_cfg_pl.svh`
- `pk_syoscb.sv`

13.76 cl_syoscb_compare Class Reference

Component which instantiates the chosen comparison algorithm.

Inherits `uvm_component`, and `uvm_component`.

Collaboration diagram for `cl_syoscb_compare`:



Public Member Functions

- void `build_phase` (`uvm_phase` phase)
UVM build phase: Gets the scoreboard's configuration and creates the comparison algorithm.
- void `extract_phase` (`uvm_phase` phase)
UVM extract phase: Check if `cl_syoscb_cfg::end_greediness` is greedy.
- virtual void `compare_trigger` (string queue_name="", `cl_syoscb_item` item=null)
Compare API: Starts a comparison by invoking the chosen compare strategy if comparisons are not disabled
- virtual void `compare_control` (bit cc)
Compare API: Toggle comparisons on or off

Private Attributes

- `cl_syoscb_cfg` `cfg`
Handle to the configuration.
- `cl_syoscb_compare_base` `compare_algo`
Handle to the actual compare algorithm to be used.

13.76.1 Detailed Description

Component which instantiates the chosen comparison algorithm.

Serves to wrap the compare algorithm in a UVM component, as well as triggering additional comparisons at the end of the run phase if the greed level prescribes this.

Definition at line 4 of file `cl_syoscb_compare.svh`.

13.76.2 Member Function Documentation

13.76.2.1 compare_control()

```
void cl_syoscb_compare::compare_control (
    bit cc ) [virtual]
```

Compare API: Toggle comparisons on or off

Parameters

<i>cc</i>	compare control bit. If 1, comparisons are enabled, if 0, comparisons are disabled
-----------	--

Definition at line 71 of file cl_syoscb_compare.svh.

References `compare_algo`, and `cl_syoscb_compare_base::compare_control()`.

Referenced by `cl_syoscb::compare_control()`.

13.76.2.2 compare_trigger()

```
void cl_syoscb_compare::compare_trigger (
    string queue_name = "",
    cl_syoscb_item item = null ) [virtual]
```

Compare API: Starts a comparison by invoking the chosen compare strategy if comparisons are not disabled

Parameters

<i>queue_name</i>	Name of the queue which had an item inserted into it
<i>item</i>	The scoreboard wrapper item that was inserted into the SCB

Definition at line 65 of file cl_syoscb_compare.svh.

References `compare_algo`, and `cl_syoscb_compare_base::compare_trigger()`.

Referenced by `cl_syoscb::compare_trigger()`.

13.76.2.3 extract_phase()

```
void cl_syoscb_compare::extract_phase (
    uvm_phase phase )
```

UVM extract phase: Check if `cl_syoscb_cfg::end_greediness` is greedy.

If yes, we want to drain all the remaining matches from the scb before moving to check_phase

Definition at line 56 of file `cl_syoscb_compare.svh`.

References `cfg`, `compare_algo`, `cl_syoscb_compare_base::compare_main()`, and `cl_syoscb_cfg::get_end_greediness()`.

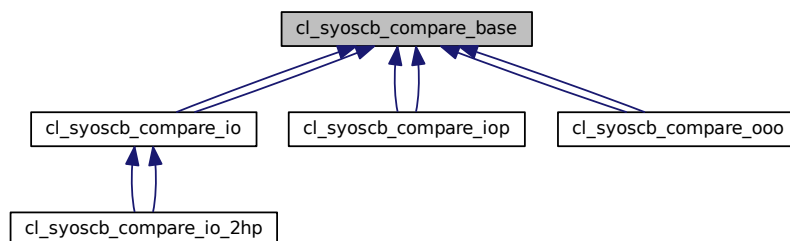
The documentation for this class was generated from the following files:

- `cl_syoscb_compare.svh`
- `pk_syoscb.sv`

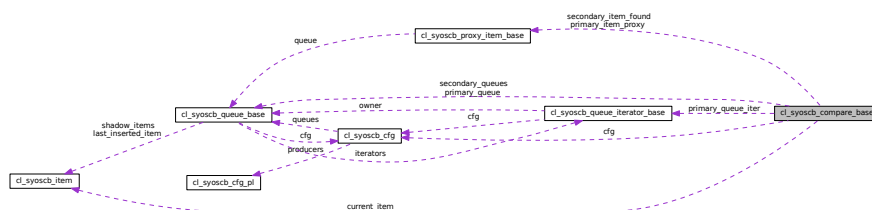
13.77 cl_syoscb_compare_base Class Reference

Base class for all compare algorithms.

Inheritance diagram for `cl_syoscb_compare_base`:



Collaboration diagram for `cl_syoscb_compare_base`:



Public Member Functions

- virtual void [compare_control](#) (bit cc)
Compare API: Toggle comparisons on or off
- virtual void [compare_trigger](#) (string queue_name="", [cl_syoscb_item](#) item=null)
Compare API: Starts a comparison by calling [compare_main](#) if comparisons are not disabled.
- virtual void [compare_main](#) (t_scb_compare_greed greed)
Compare API: Main function that contains all the actual compare operations requested by the compare algorithm.
- virtual void [set_cfg](#) ([cl_syoscb_cfg](#) cfg)
Set the scoreboard configuration associated with this comparer's scoreboard.
- virtual [cl_syoscb_cfg](#) [get_cfg](#) ()
Gets the scoreboard configuration object associated with this scoreboard.
- virtual string [generate_miscomp_table](#) ([cl_syoscb_item](#) primary_item, [cl_syoscb_item](#) secondary_item, string sec_queue_name, uvm_comparer comparer, string cmp_name)
Generates a side-by-side comparison of the seq.
- virtual void [do_copy](#) (uvm_object rhs)
Custom do_dopy implementation for secondary queues.

Protected Member Functions

- virtual void [init](#) ()
Compare Strategy API: Executes some preliminary common operations before starting comparisons:
 1. Split queues into primary and secondary
 2. Create iterator for the chosen primary queue
- virtual void [compare_do_greed](#) (t_scb_compare_greed greed)
Compare Strategy API: Try to remove a match and drain all the potential remaining matches inside the queues according to the greed level given as argument.
- virtual void [compare_init](#) ()
Compare Strategy API: Verifies if the conditions for starting a compare are met:
 1. Verify that all queues currently contain at least one element
 2. Verify that all queues have at least one element from the same producer as the producer returned by [get_count_producer\(\)](#) (the primary item being searched for) If the conditions are met then go variable is triggered, and the compare process can start.
- virtual void [compare_do](#) ()
Compare Strategy API: Starts the actual comparison operation
 1. Perform initialization on the primary queue, if necessary
 2. Start the primary queue loop
- virtual string [get_primary_queue_name](#) ()
Compare Strategy API: Gets the name of this scoreboard's primary queue.
- virtual void [split_queues](#) ()
Compare Strategy API: Splits the scoreboard's queues into 1 primary queue and N-1 secondary queues.
- virtual void [check_queues](#) ()
Compare Strategy API: Check if any queue is empty.
- virtual void [count_producers](#) (string producer="")
Compare Strategy API: Checks if the producer of the current item exists in all other queues, and whether all other queues have at least 1 item from that producer.
- virtual void [create_primary_iterator](#) ()
Compare Strategy API: Creates the iterator for the primary queue and sets the pointer to its first element
- virtual void [primary_loop_init](#) ()
Compare Strategy API: Contains all the operations to be executed immediately before starting the primary loop.
- virtual void [primary_loop_do](#) ()

Compare Strategy API: Loop over the primary queue, selecting primary items to compare against items in the secondary queues.

- virtual void [secondary_loop_do](#) ()

Compare Strategy API: Loop over all secondary queues to find a match for the primary item.

- virtual void [static_queue_split_do](#) ()

Compare Strategy API: Splits queues into primary and secondary when a primary queue has been specified.

- virtual void [dynamic_queue_split_do](#) ()

Compare Strategy API: Splits queues into primary and secondary when a primary queue has not been specified.

- virtual bit [delete](#) ()

Compare Strategy API: Deletes matched items from the primary and all secondary queues if a match was found.

- virtual string [get_count_producer](#) ()

Compare Strategy API: Returns the name of the producer that the compare method should evaluate in order to verify if it makes sense to start a comparison.

- virtual int unsigned [get_queues_item_cnt](#) ()

Compare Strategy API: Gets the total number of items in all the queues at the moment of the function call.

Protected Attributes

- [cl_syoscb_cfg](#) [cfg](#)

Handle to the configuration object.

- bit [do_split](#) = 0b1

Indicates how queues should be split into a primary queue and array of secondary queues.

- bit [go](#) = 0b1

Indicates whether a comparison can be started (1) or not (0)

- bit [disable_compare](#) = 0b0

If set to 1'b1, no comparisons are performed. If 1'b0, comparisons are executed.

- string [primary_queue_name](#)

Name of primary queue.

- [cl_syoscb_queue_base](#) [primary_queue](#)

Handle to primary queue.

- string [secondary_queue_names](#) []

Names of secondary queues.

- [cl_syoscb_queue_base](#) [secondary_queues](#) []

Handles to secondary queues.

- [cl_syoscb_proxy_item_base](#) [secondary_item_found](#) [string]

Associative array used to indicate if a matching item was found in a secondary queue.

- [cl_syoscb_proxy_item_base](#) [primary_item_proxy](#)

Proxy item for the item being searched for in all secondary queue.

- [cl_syoscb_queue_iterator_base](#) [primary_queue_iter](#)

Iterator into primary queue.

- string [current_queue_name](#)

Name of the queue currently being searched.

- [cl_syoscb_item](#) [current_item](#)

Handle to the item passed in by [cl_syoscb::add_item](#).

Private Member Functions

- virtual int [num_uvm_errors](#) ()

Returns the number of UVM_ERROR messages that have been generated so far.

13.77.1 Detailed Description

Base class for all compare algorithms.

The chosen compare algorithm defines how matches are found. For more information on the comparison algorithms included with the SyoSil UVM Scoreboard, see [Compare implementation notes](#).

Definition at line 4 of file cl_syoscb_compare_base.svh.

13.77.2 Member Function Documentation

13.77.2.1 check_queues()

```
void cl_syoscb_compare_base::check_queues ( ) [protected], [virtual]
```

Compare Strategy API: Check if any queue is empty.

Assigns 0 to the member variable [go](#) when any of the queues are empty, indicating that a comparison cannot be started. Assigns 1 if all queues are non-empty, indicating that the comparison may be started.

Definition at line 250 of file cl_syoscb_compare_base.svh.

References [cl_syoscb_queue_base::empty\(\)](#), [go](#), [primary_queue](#), and [secondary_queues](#).

Referenced by [cl_syoscb_compare_iop::compare_init\(\)](#), and [compare_init\(\)](#).

13.77.2.2 compare_control()

```
void cl_syoscb_compare_base::compare_control (
    bit cc ) [virtual]
```

Compare API: Toggle comparisons on or off

Parameters

<code>cc</code>	compare control bit. If 1, comparisons are enabled, if 0, comparisons are disabled
-----------------	--

Definition at line 114 of file cl_syoscb_compare_base.svh.

References [disable_compare](#).

Referenced by [cl_syoscb_compare::compare_control\(\)](#).

13.77.2.3 `compare_do_greed()`

```
void cl_syoscb_compare_base::compare_do_greed (
    t_scb_compare_greed greed ) [protected], [virtual]
```

Compare Strategy API: Try to remove a match and drain all the potential remaining matches inside the queues according to the greed level given as argument.

Performs the following:

1. Calling the checkers in order to verify that starting a comparison makes sense
2. Calling the actual `compare_do` function if a comparison should be starte
3. Looping to remove additional matches if the greed levels prescribes this

Parameters

<i>greed</i>	The greed level to use when performing comparisons. See cl_syoscb_cfg::trigger_greediness
--------------	---

Definition at line 165 of file `cl_syoscb_compare_base.svh`.

References `cfg`, `compare_do()`, `compare_init()`, `disable_compare`, `cl_syoscb_cfg::get_disable_compare_after_↵` `error()`, `get_queues_item_cnt()`, `go`, `num_uvm_errors()`, and `secondary_queues`.

Referenced by `compare_main()`.

13.77.2.4 `compare_init()`

```
void cl_syoscb_compare_base::compare_init ( ) [protected], [virtual]
```

Compare Strategy API: Verifies if the conditions for starting a compare are met:

1. Verify that all queues currently contain at least one element
2. Verify that all queues have at least one element from the same producer as the producer returned by [get_count_producer\(\)](#) (the primary item being searched for) If the conditions are met then go variable is triggered, and the compare process can start.

Reimplemented in [cl_syoscb_compare_iop](#), and [cl_syoscb_compare_iop](#).

Definition at line 196 of file `cl_syoscb_compare_base.svh`.

References `check_queues()`, and `count_producers()`.

Referenced by `compare_do_greed()`.

13.77.2.5 compare_main()

```
void cl_syoscb_compare_base::compare_main (
    t_scb_compare_greed greed ) [virtual]
```

Compare API: Main function that contains all the actual compare operations requested by the compare algorithm.

It cares about:

1. Splitting queues into primary and secondary queues, generating an iterator into the primary queue
2. Calling [compare_do_greed](#) with the proper draining value passed as argument
3. Deleting the primary queue iterator after the compare algo has finished all comparisons

Parameters

<i>greed</i>	The greed level to use when performing comparisons. See cl_syoscb_cfg::trigger_greediness
--------------	---

Definition at line 142 of file `cl_syoscb_compare_base.svh`.

References [compare_do_greed\(\)](#), and [init\(\)](#).

Referenced by [compare_trigger\(\)](#), and [cl_syoscb_compare::extract_phase\(\)](#).

13.77.2.6 compare_trigger()

```
void cl_syoscb_compare_base::compare_trigger (
    string queue_name = "",
    cl_syoscb_item item = null ) [virtual]
```

Compare API: Starts a comparison by calling [compare_main](#) if comparisons are not disabled.

Parameters

<i>queue_name</i>	Name of the queue which had an item inserted into it
<i>item</i>	The scoreboard wrapper item that was inserted into the SCB

Definition at line 121 of file `cl_syoscb_compare_base.svh`.

References [cfg](#), [compare_main\(\)](#), [current_item](#), [current_queue_name](#), [disable_compare](#), and [cl_syoscb_cfg::get_trigger_greediness\(\)](#).

Referenced by [cl_syoscb_compare::compare_trigger\(\)](#).

13.77.2.7 count_producers()

```
void cl_syoscb_compare_base::count_producers (
    string producer = "" ) [protected], [virtual]
```

Compare Strategy API: Checks if the producer of the current item exists in all other queues, and whether all other queues have at least 1 item from that producer.

If true, assigns 1'b1 to member variable [go](#) If false, assigns 1'b0 to member variable [go](#)

Parameters

<i>producer</i>	The producer to check if exists in all other queues. If not set, checks if the producer of current_item exists in other queues. If set, checks for that producer in other queues.
-----------------	---

Reimplemented in [cl_syoscb_compare_io](#), and [cl_syoscb_compare_io](#).

Definition at line 271 of file `cl_syoscb_compare_base.svh`.

References `cl_syoscb_queue_base::exists_cnt_producer()`, `cl_syoscb_queue_base::get_cnt_producer()`, `get_↔count_producer()`, `go`, `primary_queue`, and `secondary_queues`.

Referenced by `compare_init()`, and `cl_syoscb_compare_iop::primary_loop_do()`.

13.77.2.8 delete()

```
bit cl_syoscb_compare_base::delete ( ) [protected], [virtual]
```

Compare Strategy API: Deletes matched items from the primary and all secondary queues if a match was found.

If no match is found, no items are deleted from the queues.

Returns

0b1 if a match was found and items were deleted, 10b0 otherwise

Definition at line 303 of file `cl_syoscb_compare_base.svh`.

References `cfg`, `cl_syoscb_queue_base::delete_item()`, `cl_syoscb_cfg::get_scb_name()`, `primary_queue`, `primary_↔queue_name`, `secondary_item_found`, `secondary_queue_names`, and `secondary_queues`.

13.77.2.9 dynamic_queue_split_do()

```
void cl_syoscb_compare_base::dynamic_queue_split_do ( ) [protected], [virtual]
```

Compare Strategy API: Splits queues into primary and secondary when a primary queue has not been specified.

Selects as the primary queue the shortest queue, the rest are the secondary queues.

Definition at line 398 of file `cl_syoscb_compare_base.svh`.

References `cfg`, `cl_syoscb_cfg::get_queue()`, `cl_syoscb_cfg::get_queues()`, `cl_syoscb_queue_base::get_size()`, `primary_queue`, `primary_queue_name`, `secondary_queue_names`, and `secondary_queues`.

Referenced by `split_queues()`.

13.77.2.10 generate_miscomp_table()

```
string cl_syoscb_compare_base::generate_miscomp_table (
    cl_syoscb_item primary_item,
    cl_syoscb_item secondary_item,
    string sec_queue_name,
    uvm_comparer comparer,
    string cmp_name ) [virtual]
```

Generates a side-by-side comparison of the seq.

items that prompted a miscompare. The table includes a header with information on which queues the items originated in, a side-by-side view of the two seq. items and, if [cl_syoscb_cfg::enable_comparer_report](#) is set, it also includes a number of miscompare descriptions from the uvm_comparer used.

Parameters

<i>primary_item</i>	The primary item in the comparison
<i>secondary_item</i>	The secondary item in the comparison
<i>sec_queue_name</i>	The name of the secondary queue
<i>comparer</i>	The uvm_comparer used for the comparison
<i>cmp_name</i>	Name of the comparison type, to be used when printing the header.

Returns

The miscompare table

Definition at line 496 of file `cl_syoscb_compare_base.svh`.

References `cfg`, `cl_syoscb_string_library::generate_cmp_table_body()`, `cl_syoscb_string_library::generate_cmp_table_footer()`, `cl_syoscb_string_library::generate_cmp_table_header()`, `cl_syoscb_cfg::get_enable_comparer_report()`, `cl_syoscb_item::get_producer()`, `cl_syoscb_cfg::get_scb_name()`, and `primary_queue_name`.

Referenced by `cl_syoscb_compare_io_2hp::primary_loop_do()`, `cl_syoscb_compare_io::secondary_loop_do()`, and `cl_syoscb_compare_iop::secondary_loop_do()`.

13.77.2.11 get_count_producer()

```
string cl_syoscb_compare_base::get_count_producer ( ) [protected], [virtual]
```

Compare Strategy API: Returns the name of the producer that the compare method should evaluate in order to verify if it makes sense to start a comparison.

Note

This needs to be overridden by the derived compare methods in order to change the behaviour accordingly to the requested needs. By default, the function returns the producer of [current_item](#).

Returns

The name producer which should be evaluated

Reimplemented in [cl_syoscb_compare_iop](#), [cl_syoscb_compare_iop](#), [cl_syoscb_compare_ooo](#), and [cl_syoscb_compare_ooo](#).

Definition at line 336 of file `cl_syoscb_compare_base.svh`.

References `current_item`, and `cl_syoscb_item::get_producer()`.

Referenced by `count_producers()`.

13.77.2.12 `get_primary_queue_name()`

```
string cl_syoscb_compare_base::get_primary_queue_name ( ) [protected], [virtual]
```

Compare Strategy API: Gets the name of this scoreboard's primary queue.

Convenience method wrapping [cl_syoscb_cfg::get_primary_queue](#)

Definition at line 213 of file `cl_syoscb_compare_base.svh`.

References `cfg`, and `cl_syoscb_cfg::get_primary_queue()`.

Referenced by `static_queue_split_do()`.

13.77.2.13 `get_queues_item_cnt()`

```
int unsigned cl_syoscb_compare_base::get_queues_item_cnt ( ) [protected], [virtual]
```

Compare Strategy API: Gets the total number of items in all the queues at the moment of the function call.

Returns

Number of items currently stored in all queues

Definition at line 343 of file `cl_syoscb_compare_base.svh`.

References `cfg`, `cl_syoscb_cfg::get_queue()`, `cl_syoscb_cfg::get_queues()`, and `cl_syoscb_queue_base::get_size()`.

Referenced by `compare_do_greed()`.

13.77.2.14 `primary_loop_do()`

```
void cl_syoscb_compare_base::primary_loop_do ( ) [protected], [virtual]
```

Compare Strategy API: Loop over the primary queue, selecting primary items to compare against items in the secondary queues.

Note

Abstract method. This method must be implemented in a subclass.

Reimplemented in [cl_syoscb_compare_iop](#), [cl_syoscb_compare_iop](#), [cl_syoscb_compare_io](#), [cl_syoscb_compare_io_2hp](#), [cl_syoscb_compare_io](#), [cl_syoscb_compare_io_2hp](#), [cl_syoscb_compare_ooo](#), and [cl_syoscb_compare_ooo](#).

Definition at line 361 of file `cl_syoscb_compare_base.svh`.

Referenced by `compare_do()`.

13.77.2.15 primary_loop_init()

```
void cl_syoscb_compare_base::primary_loop_init ( ) [protected], [virtual]
```

Compare Strategy API: Contains all the operations to be executed immediately before starting the primary loop.

By default is an empty function (no other operations needed).

Definition at line 463 of file cl_syoscb_compare_base.svh.

Referenced by cl_syoscb_compare_io_2hp::compare_do(), and compare_do().

13.77.2.16 secondary_loop_do()

```
void cl_syoscb_compare_base::secondary_loop_do ( ) [protected], [virtual]
```

Compare Strategy API: Loop over all secondary queues to find a match for the primary item.

Note

Abstract method. This method must be implemented in a subclass.

Reimplemented in [cl_syoscb_compare_iop](#), [cl_syoscb_compare_iop](#), [cl_syoscb_compare_io](#), [cl_syoscb_compare_io](#), [cl_syoscb_compare_ooo](#), and [cl_syoscb_compare_ooo](#).

Definition at line 367 of file cl_syoscb_compare_base.svh.

13.77.2.17 set_cfg()

```
void cl_syoscb_compare_base::set_cfg (
    cl_syoscb_cfg cfg ) [virtual]
```

Set the scoreboard configuration associated with this comparer's scoreboard.

Parameters

<i>cfg</i>	The scoreboard configuration object
------------	-------------------------------------

Definition at line 476 of file cl_syoscb_compare_base.svh.

References [cfg](#).

Referenced by cl_syoscb_compare::build_phase().

13.77.2.18 split_queues()

```
void cl_syoscb_compare_base::split_queues ( ) [protected], [virtual]
```

Compare Strategy API: Splits the scoreboard's queues into 1 primary queue and N-1 secondary queues.

Selects the primary queue and creates an array of secondary queues with the rest. If a dynamic primary queue is used, this split is performed every time a comparison is started. If a static primary queue is used, this split is only performed on the first comparison.

Definition at line 221 of file cl_syoscb_compare_base.svh.

References `cfg`, `do_split`, `cl_syoscb_cfg::dynamic_primary_queue()`, `dynamic_queue_split_do()`, `secondary_queue_names`, `secondary_queues`, and `static_queue_split_do()`.

Referenced by `init()`.

13.77.2.19 static_queue_split_do()

```
void cl_syoscb_compare_base::static_queue_split_do ( ) [protected], [virtual]
```

Compare Strategy API: Splits queues into primary and secondary when a primary queue has been specified.

The primary queue is the one set by `cl_syoscb_cfg::set_primary_queue_name`, all other queues will be secondary queues

Definition at line 374 of file cl_syoscb_compare_base.svh.

References `cfg`, `get_primary_queue_name()`, `cl_syoscb_cfg::get_queue()`, `cl_syoscb_cfg::get_queues()`, `primary_queue`, `primary_queue_name`, `secondary_queue_names`, and `secondary_queues`.

Referenced by `split_queues()`.

13.77.3 Member Data Documentation

13.77.3.1 do_split

```
bit cl_syoscb_compare_base::do_split = 0b1 [protected]
```

Indicates how queues should be split into a primary queue and array of secondary queues.

This is done once with a static primary queue, done every time compare is invoked with a dynamic primary queue

Definition at line 13 of file cl_syoscb_compare_base.svh.

Referenced by `init()`, and `split_queues()`.

13.77.3.2 secondary_item_found

`cl_syoscb_proxy_item_base` `cl_syoscb_compare_base::secondary_item_found` [protected]

Associative array used to indicate if a matching item was found in a secondary queue.

If matches are found in all secondary queues, all items are removed from their respective queues

Definition at line 28 of file `cl_syoscb_compare_base.svh`.

Referenced by `delete()`, `cl_syoscb_compare_ooo::primary_loop_do()`, `cl_syoscb_compare_io::primary_loop_do()`, `cl_syoscb_compare_io_2hp::primary_loop_do()`, `cl_syoscb_compare_iop::primary_loop_do()`, `cl_syoscb_compare_ooo::secondary_loop_do()`, `cl_syoscb_compare_io::secondary_loop_do()`, and `cl_syoscb_compare_iop::secondary_loop_do()`.

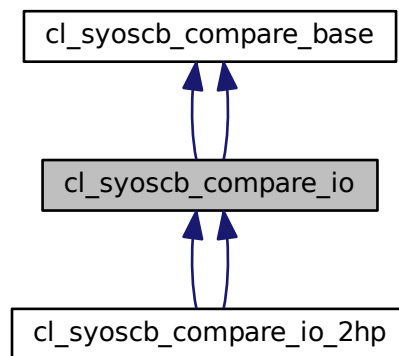
The documentation for this class was generated from the following files:

- `cl_syoscb_compare_base.svh`
- `pk_syoscb.sv`

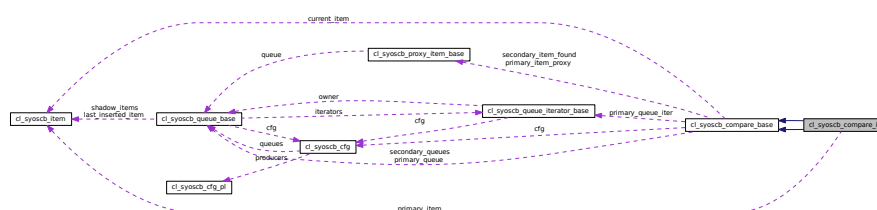
13.78 cl_syoscb_compare_io Class Reference

Implementation of the in-order comparison algorithm for N queues.

Inheritance diagram for `cl_syoscb_compare_io`:



Collaboration diagram for `cl_syoscb_compare_io`:



Public Attributes

- [cl_syoscb_item primary_item](#)
Scoreboard wrapper item from the primary queue.

Protected Member Functions

- virtual void [primary_loop_do](#) ()
Compare Strategy API: Implementation of the in-order comparison algorithm.
- virtual void [count_producers](#) (string producer="")
Compare Strategy API: Checks if the producer of the current item exists in all other queues, and whether all other queues have at least 1 item from that producer.
- virtual void [secondary_loop_do](#) ()
Compare Strategy API: Loop through all the secondary queues, checking if the first item in that secondary queues matches the first in the primary queue.
- virtual void [primary_loop_do](#) ()
Compare Strategy API: Loop over the primary queue, selecting primary items to compare against items in the secondary queues.
- virtual void [count_producers](#) (string producer="")
Compare Strategy API: Checks if the producer of the current item exists in all other queues, and whether all other queues have at least 1 item from that producer.
- virtual void [secondary_loop_do](#) ()
Compare Strategy API: Loop over all secondary queues to find a match for the primary item.

Additional Inherited Members

13.78.1 Detailed Description

Implementation of the in-order comparison algorithm for N queues.

Definition at line 2 of file `cl_syoscb_compare_io.svh`.

13.78.2 Member Function Documentation

13.78.2.1 [count_producers\(\)](#) [1/2]

```
void cl_syoscb_compare_io::count_producers (
    string producer = "" ) [protected], [virtual]
```

Compare Strategy API: Checks if the producer of the current item exists in all other queues, and whether all other queues have at least 1 item from that producer.

If true, assigns 1'b1 to member variable [go](#) If false, assigns 1'b0 to member variable [go](#)

Parameters

<i>producer</i>	The producer to check if exists in all other queues. If not set, checks if the producer of current_item exists in other queues. If set, checks for that producer in other queues.
-----------------	---

Reimplemented from [cl_syoscb_compare_base](#).

Definition at line 63 of file cl_syoscb_compare_io.svh.

13.78.2.2 count_producers() [2/2]

```
virtual void cl_syoscb_compare_io::count_producers (
    string producer = "" ) [protected], [virtual]
```

Compare Strategy API: Checks if the producer of the current item exists in all other queues, and whether all other queues have at least 1 item from that producer.

If true, assigns 1'b1 to member variable [go](#) If false, assigns 1'b0 to member variable [go](#)

Parameters

<i>producer</i>	The producer to check if exists in all other queues. If not set, checks if the producer of current_item exists in other queues. If set, checks for that producer in other queues.
-----------------	---

Reimplemented from [cl_syoscb_compare_base](#).

13.78.2.3 primary_loop_do() [1/2]

```
void cl_syoscb_compare_io::primary_loop_do ( ) [protected], [virtual]
```

Compare Strategy API: Implementation of the in-order comparison algorithm.

In the primary loop, the algorithm extracts the oldest inserted element from the primary queue, and then starts looping over all secondary queues to find a matching item in [secondary_loop_do](#). If matching items are found, these are removed from all of the queues If no matching items are found, a miscompare is generated and a UVM↵_ERROR is issued.

Reimplemented from [cl_syoscb_compare_base](#).

Reimplemented in [cl_syoscb_compare_io_2hp](#), and [cl_syoscb_compare_io_2hp](#).

Definition at line 39 of file cl_syoscb_compare_io.svh.

References [cl_syoscb_compare_base::cfg](#), [cl_syoscb_queue_base::get_item\(\)](#), [cl_syoscb_cfg::get_scb_name\(\)](#), [cl_syoscb_queue_iterator_base::next\(\)](#), [primary_item](#), [cl_syoscb_compare_base::primary_item_proxy](#), [cl_↵syoscb_compare_base::primary_queue](#), [cl_syoscb_compare_base::primary_queue_iter](#), [cl_syoscb_compare_↵base::secondary_item_found](#), [secondary_loop_do\(\)](#), and [cl_syoscb_string_library::sprint_item\(\)](#).

13.78.2.4 `primary_loop_do()` [2/2]

```
virtual void cl_syoscb_compare_io::primary_loop_do ( ) [protected], [virtual]
```

Compare Strategy API: Loop over the primary queue, selecting primary items to compare against items in the secondary queues.

Note

Abstract method. This method must be implemented in a subclass.

Reimplemented from [cl_syoscb_compare_base](#).

Reimplemented in [cl_syoscb_compare_io_2hp](#), and [cl_syoscb_compare_io_2hp](#).

13.78.2.5 `secondary_loop_do()` [1/2]

```
void cl_syoscb_compare_io::secondary_loop_do ( ) [protected], [virtual]
```

Compare Strategy API: Loop through all the secondary queues, checking if the first item in that secondary queues matches the first in the primary queue.

If a match is found, this is recorded in `cl_syoscb_compare_base::secondary_items_found`

Reimplemented from [cl_syoscb_compare_base](#).

Definition at line 74 of file `cl_syoscb_compare_io.svh`.

References `cl_syoscb_compare_base::cfg`, `cl_syoscb_queue_base::create_iterator()`, `cl_syoscb_queue_iterator_base::first()`, `cl_syoscb_compare_base::generate_miscomp_table()`, `cl_syoscb_cfg::get_comparer()`, `cl_syoscb_cfg::get_default_comparer()`, `cl_syoscb_queue_base::get_item()`, `cl_syoscb_queue_base::get_iterator()`, `cl_syoscb_item::get_producer()`, `cl_syoscb_cfg::get_scb_name()`, `cl_syoscb_queue_iterator_base::next()`, `primary_item`, `cl_syoscb_compare_base::primary_queue_name`, `cl_syoscb_compare_base::secondary_item_found`, `cl_syoscb_compare_base::secondary_queue_names`, `cl_syoscb_compare_base::secondary_queues`, and `cl_syoscb_string_library::sprint_item()`.

Referenced by `primary_loop_do()`.

13.78.2.6 `secondary_loop_do()` [2/2]

```
virtual void cl_syoscb_compare_io::secondary_loop_do ( ) [protected], [virtual]
```

Compare Strategy API: Loop over all secondary queues to find a match for the primary item.

Note

Abstract method. This method must be implemented in a subclass.

Reimplemented from [cl_syoscb_compare_base](#).

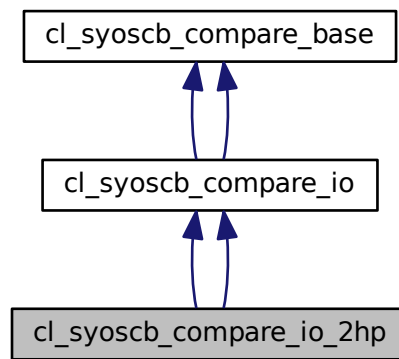
The documentation for this class was generated from the following files:

- `cl_syoscb_compare_io.svh`
- `pk_syoscb.sv`

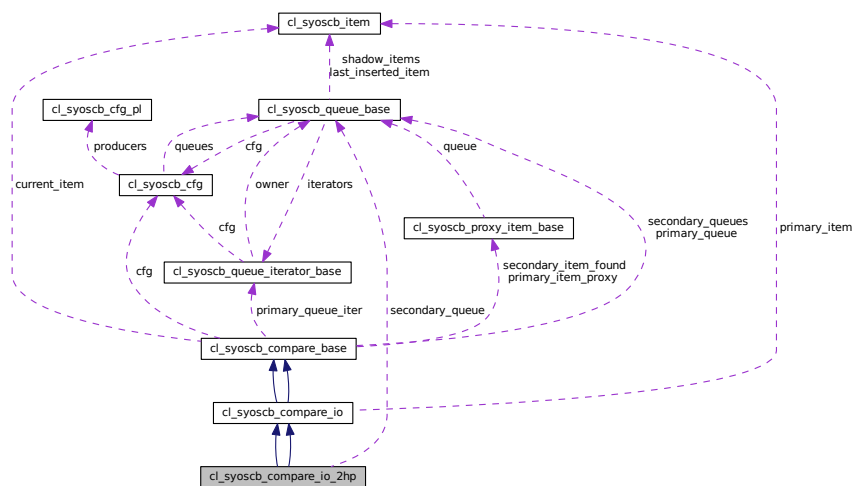
13.79 cl_syoscb_compare_io_2hp Class Reference

Implementation of the 2-queue, high speed in-order comparison algorithm.

Inheritance diagram for cl_syoscb_compare_io_2hp:



Collaboration diagram for cl_syoscb_compare_io_2hp:



Protected Member Functions

- virtual void `compare_do` ()

Compare Strategy API: Mandatory overwriting of the base class' `do_compare` method.

- virtual void `primary_loop_do` ()

Compare Strategy API: Selects the primary queue's first element, comparing it to the secondary queue's first element.

- virtual void [compare_do](#) ()

Compare Strategy API: Starts the actual comparison operation

1. Perform initialization on the primary queue, if necessary
2. Start the primary queue loop

- virtual void [primary_loop_do](#) ()

Compare Strategy API: Implementation of the in-order comparison algorithm.

Protected Attributes

- [cl_syoscb_queue_base secondary_queue](#)

Handle to the secondary queue.

Additional Inherited Members

13.79.1 Detailed Description

Implementation of the 2-queue, high speed in-order comparison algorithm.

Definition at line 2 of file [cl_syoscb_compare_io_2hp.svh](#).

13.79.2 Member Function Documentation

13.79.2.1 [compare_do\(\)](#)

```
void cl_syoscb_compare_io_2hp::compare_do ( ) [protected], [virtual]
```

Compare Strategy API: Mandatory overwriting of the base class' `do_compare` method.

Here the actual in-order 2-queue compare is implemented.

The algorithm is a specialization of the normal in-order compare which handles N queues. Here, only 2 queues are allowed and the compare simply just checks if the first item in the primary queue matches the first item in the secondary queue. If not then a UVM error is issued.

Reimplemented from [cl_syoscb_compare_base](#).

Definition at line 31 of file [cl_syoscb_compare_io_2hp.svh](#).

References [primary_loop_do\(\)](#), [cl_syoscb_compare_base::primary_loop_init\(\)](#), and [cl_syoscb_compare_base::secondary_queues](#).

13.79.2.2 primary_loop_do() [1/2]

```
void cl_syoscb_compare_io_2hp::primary_loop_do ( ) [protected], [virtual]
```

Compare Strategy API: Selects the primary queue's first element, comparing it to the secondary queue's first element.

Does this without using [secondary_loop_do](#), as no looping is required.

Reimplemented from [cl_syoscb_compare_io](#).

Definition at line 44 of file cl_syoscb_compare_io_2hp.svh.

References [cl_syoscb_compare_base::cfg](#), [cl_syoscb_queue_base::create_iterator\(\)](#), [cl_syoscb_queue_iterator_base::first\(\)](#), [cl_syoscb_compare_base::generate_miscomp_table\(\)](#), [cl_syoscb_cfg::get_comparer\(\)](#), [cl_syoscb_cfg::get_default_comparer\(\)](#), [cl_syoscb_queue_base::get_item\(\)](#), [cl_syoscb_queue_base::get_iterator\(\)](#), [cl_syoscb_item::get_producer\(\)](#), [cl_syoscb_cfg::get_scb_name\(\)](#), [cl_syoscb_queue_iterator_base::next\(\)](#), [cl_syoscb_compare_io::primary_item](#), [cl_syoscb_compare_base::primary_item_proxy](#), [cl_syoscb_compare_base::primary_queue](#), [cl_syoscb_compare_base::primary_queue_iter](#), [cl_syoscb_compare_base::primary_queue_name](#), [cl_syoscb_compare_base::secondary_item_found](#), [cl_syoscb_compare_base::secondary_queue_names](#), [cl_syoscb_compare_base::secondary_queues](#), and [cl_syoscb_string_library::sprint_item\(\)](#).

Referenced by [compare_do\(\)](#).

13.79.2.3 primary_loop_do() [2/2]

```
virtual void cl_syoscb_compare_io_2hp::primary_loop_do ( ) [protected], [virtual]
```

Compare Strategy API: Implementation of the in-order comparison algorithm.

In the primary loop, the algorithm extracts the oldest inserted element from the primary queue, and then starts looping over all secondary queues to find a matching item in [secondary_loop_do](#). If matching items are found, these are removed from all of the queues. If no matching items are found, a miscompare is generated and a UVM_ERROR is issued.

Reimplemented from [cl_syoscb_compare_io](#).

The documentation for this class was generated from the following files:

- cl_syoscb_compare_io_2hp.svh
- pk_syoscb.sv

- virtual void [secondary_loop_do](#) ()

Compare Strategy API: Loop over all secondary queues to find a match for the primary item.

- virtual string [get_count_producer](#) ()

Compare Strategy API: Returns the name of the producer that the compare method should evaluate in order to verify if it makes sense to start a comparison.

Protected Attributes

- [cl_syoscb_item](#) primary_item

Scoreboard wrapper item from the primary queue.

- [cl_syoscb_item](#) secondary_item

Scoreboard wrapper item from the secondary queue currently being inspected.

Additional Inherited Members

13.80.1 Detailed Description

Class which implements the in order by producer compare algorithm.

Definition at line 2 of file cl_syoscb_compare_iop.svh.

13.80.2 Member Function Documentation

13.80.2.1 `compare_init()` [1/2]

```
void cl_syoscb_compare_iop::compare_init ( ) [protected], [virtual]
```

Compare Strategy API: Verifies if the conditions for starting a compare are met.

For IOP comparison, we only check whether all queues have at least one item in them, but do not check if primary queue's oldest item's producer exists in all queues. Checking if all queues have an item from the same producer is moved to `primary_loop_do`

Reimplemented from [cl_syoscb_compare_base](#).

Definition at line 39 of file cl_syoscb_compare_iop.svh.

References [cl_syoscb_compare_base::check_queues\(\)](#).

13.80.2.2 `compare_init()` [2/2]

```
virtual void cl_syoscb_compare_iop::compare_init ( ) [protected], [virtual]
```

Compare Strategy API: Verifies if the conditions for starting a compare are met:

1. Verify that all queues currently contain at least one element
2. Verify that all queues have at least one element from the same producer as the producer returned by [get_count_producer\(\)](#) (the primary item being searched for) If the conditions are met then go variable is triggered, and the compare process can start.

Reimplemented from [cl_syoscb_compare_base](#).

13.80.2.3 `get_count_producer()` [1/2]

```
string cl_syoscb_compare_iop::get_count_producer ( ) [protected], [virtual]
```

Compare Strategy API: For IOP comparisons, this function returns the producer of the first element (the oldest) inside the primary queue, and not the most recently inserted item.

Reimplemented from [cl_syoscb_compare_base](#).

Definition at line 165 of file `cl_syoscb_compare_iop.svh`.

References `cl_syoscb_queue_base::create_iterator()`, `cl_syoscb_queue_iterator_base::first()`, `cl_syoscb_queue_base::get_item()`, `cl_syoscb_item::get_producer()`, `cl_syoscb_queue_iterator_base::next()`, `cl_syoscb_compare_base::primary_queue`, and `cl_syoscb_compare_base::primary_queue_iter`.

13.80.2.4 `get_count_producer()` [2/2]

```
virtual string cl_syoscb_compare_iop::get_count_producer ( ) [protected], [virtual]
```

Compare Strategy API: Returns the name of the producer that the compare method should evaluate in order to verify if it makes sense to start a comparison.

Note

This needs to be overridden by the derived compare methods in order to change the behaviour accordingly to the requested needs. By default, the function returns the producer of [current_item](#).

Returns

The name producer which should be evaluated

Reimplemented from [cl_syoscb_compare_base](#).

13.80.2.5 primary_loop_do() [1/2]

```
void cl_syoscb_compare_iop::primary_loop_do ( ) [protected], [virtual]
```

Compare Strategy API: Implementation of the in-order by producer comparison.

The algorithm gets the primary queue, extracting the oldest element. It then checks if all other queues also contain an element from this element's producer. If true, it attempts to find a match for the primary item in all secondary queues. If false, extracts the second-oldest element from primary, checking if this item's producer has at least one item in all other queues. Continues performing this loop over items in primary queue until one of three things happen:

1. A match is found for the item from the primary queue, the item and matches are removed from their queues.
2. An item from a secondary queue has the same producer but does not match primary item. This generates a mismatch and raises a UVM_ERROR.
3. No matches are found. Will search over at most [cl_syoscb_cfg::max_search_window](#) elements in the primary and secondary queues. Does not raise a UVM_ERROR Note that this may leave the queues non_empty at the end of simulation without triggering any errors. These orphaned items in queues are caught in the check_phase.

Reimplemented from [cl_syoscb_compare_base](#).

Definition at line 59 of file cl_syoscb_compare_iop.svh.

References [cl_syoscb_compare_base::cfg](#), [cl_syoscb_compare_base::count_producers\(\)](#), [cl_syoscb_queue_iterator_base::first\(\)](#), [cl_syoscb_queue_base::get_item\(\)](#), [cl_syoscb_cfg::get_max_search_window\(\)](#), [cl_syoscb_item::get_producer\(\)](#), [cl_syoscb_cfg::get_scb_name\(\)](#), [cl_syoscb_compare_base::go](#), [cl_syoscb_queue_iterator_base::has_next\(\)](#), [cl_syoscb_queue_iterator_base::next\(\)](#), [cl_syoscb_queue_iterator_base::next_index\(\)](#), [primary_item](#), [cl_syoscb_compare_base::primary_item_proxy](#), [cl_syoscb_compare_base::primary_queue](#), [cl_syoscb_compare_base::primary_queue_iter](#), [cl_syoscb_compare_base::primary_queue_name](#), [cl_syoscb_compare_base::secondary_item_found](#), [secondary_loop_do\(\)](#), and [cl_syoscb_string_library::sprint_item\(\)](#).

13.80.2.6 primary_loop_do() [2/2]

```
virtual void cl_syoscb_compare_iop::primary_loop_do ( ) [protected], [virtual]
```

Compare Strategy API: Loop over the primary queue, selecting primary items to compare against items in the secondary queues.

Note

Abstract method. This method must be implemented in a subclass.

Reimplemented from [cl_syoscb_compare_base](#).

13.80.2.7 `secondary_loop_do()` [1/2]

```
void cl_syoscb_compare_iop::secondary_loop_do ( ) [protected], [virtual]
```

Compare Strategy API: Loop through all secondary queues, attempting to find an item which matches the primary item.

Reimplemented from [cl_syoscb_compare_base](#).

Definition at line 99 of file `cl_syoscb_compare_iop.svh`.

References `cl_syoscb_compare_base::cfg`, `cl_syoscb_queue_base::create_iterator()`, `cl_syoscb_queue_iterator_base::first()`, `cl_syoscb_compare_base::generate_miscomp_table()`, `cl_syoscb_cfg::get_comparer()`, `cl_syoscb_cfg::get_default_comparer()`, `cl_syoscb_queue_base::get_item()`, `cl_syoscb_queue_base::get_iterator()`, `cl_syoscb_cfg::get_max_search_window()`, `cl_syoscb_item::get_producer()`, `cl_syoscb_cfg::get_scb_name()`, `cl_syoscb_queue_iterator_base::has_next()`, `cl_syoscb_queue_iterator_base::next()`, `cl_syoscb_queue_iterator_base::next_index()`, `cl_syoscb_queue_iterator_base::previous_index()`, `primary_item`, `cl_syoscb_compare_base::primary_queue_name`, `secondary_item`, `cl_syoscb_compare_base::secondary_item_found`, `cl_syoscb_compare_base::secondary_queue_names`, `cl_syoscb_compare_base::secondary_queues`, and `cl_syoscb_string_library::sprint_item()`.

Referenced by `primary_loop_do()`.

13.80.2.8 `secondary_loop_do()` [2/2]

```
virtual void cl_syoscb_compare_iop::secondary_loop_do ( ) [protected], [virtual]
```

Compare Strategy API: Loop over all secondary queues to find a match for the primary item.

Note

Abstract method. This method must be implemented in a subclass.

Reimplemented from [cl_syoscb_compare_base](#).

The documentation for this class was generated from the following files:

- `cl_syoscb_compare_iop.svh`
- `pk_syoscb.sv`

Additional Inherited Members

13.81.1 Detailed Description

Class which implements the out of order compare algorithm.

Definition at line 2 of file `cl_syoscb_compare_ooo.svh`.

13.81.2 Member Function Documentation

13.81.2.1 `get_count_producer()` [1/2]

```
string cl_syoscb_compare_ooo::get_count_producer ( ) [protected], [virtual]
```

Compare Strategy API: For OOO comparisons, the overridden function returns the producer of the first element (the oldest) inside the primary queue.

Reimplemented from [cl_syoscb_compare_base](#).

Definition at line 90 of file `cl_syoscb_compare_ooo.svh`.

References `cl_syoscb_queue_base::create_iterator()`, `cl_syoscb_queue_iterator_base::first()`, `cl_syoscb_queue_base::get_item()`, `cl_syoscb_item::get_producer()`, `cl_syoscb_queue_iterator_base::next()`, `cl_syoscb_compare_base::primary_queue`, and `cl_syoscb_compare_base::primary_queue_iter`.

13.81.2.2 `get_count_producer()` [2/2]

```
virtual string cl_syoscb_compare_ooo::get_count_producer ( ) [protected], [virtual]
```

Compare Strategy API: Returns the name of the producer that the compare method should evaluate in order to verify if it makes sense to start a comparison.

Note

This needs to be overridden by the derived compare methods in order to change the behaviour accordingly to the requested needs. By default, the function returns the producer of [current_item](#).

Returns

The name producer which should be evaluated

Reimplemented from [cl_syoscb_compare_base](#).

13.81.2.3 primary_loop_do() [1/2]

```
void cl_syoscb_compare_ooo::primary_loop_do ( ) [protected], [virtual]
```

Compare Strategy API: Implementation of the out-of-order comparison is here.

The algorithm iterates over the primary queue, starting from the oldest inserted item. For each item in the primary queue, it then loops over all secondary queues, attempting to find a matching item in the secondary queue. If a match for an item in the primary queue is found in all secondary queues, all of those items are removed from their respective queues. If a match is not found in all queues, nothing is deleted. Note that this means that if some items are not matched, the queues will be non-empty at the end of simulation. This is caught in the [cl_syoscb::check_phase](#).

The number of items that are inspected in each queue is controlled by the value of [cl_syoscb_cfg::max_search_window](#) for that specific queue.

Reimplemented from [cl_syoscb_compare_base](#).

Definition at line 39 of file [cl_syoscb_compare_ooo.svh](#).

References [cl_syoscb_compare_base::cfg](#), [cl_syoscb_queue_iterator_base::first\(\)](#), [cl_syoscb_cfg::get_max_search_window\(\)](#), [cl_syoscb_queue_iterator_base::has_next\(\)](#), [cl_syoscb_queue_iterator_base::next\(\)](#), [cl_syoscb_queue_iterator_base::next_index\(\)](#), [cl_syoscb_compare_base::primary_item_proxy](#), [cl_syoscb_compare_base::primary_queue_iter](#), [cl_syoscb_compare_base::primary_queue_name](#), [cl_syoscb_compare_base::secondary_item_found](#), and [secondary_loop_do\(\)](#).

13.81.2.4 primary_loop_do() [2/2]

```
virtual void cl_syoscb_compare_ooo::primary_loop_do ( ) [protected], [virtual]
```

Compare Strategy API: Loop over the primary queue, selecting primary items to compare against items in the secondary queues.

Note

Abstract method. This method must be implemented in a subclass.

Reimplemented from [cl_syoscb_compare_base](#).

13.81.2.5 secondary_loop_do() [1/2]

```
void cl_syoscb_compare_ooo::secondary_loop_do ( ) [protected], [virtual]
```

Compare Strategy API: Loop through all secondary queues, attempting to find an item which matches the item from the primary queue (as specified by [primary_item_proxy](#)).

Searches at most [cl_syoscb_cfg::max_search_window](#) in each secondary queue if std. queues are used. If MD5 queues are used, the max search window is not applied.

Reimplemented from [cl_syoscb_compare_base](#).

Definition at line 63 of file [cl_syoscb_compare_ooo.svh](#).

References [cl_syoscb_queue_base::get_locator\(\)](#), [cl_syoscb_compare_base::primary_item_proxy](#), [cl_syoscb_queue_locator_base::search\(\)](#), [cl_syoscb_compare_base::secondary_item_found](#), [cl_syoscb_compare_base::secondary_queue_names](#), and [cl_syoscb_compare_base::secondary_queues](#).

Referenced by [primary_loop_do\(\)](#).

13.81.2.6 secondary_loop_do() [2/2]

```
virtual void cl_syoscb_compare_ooo::secondary_loop_do ( ) [protected], [virtual]
```

Compare Strategy API: Loop over all secondary queues to find a match for the primary item.

Note

Abstract method. This method must be implemented in a subclass.

Reimplemented from [cl_syoscb_compare_base](#).

The documentation for this class was generated from the following files:

- cl_syoscb_compare_ooo.svh
- pk_syoscb.sv

13.82 cl_syoscb_comparer_config Class Reference

Utility class used to perform manipulations of uvm_comparer objects.

Inherits uvm_object, and uvm_object.

Static Public Member Functions

- static void [set_verbosity](#) (uvm_comparer comparer, int unsigned cv=UVM_DEBUG)
Sets the verbosity level of a given comparer.
- static int unsigned [get_verbosity](#) (uvm_comparer comparer)
Gets the verbosity level for a given comparer.
- static void [copy_comparer](#) (uvm_comparer from, uvm_comparer to)
Copies all config information from one comparer into another.
- static string [get_miscompares_from_comparer](#) (uvm_comparer comparer)
Returns a string containing all miscompares from the given comparer.
- static void [do_help_pack](#) (uvm_comparer comparer, uvm_packer packer)
Packs all configuration data for the given uvm_comparer using the given uvm_packer.
- static uvm_comparer [do_help_unpack](#) (uvm_packer packer)
Unpacks comparer configuration data and returns a comparer with that configuration.
- static void [set_show_max](#) (uvm_comparer comparer, int unsigned sm)
Sets the value of the show_max knob in the given comparer.
- static int unsigned [get_show_max](#) (uvm_comparer comparer)
Gets the value of the show_max knob in the given comparer.

13.82.1 Detailed Description

Utility class used to perform manipulations of uvm_comparer objects.

Contains a number of functions that simplify uvm_comparer related code, as the comparer API differs based on the UVM version used. These functions encapsulate those differences, providing a unified API regardless of UVM version.

Definition at line 5 of file cl_syoscb_comparer_config.svh.

13.82.2 Member Function Documentation

13.82.2.1 copy_comparer()

```
void cl_syoscb_comparer_config::copy_comparer (
    uvm_comparer from,
    uvm_comparer to ) [static]
```

Copies all config information from one comparer into another.

Parameters

<i>from</i>	Comparer containing the data to be copied
<i>to</i>	Comparer to inherit configuration data in <i>from</i>

Definition at line 70 of file cl_syoscb_comparer_config.svh.

13.82.2.2 do_help_pack()

```
void cl_syoscb_comparer_config::do_help_pack (
    uvm_comparer comparer,
    uvm_packer packer ) [static]
```

Packs all configuration data for the given uvm_comparer using the given uvm_packer.

Since uvm_comparer does not natively support pack/unpack operations, these helper methods can be used to pack/unpack a comparer

Parameters

<i>comparer</i>	The uvm_comparer for which all configuration values should be packed
<i>packer</i>	The uvm_packer to use when packing the item

Definition at line 98 of file cl_syoscb_comparer_config.svh.

13.82.2.3 do_help_unpack()

```
uvm_comparer cl_syoscb_comparer_config::do_help_unpack (
    uvm_packer packer ) [static]
```

Unpacks comparer configuration data and returns a comparer with that configuration.

Since uvm_comparer does not natively support pack/unpack operations, these helper methods can be used to pack/unpack a comparer

Parameters

<i>packer</i>	The uvm_packer that was previously used to pack a uvm_comparer
---------------	--

Returns

A uvm_comparer with the packed configuration

Definition at line 150 of file cl_syoscb_comparer_config.svh.

13.82.2.4 get_miscompares_from_comparer()

```
string cl_syoscb_comparer_config::get_miscompares_from_comparer (  
    uvm_comparer comparer ) [static]
```

Returns a string containing all miscompares from the given comparer.

Parameters

<i>comparer</i>	The comparer from which to get all miscompares
-----------------	--

Returns

A string containing information about the miscompares in the comparer

Definition at line 230 of file cl_syoscb_comparer_config.svh.

13.82.2.5 get_show_max()

```
int unsigned cl_syoscb_comparer_config::get_show_max (  
    uvm_comparer comparer ) [static]
```

Gets the value of the show_max knob in the given comparer.

Parameters

<i>comparer</i>	The comparer to get the show_max knob for
-----------------	---

Returns

The value of show_max in the given comparer

Definition at line 254 of file cl_syoscb_comparer_config.svh.

13.82.2.6 get_verbosity()

```
int unsigned cl_syoscb_comparer_config::get_verbosity (
    uvm_comparer comparer ) [static]
```

Gets the verbosity level for a given comparer.

Parameters

<i>comparer</i>	The comparer object for which to get the verbosity level
-----------------	--

Returns

That comparers verbosity level

Definition at line 58 of file cl_syoscb_comparer_config.svh.

13.82.2.7 set_show_max()

```
void cl_syoscb_comparer_config::set_show_max (
    uvm_comparer comparer,
    int unsigned sm ) [static]
```

Sets the value of the `show_max` knob in the given comparer.

Parameters

<i>comparer</i>	The comparer to set the <code>show_max</code> knob for
<i>sm</i>	The new value of the <code>show_max</code> knob

Definition at line 242 of file cl_syoscb_comparer_config.svh.

13.82.2.8 set_verbosity()

```
void cl_syoscb_comparer_config::set_verbosity (
    uvm_comparer comparer,
    int unsigned cv = UVM_DEBUG ) [static]
```

Sets the verbosity level of a given comparer.

Parameters

<i>comparer</i>	The comparer object on which to set a new verbosity level
<i>cv</i>	The new comparer verbosity level

Definition at line 46 of file cl_syoscb_comparer_config.svh.

Referenced by cl_syoscb_cfg::get_default_comparer(), and cl_syoscb_cfg::init().

The documentation for this class was generated from the following files:

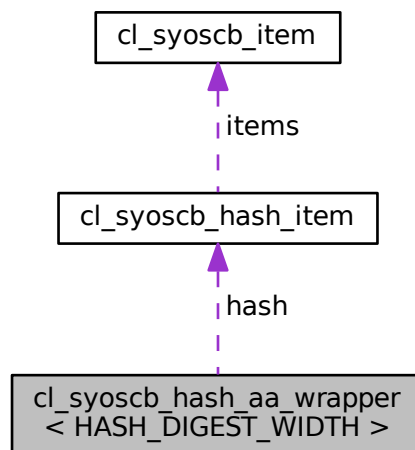
- cl_syoscb_comparer_config.svh
- pk_syoscb.sv

13.83 cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH > Class Template Reference

A wrapper around an associative array, used for storing hash queues.

Inherits uvm_object, and uvm_object.

Collaboration diagram for cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >:



Public Types

- typedef `cl_syoscb_hash_base< HASH_DIGEST_WIDTH >::tp_hash_digest` `tp_digest`
Typedef for hash algorithm digests.
- typedef `cl_syoscb_hash_base< HASH_DIGEST_WIDTH >::tp_hash_digest` `tp_digest`
Typedef for hash algorithm digests.

Public Member Functions

- `int size ()`
Returns the size (number of entries) in the wrapped assoc array.
- `int get_size (tp_digest digest)`
Return the size of a hash item in the wrapped assoc array.
- `void insert (tp_digest digest, cl_syoscb_item item)`
Inserts an item into the wrapped assoc array.
- `cl_syoscb_item get_item (tp_digest digest, int unsigned idx=0)`
Gets the scoreboard item at an index with a given hash value.
- `cl_syoscb_hash_item get_hash_item (tp_digest digest)`
Gets a hash item in the wrapped assoc array.
- `void delete (tp_digest digest, int unsigned idx=0)`
Deletes a sequence item with a given hash value.
- `void delete_all ()`
Deletes all items in the assoc array.
- `bit exists (tp_digest digest)`
Checks if an entry exists with the given hash value.
- `bit first (ref tp_digest digest)`
Retrieves the hash of the first item in the wrapped AA The first item is not necessarily the item first inserted, but the item that comes first "alphabetically".
- `bit last (ref tp_digest digest)`
Retrieves the hash of the last item in the wrapped AA The last item is not necessarily the item last inserted, but the item that comes last "alphabetically".
- `bit next (ref tp_digest digest)`
Retrieves the hash of the next item in the wrapped AA The first item is not necessarily the next item in insertion order, but the item that comes next "alphabetically".
- `bit prev (ref tp_digest digest)`
Retrieves the hash of the previous item in the wrapped AA The previous item is not necessarily the previous item in insertion order, but the previous item "alphabetically".

Public Attributes

- `cl_syoscb_hash_item hash [tp_digest]`
Queue implemented as assoc array.

13.83.1 Detailed Description

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
class cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >
```

A wrapper around an associative array, used for storing hash queues.

Supports all of the same functions that an ordinary AA supports

Definition at line 3 of file cl_syoscb_hash_aa_wrapper.svh.

13.83.2 Member Function Documentation

13.83.2.1 delete()

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
void cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::delete (
    tp_digest digest,
    int unsigned idx = 0 )
```

Deletes a sequence item with a given hash value.

Parameters

<i>digest</i>	The hash digest of the item to delete
<i>idx</i>	The index in the hash item of the sequence item to delete. Defaults to 0.

Must remove to avoid iterating over empty hash items

Definition at line 103 of file cl_syoscb_hash_aa_wrapper.svh.

References `cl_syoscb_hash_item::delete_item()`, `cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::get_size()`, and `cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::hash`.

13.83.2.2 exists()

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
bit cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::exists (
    tp_digest digest )
```

Checks if an entry exists with the given hash value.

Parameters

<i>digest</i>	The hash value to check for
---------------	-----------------------------

Returns

1 if an item with that hash exists, 0 otherwise

Definition at line 119 of file cl_syoscb_hash_aa_wrapper.svh.

References `cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::hash`.

Referenced by `cl_syoscb_queue_locator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::search()`.

13.83.2.3 first()

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
bit cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::first (
    ref tp_digest digest )
```

Retrieves the hash of the first item in the wrapped AA The first item is not necessarily the item first inserted, but the item that comes first "alphabetically".

Parameters

<i>digest</i>	A reference to a digest, where the digest of the first entry is returned
---------------	--

Returns

1 if the digest is valid, 0 otherwise

Definition at line 128 of file `cl_syoscb_hash_aa_wrapper.svh`.

References `cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::hash`.

Referenced by `cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::first()`, `cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::get_item_proxy()`, and `cl_syoscb_queue_locator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::validate_no_match()`.

13.83.2.4 `get_hash_item()`

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
cl_syoscb_hash_item cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::get_hash_item (
    tp_digest digest )
```

Gets a hash item in the wrapped assoc array.

Parameters

<i>digest</i>	The hash digest of the item to get
---------------	------------------------------------

Returns

The hash item at that digest, or null if none exists

Definition at line 82 of file `cl_syoscb_hash_aa_wrapper.svh`.

References `cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::hash`.

13.83.2.5 `get_item()`

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
cl_syoscb_item cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::get_item (
    tp_digest digest,
    int unsigned idx = 0 )
```

Gets the scoreboard item at an index with a given hash value.

Parameters

<i>digest</i>	The hash value of the item to get
<i>idx</i>	The index in the hash item. Defaults to 0.

Returns

That item, or null if no item with that hash exists or idx was too large

Definition at line 93 of file `cl_syoscb_hash_aa_wrapper.svh`.

References `cl_syoscb_hash_item::get_item()`, and `cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::hash`.

Referenced by `cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::next()`, `cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::previous()`, `cl_syoscb_queue_locator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::search()`, and `cl_syoscb_queue_locator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::validate_no_match()`.

13.83.2.6 get_size()

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
int cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::get_size (
    tp_digest digest )
```

Return the size of a hash item in the wrapped assoc array.

Parameters

<i>digest</i>	The digest of the hash item to retrieve
---------------	---

Returns

The size of that hash item, 0 if none exists

Definition at line 60 of file `cl_syoscb_hash_aa_wrapper.svh`.

References `cl_syoscb_hash_item::get_size()`, and `cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::hash`.

Referenced by `cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::delete()`, `cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::last()`, `cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::next()`, `cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::previous()`, `cl_syoscb_queue_locator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::search()`, and `cl_syoscb_queue_locator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::validate_no_match()`.

13.83.2.7 insert()

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
void cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::insert (
    tp_digest digest,
    cl_syoscb_item item )
```

Inserts an item into the wrapped assoc array.

Parameters

<i>digest</i>	The hash digest of the item to insert
<i>item</i>	The item to insert

Definition at line 70 of file cl_syoscb_hash_aa_wrapper.svh.

References `cl_syoscb_hash_item::add_item()`, and `cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::hash`.

13.83.2.8 last()

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
bit cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::last (
    ref tp_digest digest )
```

Retrieves the hash of the last item in the wrapped AA The last item is not necessarily the item last inserted, but the item that comes last "alphabetically".

Parameters

<i>digest</i>	A reference to a digest, where the digest of the last entry is returned
---------------	---

Returns

1 if the digest is valid, 0 otherwise

Definition at line 137 of file cl_syoscb_hash_aa_wrapper.svh.

References `cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::hash`.

Referenced by `cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::last()`.

13.83.2.9 next()

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
bit cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::next (
    ref tp_digest digest )
```

Retrieves the hash of the next item in the wrapped AA The first item is not necessarily the next item in insertion order, but the item that comes next "alphabetically".

Parameters

<i>digest</i>	A reference to the digest of the current value. The digest of the next entry is returned here
---------------	---

Returns

1 if the digest is valid, 0 otherwise

Definition at line 146 of file `cl_syoscb_hash_aa_wrapper.svh`.

References `cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::hash`.

Referenced by `cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::next()`, and `cl_syoscb_queue_locator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::validate_no_match()`.

13.83.2.10 prev()

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
bit cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::prev (
    ref tp_digest digest )
```

Retrieves the hash of the previous item in the wrapped AA The previous item is not necessarily the previous item in insertion order, but the previous item "alphabetically".

Parameters

<i>digest</i>	A reference to the digest of the current value. The digest of the previous entry is returned here
---------------	---

Returns

1 if the digest is valid, 0 otherwise

Definition at line 155 of file `cl_syoscb_hash_aa_wrapper.svh`.

References `cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::hash`.

Referenced by `cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::previous()`.

13.83.2.11 size()

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
int cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::size ( )
```

Returns the size (number of entries) in the wrapped assoc array.

Note

This does not necessarily match the size of the contained queue, as a hash item may have multiple entries

Definition at line 53 of file cl_syoscb_hash_aa_wrapper.svh.

References cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::hash.

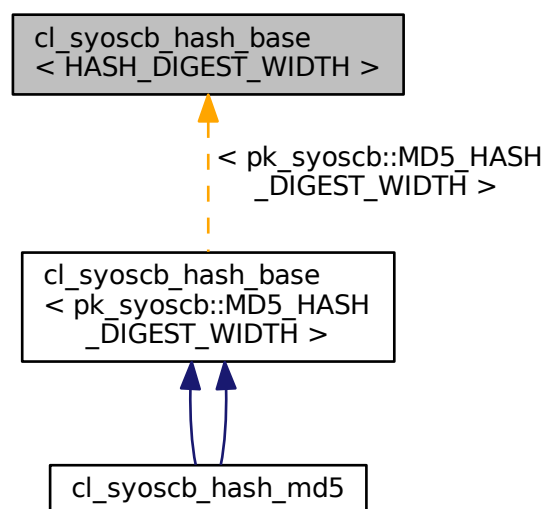
The documentation for this class was generated from the following files:

- cl_syoscb_hash_aa_wrapper.svh
- pk_syoscb.sv

13.84 cl_syoscb_hash_base< HASH_DIGEST_WIDTH > Class Template Reference

Class which defines the base concept of a hash algorithm.

Inheritance diagram for cl_syoscb_hash_base< HASH_DIGEST_WIDTH >:



13.84.1 Detailed Description

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
class cl_syoscb_hash_base< HASH_DIGEST_WIDTH >
```

Class which defines the base concept of a hash algorithm.

All hash functions must extend this class and implement the hash API.

Parameters

<code>HASH_DIGEST_WIDTH</code>	The number of bits in the hash digest for that hashing algorithm
--------------------------------	--

Definition at line 4 of file `cl_syoscb_hash_base.svh`.

13.84.2 Member Function Documentation

13.84.2.1 `do_hash()`

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
cl_syoscb_hash_base::tp_hash_digest cl_syoscb_hash_base< HASH_DIGEST_WIDTH >::do_hash (
    bit ser[] ) [protected], [virtual]
```

Hash API: Returns the hash value of the given bitstream.

The bitstream must comply with the chosen hash algorithm's requirements.

Parameters

<code>ser</code>	The bitstream to generate the hash for
------------------	--

Returns

The hash of the input bitstream

Reimplemented in [cl_syoscb_hash_md5](#), and [cl_syoscb_hash_md5](#).

Definition at line 54 of file `cl_syoscb_hash_base.svh`.

Referenced by `cl_syoscb_hash_base< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::hash()`, and `cl_syoscb_hash_base< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::hash_str()`.

13.84.2.2 `hash()`

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
cl_syoscb_hash_base::tp_hash_digest cl_syoscb_hash_base< HASH_DIGEST_WIDTH >::hash (
    cl_syoscb_item item ) [virtual]
```

Hash API: Hashes a [cl_syoscb_item](#), returning its hash value

Parameters

<i>item</i>	The item to hash
-------------	------------------

Returns

The hash value of that string

Definition at line 83 of file `cl_syoscb_hash_base.svh`.

13.84.2.3 hash_str()

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
cl_syoscb_hash_base::tp_hash_digest cl_syoscb_hash_base< HASH_DIGEST_WIDTH >::hash_str (
    string str ) [virtual]
```

Hash API: Hashes a string, returning its hash value

Parameters

<i>str</i>	The string to hash
------------	--------------------

Returns

The hash value of that string

Definition at line 62 of file `cl_syoscb_hash_base.svh`.

13.84.3 Member Data Documentation**13.84.3.1 packer**

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
cl_syoscb_hash_packer cl_syoscb_hash_base< HASH_DIGEST_WIDTH >::packer [protected]
```

Handle to a packer suited for this hash algorithm.

The packer should be set in the implementing class' constructor

Definition at line 16 of file `cl_syoscb_hash_base.svh`.

Referenced by `cl_syoscb_hash_base< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::hash()`, and `cl_syoscb_hash_base< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::hash_str()`.

The documentation for this class was generated from the following files:

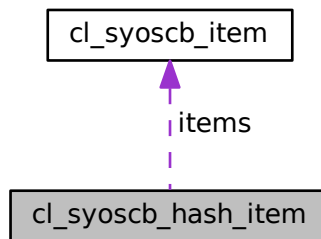
- `cl_syoscb_hash_base.svh`
- `pk_syoscb.sv`

13.85 cl_syoscb_hash_item Class Reference

A utility class used to wrap [cl_syoscb_item](#) objects when when using hash queues.

Inherits [uvm_object](#), and [uvm_object](#).

Collaboration diagram for [cl_syoscb_hash_item](#):



Public Member Functions

- virtual [cl_syoscb_item](#) [get_item](#) (int unsigned idx=0)
***Item API:** Returns an item from this hash item's queue. If called without parameters, returns the first item from the queue. If idx is not a valid index in the queue, raises a UVM_WARNING and returns null.*
- virtual void [add_item](#) ([cl_syoscb_item](#) item)
***Item API:** Adds an item to this hash item.*
- virtual int unsigned [get_size](#) ()
***Item API:** Returns the number of items stored in this hash item.*
- virtual void [delete_item](#) (int unsigned idx=0)
***Item API:** Deletes an item from this hash item.*

Private Attributes

- [cl_syoscb_item](#) [items](#) [\$]
Queue of [cl_syoscb_item](#) with the same hash.

13.85.1 Detailed Description

A utility class used to wrap [cl_syoscb_item](#) objects when when using hash queues.

In case of a hash collision, this class contains a queue of all items with the same hash.

Definition at line 3 of file `cl_syoscb_hash_item.svh`.

13.85.2 Member Function Documentation

13.85.2.1 add_item()

```
void cl_syoscb_hash_item::add_item (  
    cl_syoscb_item item ) [virtual]
```

Item API: Adds an item to this hash item

Parameters

<i>item</i>	The item to add
-------------	-----------------

Definition at line 52 of file cl_syoscb_hash_item.svh.

References items.

Referenced by cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::insert().

13.85.2.2 delete_item()

```
void cl_syoscb_hash_item::delete_item (
    int unsigned idx = 0 ) [virtual]
```

Item API: Deletes an item from this hash item

Parameters

<i>idx</i>	The index of the item to delete. If index is out range, generates a UVM_ERROR
------------	---

Definition at line 65 of file cl_syoscb_hash_item.svh.

References items.

Referenced by cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::delete().

13.85.2.3 get_item()

```
cl_syoscb_item cl_syoscb_hash_item::get_item (
    int unsigned idx = 0 ) [virtual]
```

Item API: Returns an item from this hash item's queue If called without parameters, returns the first item from the queue If idx is not a valid index in the queue, raises a UVM_WARNING and returns null

Parameters

<i>idx</i>	The index to access. Defaults to 0
------------	------------------------------------

Returns

The item at that index, or null if no items exist / the index is invalid

Definition at line 39 of file cl_syoscb_hash_item.svh.

References items.

Referenced by `cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH >::get_item()`.

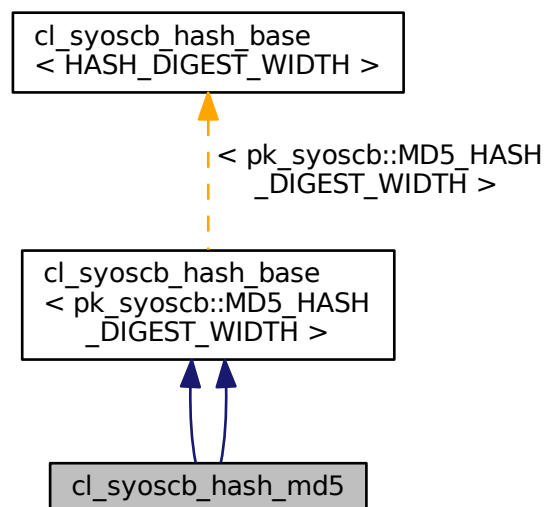
The documentation for this class was generated from the following files:

- `cl_syoscb_hash_item.svh`
- `pk_syoscb.sv`

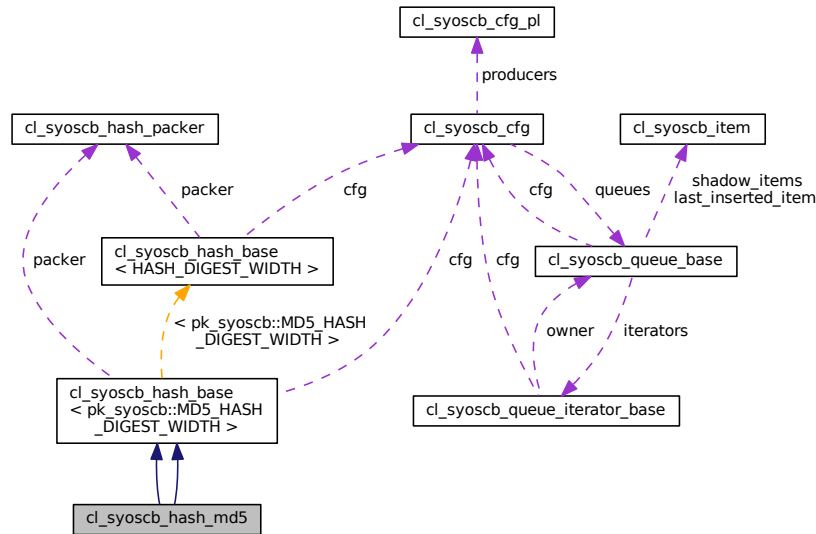
13.86 cl_syoscb_hash_md5 Class Reference

MD5 hash algorithm implementation.

Inheritance diagram for `cl_syoscb_hash_md5`:



Collaboration diagram for cl_syoscb_hash_md5:



Protected Member Functions

- virtual [tp_hash_digest do_hash](#) (bit ser [])

Hash API: See [cl_syoscb_hash_base::do_hash](#) for more details Expects a bitstream with a length which is a multiple of 512, which follows the below format, specified in RFC 1321:

- virtual [tp_hash_digest do_hash](#) (bit ser [])

Hash API: Returns the hash value of the given bitstream.

Additional Inherited Members

13.86.1 Detailed Description

MD5 hash algorithm implementation.

The class implements the hash API as defined by the hash base class.

Definition at line 3 of file `cl_syoscb_hash_md5.svh`.

13.86.2 Member Function Documentation

13.86.2.1 do_hash() [1/2]

```
cl_syoscb_hash_md5::tp_hash_digest cl_syoscb_hash_md5::do_hash (
    bit ser[] ) [protected], [virtual]
```

Hash API: See [cl_syoscb_hash_base::do_hash](#) for more details Expects a bitstream with a length which is a multiple of 512, which follows the below format, specified in RFC 1321:

Bits	Information
0 - length of the serialized item-1	Serialized item
length of the serialized item - length of the serialized item+6	Zeros
length of the serialized item+7	One
length of the serialized item+8 - (512*x-64)	Zeros
last 64 bits	length of the item modulo 2^{64}

A bitstream of this kind can be generated by packing with [cl_syoscb_md5_packer](#), which automatically appends the required metadata to the end of the bitstream

Reimplemented from [cl_syoscb_hash_base](#)< [pk_syoscb::MD5_HASH_DIGEST_WIDTH](#) >.

Definition at line 71 of file [cl_syoscb_hash_md5.svh](#).

13.86.2.2 do_hash() [2/2]

```
virtual tp\_hash\_digest cl_syoscb_hash_md5::do_hash (
    bit ser[] ) [protected], [virtual]
```

Hash API: Returns the hash value of the given bitstream.

The bitstream must comply with the chosen hash algorithm's requirements.

Parameters

<i>ser</i>	The bitstream to generate the hash for
------------	--

Returns

The hash of the input bitstream

Reimplemented from [cl_syoscb_hash_base](#)< [pk_syoscb::MD5_HASH_DIGEST_WIDTH](#) >.

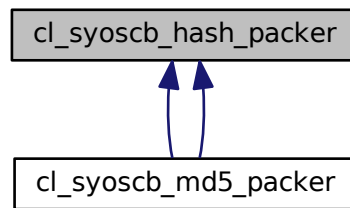
The documentation for this class was generated from the following files:

- [cl_syoscb_hash_md5.svh](#)
- [pk_syoscb.sv](#)

13.87 cl_syoscb_hash_packer Class Reference

A base class for packers which should be used with hash algorithms in the scoreboard.

Inheritance diagram for cl_syoscb_hash_packer:



Public Member Functions

- virtual void [clean](#) ()

Clean and reset the underlying packer data structure This method correctly calls packer.flush on UVM-IEEE and packer.reset on previous versions of UVM, removing the need for version-specific code in the caller.

13.87.1 Detailed Description

A base class for packers which should be used with hash algorithms in the scoreboard.

The packers should implement `get_bits` or `get_packed_bits` (depending on UVM version), returning a bitstream which conforms to the given hash algorithm's requirements

Definition at line 6 of file `cl_syoscb_hash_packer.svh`.

The documentation for this class was generated from the following files:

- `cl_syoscb_hash_packer.svh`
- `pk_syoscb.sv`

13.88 cl_syoscb_item Class Reference

The UVM scoreboard item which wraps `uvm_sequence_item` .

Inherits `uvm_object`, and `uvm_object`.

Public Member Functions

- virtual string [get_producer](#) ()
Item API: Returns the producer of the wrapped sequence item
- virtual void [set_producer](#) (string producer)
Item API: Sets the producer of the wrapped sequence item.
- virtual uvm_sequence_item [get_item](#) ()
Item API: Returns the wrapped uvm_sequence_item
- virtual void [set_item](#) (uvm_sequence_item item)
Item API: Sets the uvm_sequence_item wrapped by this wrapper item
- virtual void [set_insertion_index](#) (longint unsigned ii)
Item API: Gets the insertion index of the wrapped sequence item
- virtual longint unsigned [get_insertion_index](#) ()
Item API: Sets the insertion index of the wrapped sequence item
- virtual void [set_queue_index](#) (longint qi)
Item API: Sets the queue index of the wrapped sequence item
- virtual longint [get_queue_index](#) ()
Item API: Gets the queue index of the wrapped sequence item
- virtual string [convert2string](#) ()
Converts a [cl_syoscb_item](#) to a compact string representation.

Private Attributes

- string [producer](#)
Name of the producer that generated this seq. item.
- uvm_sequence_item [item](#)
Handle to the wrapped uvm_sequence_item.
- longint unsigned [insertion_index](#)
Insertion index N means that this is the N'th item inserted in that queue.
- longint [queue_index](#)
This item's position in the queue.

13.88.1 Detailed Description

The UVM scoreboard item which wraps uvm_sequence_item .

This ensures that future extensions to the UVM scoreboard will always be able to use all uvm_sequence_items from already existing testbenches etc. even though more META data is added to the wrapping item.

Definition at line 4 of file cl_syoscb_item.svh.

13.88.2 Member Function Documentation

13.88.2.1 convert2string()

```
string cl_syoscb_item::convert2string ( ) [virtual]
```

Converts a [cl_syoscb_item](#) to a compact string representation.

Does this by simply returning the convert2string-implementation of the wrapped sequence item.

Note

Raises a warning if newlines are contained in the output, as this may make the output uglier

Definition at line 136 of file cl_syoscb_item.svh.

Referenced by cl_syoscb_queue_base::dump().

13.88.2.2 set_producer()

```
void cl_syoscb_item::set_producer (
    string producer ) [virtual]
```

Item API: Sets the producer of the wrapped sequence item.

The validity of the producer name must be checked by the caller before setting it in this item.

Parameters

<i>producer</i>	The name of the producer of the wrapped seq. item.
-----------------	--

Definition at line 97 of file cl_syoscb_item.svh.

References producer.

Referenced by cl_syoscb_queue_base::pre_add_item().

13.88.3 Member Data Documentation

13.88.3.1 queue_index

```
longint cl_syoscb_item::queue_index [private]
```

This item's position in the queue.

This field is only valid when the queue is dumped, as a queue index may change throughout simulation as items ahead of this item are removed from the queue.

Definition at line 20 of file cl_syoscb_item.svh.

Referenced by get_queue_index().

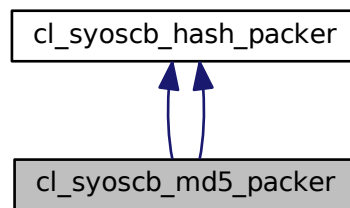
The documentation for this class was generated from the following files:

- `cl_syoscb_item.svh`
- `pk_syoscb.sv`

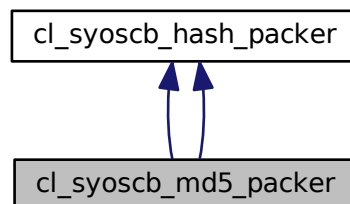
13.89 `cl_syoscb_md5_packer` Class Reference

An implementation of a `uvm_packer` which returns bitstreams that are ready for md5 packing.

Inheritance diagram for `cl_syoscb_md5_packer`:



Collaboration diagram for `cl_syoscb_md5_packer`:



Public Member Functions

- virtual void [get_bits](#) (ref bit unsigned bits[])
Gets the packer's bitstream, modifying the contents such that it conforms to RFC 1321.

13.89.1 Detailed Description

An implementation of a `uvm_packer` which returns bitstreams that are ready for md5 packing.

Generates bitstreams which follow the format below, as specified in RFC 1321

Bits	Information
0 - length of the serialized item-1	Serialized item
length of the serialized item - lentgh of the serialized item+6	Zeros
length of the serialized item+7	One
length of the serialized item+8 - (512*x-64)	Zeros
last 64 bits	length of the item modulo 2^{64}

NOTICE: The current implementation of the md5_packer only manipulates the bitstream returned when `get_bits` or `#get_packed_bits` is called, and does not modify the underlying bitstream. After calling `get_packed_bits` or `get_bits`, call `clean` to clean the underlying bitstream, allowing the packer to be reused

Definition at line 35 of file `cl_syoscb_md5_packer.svh`.

The documentation for this class was generated from the following files:

- `cl_syoscb_md5_packer.svh`
- `pk_syoscb.sv`

13.90 cl_syoscb_printer_config Class Reference

Utility class used to perform manipulations of `uvm_printer` objects.

Inherits `uvm_object`, and `uvm_object`.

Static Public Member Functions

- static void `set_file_descriptor` (`uvm_printer` printer, int fd=0)
Sets the file descriptor to be used for a given printer.
- static int `get_file_descriptor` (`uvm_printer` printer)
Gets the file descriptor used for a given printer.
- static void `copy_printer` (`uvm_printer` from, `uvm_printer` to)
Copies all config information from one printer to another printer.
- static void `set_printer_begin_elements` (`uvm_printer` printer, int elements)
Sets the number of elements to print at the head of a list whenever the printer is used to print a tx item.
- static void `set_printer_end_elements` (`uvm_printer` printer, int elements)
Sets the number of elements to print at the tail of a list whenever the printer is used to print a tx item.
- static void `do_help_pack` (`uvm_printer` printer, `uvm_packer` packer)
Packs all configuration data for the given uvm_printer using the given uvm_packer.
- static `uvm_printer` `do_help_unpack` (`uvm_packer` packer)
Unpacks printer configuration data and returns a printer with that configuration.
- static `t_printer_type` `get_printer_type` (`uvm_printer` printer)
Gets the type of printer that a given uvm_printer represents.
- static `uvm_printer` `get_printer_of_type` (`t_printer_type` ptype)
Generates a new printer of the correct type.

13.90.1 Detailed Description

Utility class used to perform manipulations of `uvm_printer` objects.

Contains a number of functions that simplify `uvm_printer` related code, as the printer API differs based on the UVM version used. These functions encapsulate those differences, providing a unified API regardless of UVM version.

Definition at line 5 of file `cl_syoscb_printer_config.svh`.

13.90.2 Member Function Documentation

13.90.2.1 `copy_printer()`

```
void cl_syoscb_printer_config::copy_printer (
    uvm_printer from,
    uvm_printer to ) [static]
```

Copies all config information from one printer to another printer.

Parameters

<i>from</i>	Printer containing the data to be copied
<i>to</i>	Printer to inherit configuration data in <i>from</i>

Definition at line 160 of file `cl_syoscb_printer_config.svh`.

13.90.2.2 `do_help_pack()`

```
void cl_syoscb_printer_config::do_help_pack (
    uvm_printer printer,
    uvm_packer packer ) [static]
```

Packs all configuration data for the given `uvm_printer` using the given `uvm_packer`.

Since `uvm_printer` does not natively support pack/unpack operations, these helper methods can be used to pack/unpack a printer

Note

The `uvm_packer` used *must* have the flag `use_metadata` set to 0b1 for this to work correctly

Parameters

<i>printer</i>	The <code>uvm_printer</code> for which all configuration values should be packed
<i>packer</i>	The <code>uvm_packer</code> to use when packing the item

Definition at line 213 of file cl_syoscb_printer_config.svh.

References `get_printer_type()`.

13.90.2.3 do_help_unpack()

```
uvm_printer cl_syoscb_printer_config::do_help_unpack (
    uvm_packer packer ) [static]
```

Unpacks printer configuration data and returns a printer with that configuration.

Since `uvm_printer` does not natively support pack/unpack operations, these helper methods can be used to pack/unpack a printer.

Note

The `uvm_packer` used must have the `use_metadata` flag set to 0b1 for this to work correctly

Parameters

<i>packer</i>	The <code>uvm_packer</code> that was previously used to pack a <code>uvm_printer</code>
---------------	---

Returns

A `uvm_printer` with the packed configuration

Definition at line 314 of file cl_syoscb_printer_config.svh.

References `get_printer_of_type()`.

13.90.2.4 get_file_descriptor()

```
int cl_syoscb_printer_config::get_file_descriptor (
    uvm_printer printer ) [static]
```

Gets the file descriptor used for a given printer.

Parameters

<i>printer</i>	The printer for which to get the file descriptor
----------------	--

Returns

The file descriptor used by this printer

Definition at line 56 of file cl_syoscb_printer_config.svh.

13.90.2.5 get_printer_of_type()

```
uvm_printer cl_syoscb_printer_config::get_printer_of_type (
    t_printer_type ptype ) [static]
```

Generates a new printer of the correct type.

The type is one of the enum values defined in `pk_syoscb::t_printer_type` (TABLE, LINE, TREE or XML) If the given enum does not match one of the 4 valid options, an error is thrown

Parameters

<i>ptype</i>	The type of printer which should be generated
--------------	---

Returns

A printer of the indicated type, null if the printer type was not recognized

Definition at line 107 of file `cl_syoscb_printer_config.svh`.

Referenced by `do_help_unpack()`.

13.90.2.6 get_printer_type()

```
t_printer_type cl_syoscb_printer_config::get_printer_type (
    uvm_printer printer ) [static]
```

Gets the type of printer that a given `uvm_printer` represents.

The valid printer types are limited to `uvm_table_printer`, `uvm_line_printer`, `uvm_tree_printer` and [uvm_xml_printer](#). If the given printer does not match one of these 4 types, an error is thrown

Parameters

<i>printer</i>	The printer for which the type should be found
----------------	--

Returns

A SYOSCB_PRINTER_TYPE enum indicating which type of printer was passed

Definition at line 71 of file `cl_syoscb_printer_config.svh`.

Referenced by `do_help_pack()`.

13.90.2.7 set_file_descriptor()

```
void cl_syoscb_printer_config::set_file_descriptor (
    uvm_printer printer,
    int fd = 0 ) [static]
```

Sets the file descriptor to be used for a given printer.

Parameters

<i>printer</i>	The printer on which to set the file descriptor
<i>fd</i>	The file descriptor

Definition at line 44 of file cl_syoscb_printer_config.svh.

Referenced by cl_syoscb_queue_base::dump(), and cl_syoscb_queue_base::dump_orphans_to_file().

13.90.2.8 set_printer_begin_elements()

```
void cl_syoscb_printer_config::set_printer_begin_elements (
    uvm_printer printer,
    int elements ) [static]
```

Sets the number of elements to print at the head of a list whenever the printer is used to print a tx item.

Parameters

<i>printer</i>	The printer to set the number of elements for
<i>elements</i>	The number of elements to print

Definition at line 441 of file cl_syoscb_printer_config.svh.

13.90.2.9 set_printer_end_elements()

```
void cl_syoscb_printer_config::set_printer_end_elements (
    uvm_printer printer,
    int elements ) [static]
```

Sets the number of elements to print at the tail of a list whenever the printer is used to print a tx item.

Parameters

<i>printer</i>	The printer to set the number of elements for
<i>elements</i>	The number of elements to print

Definition at line 454 of file `cl_syoscb_printer_config.svh`.

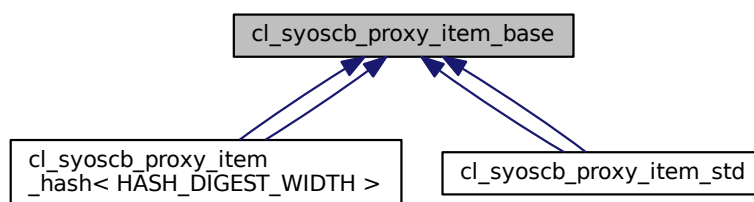
The documentation for this class was generated from the following files:

- `cl_syoscb_printer_config.svh`
- `pk_syoscb.sv`

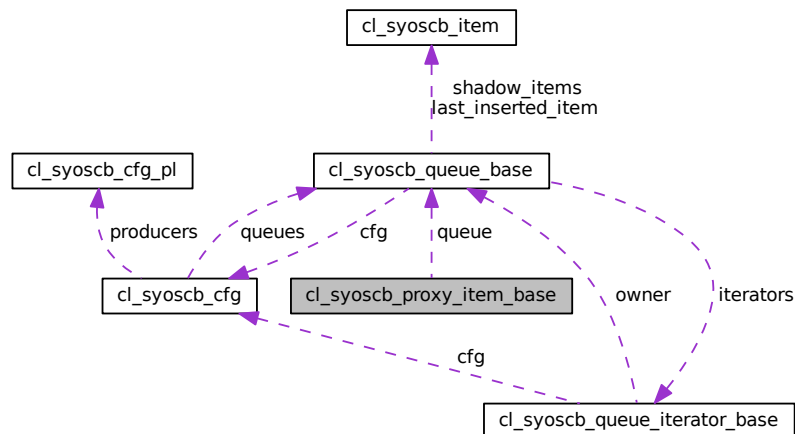
13.91 `cl_syoscb_proxy_item_base` Class Reference

Base class for all proxy items.

Inheritance diagram for `cl_syoscb_proxy_item_base`:



Collaboration diagram for `cl_syoscb_proxy_item_base`:



Public Member Functions

- virtual `cl_syoscb_item get_item ()`
Item API: Get the scoreboard item that this proxy item represents
- virtual void `set_queue (cl_syoscb_queue_base queue)`
Item API: Sets the queue that the referenced item belongs to
- virtual `cl_syoscb_queue_base get_queue ()`
Item API: Gets the queue that this proxy item depends on

13.91.1 Detailed Description

Base class for all proxy items.

A proxy item is used to decouple the act of iterating over a queue from the queue's implementation. Proxy items encode information that specify where in a given queue a specific [cl_syoscb_item](#) can be found.

Definition at line 4 of file cl_syoscb_proxy_item_base.svh.

13.91.2 Member Function Documentation

13.91.2.1 get_item()

```
cl_syoscb_item cl_syoscb_proxy_item_base::get_item ( ) [virtual]
```

Item API: Get the scoreboard item that this proxy item represents

Returns

That item

Definition at line 32 of file cl_syoscb_proxy_item_base.svh.

References [cl_syoscb_queue_base::get_item\(\)](#).

Referenced by [cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::delete_item\(\)](#), [cl_syoscb_queue_base::dump_orphans_to_file\(\)](#), [cl_syoscb_queue_base::dump_orphans_to_stdout\(\)](#), [cl_syoscb_queue_locator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::search\(\)](#), [cl_syoscb_queue_locator_std::search\(\)](#), [cl_syoscb_queue_locator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::validate_match\(\)](#), and [cl_syoscb_queue_locator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::validate_no_match\(\)](#).

13.91.2.2 get_queue()

```
cl_syoscb_queue_base cl_syoscb_proxy_item_base::get_queue ( ) [virtual]
```

Item API: Gets the queue that this proxy item depends on

Returns

A handle to that queue

Definition at line 44 of file cl_syoscb_proxy_item_base.svh.

Referenced by [cl_syoscb_queue_locator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::search\(\)](#).

13.91.2.3 set_queue()

```
void cl_syoscb_proxy_item_base::set_queue (
    cl_syoscb_queue_base queue ) [virtual]
```

Item API: Sets the queue that the referenced item belongs to

Parameters

A	handle to the queue
---	---------------------

Definition at line 38 of file `cl_syoscb_proxy_item_base.svh`.

Referenced by `cl_syoscb_queue_iterator_std::get_item_proxy()`, `cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::get_item_proxy()`, `cl_syoscb_queue_locator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::search()`, and `cl_syoscb_queue_locator_std::search()`.

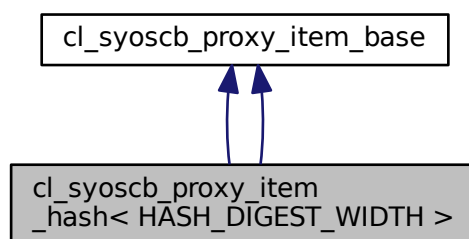
The documentation for this class was generated from the following files:

- `cl_syoscb_proxy_item_base.svh`
- `pk_syoscb.sv`

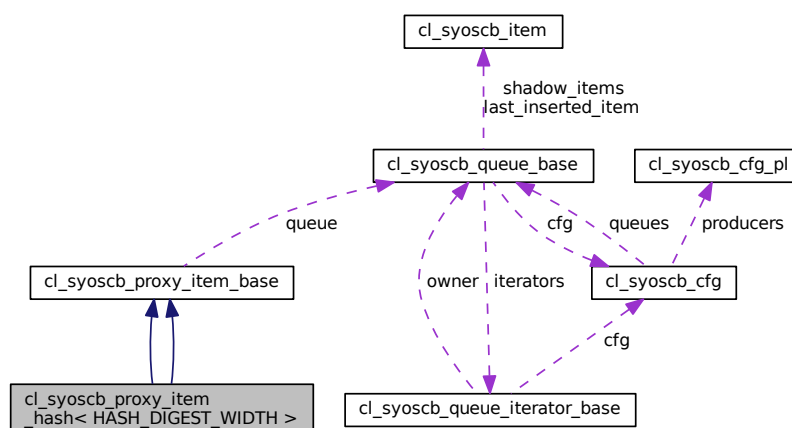
13.92 `cl_syoscb_proxy_item_hash< HASH_DIGEST_WIDTH >` Class Template Reference

Proxy item implementation for hash queues.

Inheritance diagram for `cl_syoscb_proxy_item_hash< HASH_DIGEST_WIDTH >`:



Collaboration diagram for `cl_syoscb_proxy_item_hash< HASH_DIGEST_WIDTH >`:



Public Attributes

- `cl_syoscb_hash_base< HASH_DIGEST_WIDTH >::tp_hash_digest digest`
The digest for the hashed scoreboard item.
- `int unsigned idx = 0`
The index in the `cl_syoscb_hash_item` with that digest where the item is located.

Additional Inherited Members

13.92.1 Detailed Description

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
class cl_syoscb_proxy_item_hash< HASH_DIGEST_WIDTH >
```

Proxy item implementation for hash queues.

Contains a reference to the digest value of the item for easy AA lookup.

Parameters

<code>HASH_DIGEST_WIDTH</code>	Number of bits used for hash digests in the used hash algorithm
--------------------------------	---

Definition at line 4 of file `cl_syoscb_proxy_item_hash.svh`.

13.92.2 Member Data Documentation

13.92.2.1 `idx`

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
int unsigned cl_syoscb_proxy_item_hash< HASH_DIGEST_WIDTH >::idx = 0
```

The index in the `cl_syoscb_hash_item` with that digest where the item is located.

This field is only really used when hash collisions occur (very rarely)

Definition at line 13 of file `cl_syoscb_proxy_item_hash.svh`.

Referenced by `cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::delete_item()`, `cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::get_item()`, `cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::get_item_proxy()`, and `cl_syoscb_queue_locator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::search()`.

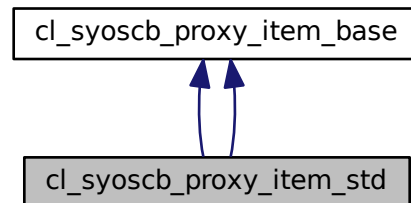
The documentation for this class was generated from the following files:

- `cl_syoscb_proxy_item_hash.svh`
- `pk_syoscb.sv`

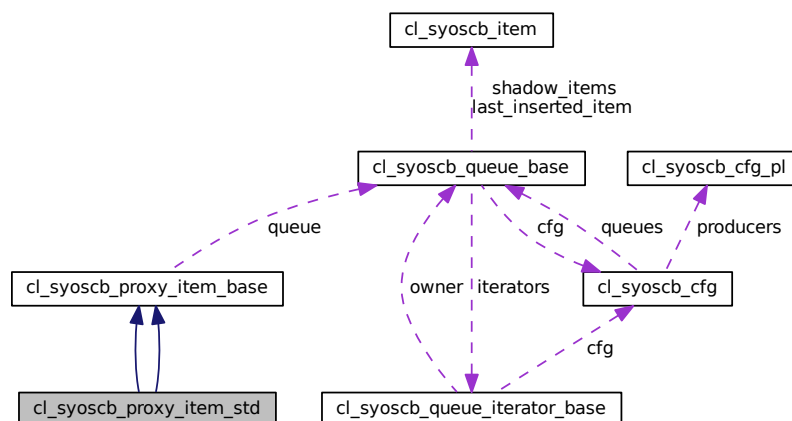
13.93 cl_syoscb_proxy_item_std Class Reference

Proxy item implementation for standard queues.

Inheritance diagram for cl_syoscb_proxy_item_std:



Collaboration diagram for cl_syoscb_proxy_item_std:



Public Attributes

- int unsigned `idx`
Position in the queue.

Additional Inherited Members

13.93.1 Detailed Description

Proxy item implementation for standard queues.

Contains the index in the queue at which the item is located.

Definition at line 3 of file cl_syoscb_proxy_item_std.svh.

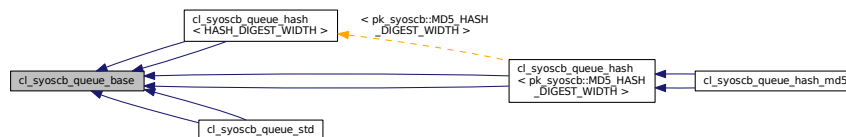
The documentation for this class was generated from the following files:

- cl_syoscb_proxy_item_std.svh
- pk_syoscb.sv

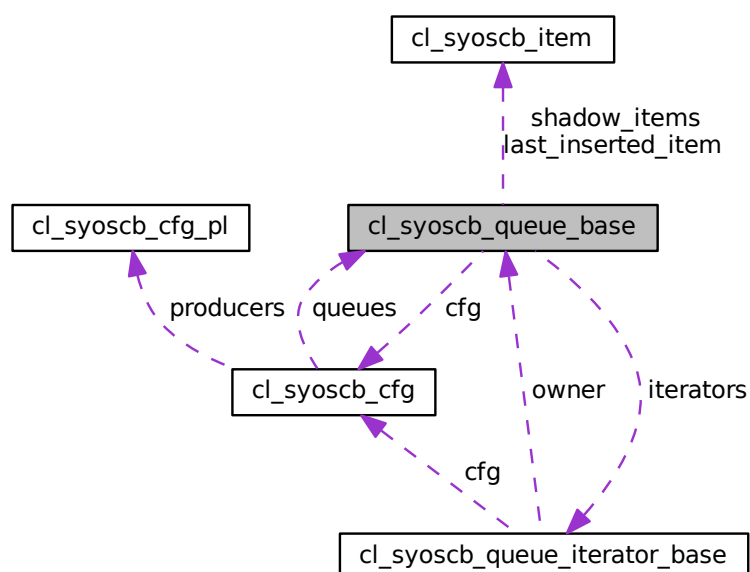
13.94 cl_syoscb_queue_base Class Reference

Class which represents the base concept of a queue.

Inheritance diagram for cl_syoscb_queue_base:



Collaboration diagram for cl_syoscb_queue_base:



Public Member Functions

- void [build_phase](#) (uvm_phase phase)
UVM Build Phase. Gets the scoreboard configuration for this SCB.
- void [check_phase](#) (uvm_phase phase)
UVM check phase.
- virtual bit [add_item](#) (string producer, uvm_sequence_item item)
Queue API: Adds a *uvm_sequence_item* to this queue.
- virtual bit [delete_item](#) ([cl_syoscb_proxy_item_base](#) proxy_item)
Queue API: Deletes the item indicated by the proxy item from the queue.
- virtual void [dump](#) (uvm_printer printer=null, int fd=UVM_STDOUT)
Queue API: Loop over all the items in the shadow queue and dump them.
- virtual [cl_syoscb_item](#) [get_item](#) ([cl_syoscb_proxy_item_base](#) proxy_item)
Queue API: Gets the item pointed to by the proxy item from the queue.
- virtual int unsigned [get_size](#) ()
Queue API: Returns the current size of the queue.
- virtual bit [empty](#) ()
Queue API: Returns whether or not the queue is empty.
- virtual bit [insert_item](#) (string producer, uvm_sequence_item item, int unsigned idx)
Queue API: Inserts a *uvm_sequence_item* at index *idx*.
- virtual void [flush_queue](#) ()
Queue API: Deletes all elements from the queue.
- virtual [cl_syoscb_queue_iterator_base](#) [create_iterator](#) (string name="")
Queue API: Creates an iterator for this queue.
- virtual [cl_syoscb_queue_iterator_base](#) [get_iterator](#) (string name)
Queue API: Gets the iterator from this queue with a given name.
- virtual bit [delete_iterator](#) ([cl_syoscb_queue_iterator_base](#) iterator)
Queue API: Deletes an iterator from this queue.
- virtual [cl_syoscb_queue_locator_base](#) [get_locator](#) ()
Queue API: Creates a locator for this queue.
- virtual bit [exists_cnt_producer](#) (string producer)
Queue API: Check if a given producer exists in the producer counter for this queue
- virtual int unsigned [get_cnt_producer](#) (string producer)
Queue API: Get the producer count for a given producer.
- virtual int unsigned [get_cnt_add_item](#) ()
Queue API: Returns the number of items that have been inserted in this queue
- virtual int unsigned [get_max_items](#) ()
Queue API: Returns the maximum number of elements that have been in the queue.
- virtual int unsigned [get_cnt_flushed_item](#) ()
Queue API: Returns the total number of elements flushed from this queue.
- virtual int unsigned [get_cnt_matched_item](#) ()
Queue API: Returns the total number of elements matched in this queue
- virtual [cl_syoscb_item](#) [get_last_inserted_item](#) ()
Queue API: Gets the last inserted item in the queue
- virtual string [get_failed_checks](#) ()
Queue API: Gets a string containing all queue checks that this queue have failed.
- virtual string [create_queue_report](#) (int unsigned offset, int unsigned first_column_width)
Queue API: Returns a string with overall queues statistics.

Public Attributes

- [cl_syoscb_item shadow_items](#) [\$]
Shadow queue tracking all items inserted into the queue, used for scoreboard dumps.

Protected Member Functions

- virtual [cl_syoscb_item pre_add_item](#) (string producer, uvm_sequence_item item)
Perform some basic bookkeeping that is the same for all sequence items before insertion.
- virtual void [post_add_item](#) ([cl_syoscb_item](#) item)
Perform some basic bookkeeping that is the same for all sequence items after insertion.
- virtual void [do_flush_queue](#) ()
Performs the actual element deletion from the queue when called by [flush_queue](#).
- virtual void [incr_cnt_producer](#) (string producer)
Increment the producer counter for a given producer.
- virtual void [decr_cnt_producer](#) (string producer)
Decrement the producer counter for a given producer.
- virtual void [dump_orphans_to_file](#) ()
Dumps orphans remaining in the queue into a logfile.
- virtual void [dump_orphans_to_stdout](#) ()
Prints orphans remaining in the queue to stdout.
- virtual string [create_producer_stats](#) (int unsigned offset, int unsigned first_column_width)
Returns a table with statistics for all producers in this queue.
- virtual string [get_dump_extension](#) (t_dump_type dump_type)
Gets the file extension to be used for a dump file.
- virtual void [print_orphan_xml_header](#) (int fd)
Prints the header for an XML orphan dump.
- virtual void [print_orphan_xml_footer](#) (int fd)
Prints the footer for an XML orphan dump.

Protected Attributes

- [cl_syoscb_cfg](#) cfg
Handle to the configuration.
- [cl_syoscb_queue_iterator_base](#) iterators [[cl_syoscb_queue_iterator_base](#)]
List of iterators registered with this queue.
- semaphore [iter_sem](#)
Semaphore guarding exclusive access to the queue when multiple iterators are in play.
- int unsigned [cnt_producer](#) [string]
Associative array counting the number of items by a given producer that currently exist in the queue.
- int unsigned [cnt_add_item](#) = 0
Number of items that have been inserted into this queue.
- int unsigned [max_items](#) = 0
Maximum number of items that have been in this queue so far.
- [cl_syoscb_item](#) [last_inserted_item](#)
The most recently inserted item in this queue.
- int [num_iters_created](#) = 0
The number of iterators that have been created for this queue so far.

Private Attributes

- int unsigned `nbr_items_dumped`
Number of items that have been dumped from this queue when performing a scoreboard dump.
- int unsigned `total_cnt_producer` [string]
Associative array counting the total number of items by a given producer that have been inserted in the queue.
- int unsigned `total_cnt_flushed_producer` [string]
Associative array counter the total number of items by a given producer that have been flused form the queue.
- string `failed_checks` [string]
AA for storing queue debug checks during the UVM check phase.

13.94.1 Detailed Description

Class which represents the base concept of a queue.

All queues must extend this class and implement the queue API.

Definition at line 3 of file `cl_syoscb_queue_base.svh`.

13.94.2 Member Function Documentation

13.94.2.1 `add_item()`

```
bit cl_syoscb_queue_base::add_item (
    string producer,
    uvm_sequence_item item ) [virtual]
```

Queue API: Adds a `uvm_sequence_item` to this queue.

The basic job of the `add_item` method is:

1. Create the new `cl_syoscb_item` and give it a unique name
2. Set the producer and other metadata of the scoreboard item
3. Wrap the `uvm_sequence_item` inside the scoreboard item
4. Insert the item into the queue and shadow queue
5. Update the producer counter and insert counter

Parameters

<i>producer</i>	The producer of the sequence item
<i>item</i>	The item that should be add to the queue

Returns

1 if the item was successfully added, 0 otherwise

Note

Abstract method. Must be implemented in a subclass

Reimplemented in [cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_std](#), and [cl_syoscb_queue_std](#).

Definition at line 186 of file [cl_syoscb_queue_base.svh](#).

Referenced by [cl_syoscb::add_item\(\)](#).

13.94.2.2 check_phase()

```
void cl_syoscb_queue_base::check_phase (
    uvm_phase phase )
```

UVM check phase.

Checks if the queue is empty and if it had zero insertions. If either is true, a UVM_ERROR is generated in [cl_syoscb](#)

Definition at line 150 of file [cl_syoscb_queue_base.svh](#).

References [cfg](#), [dump_orphans_to_file\(\)](#), [dump_orphans_to_stdout\(\)](#), [empty\(\)](#), [failed_checks](#), [cl_syoscb_cfg::get_dump_orphans_to_files\(\)](#), [cl_syoscb_cfg::get_enable_no_insert_check\(\)](#), [cl_syoscb_cfg::get_max_print_orphans\(\)](#), and [get_size\(\)](#).

13.94.2.3 create_iterator()

```
cl_syoscb_queue_iterator_base cl_syoscb_queue_base::create_iterator (
    string name = "" ) [virtual]
```

Queue API: Creates an iterator for this queue.

Iterators are by default named "[name]_iter[X]", where [name] is the name of the queue, and [X] is the number of iterators that have previously been created for this queue

Parameters

<i>name</i>	A name to be used for the iterator. If an iterator with this name already exists, prints a UVM_DEBUG message
-------------	--

Returns

An iterator over this queue, or null if a queue with the requested name already exists

Reimplemented in [cl_syoscb_queue_std](#), [cl_syoscb_queue_std](#), [cl_syoscb_queue_hash_md5](#), and [cl_syoscb_queue_hash_md5](#).

Definition at line 320 of file `cl_syoscb_queue_base.svh`.

Referenced by `cl_scb_test_iterator_unit_tests::check_first()`, `cl_scb_test_iterator_unit_tests::check_flush()`, `cl_scb_test_iterator_unit_tests::check_last()`, `cl_scb_test_iterator_unit_tests::check_names()`, `cl_scb_test_iterator_unit_tests::check_next()`, `cl_scb_test_iterator_unit_tests::check_prev()`, `cl_scb_test_iterator_unit_tests::check_set_queue()`, `cl_syoscb_compare_base::create_primary_iterator()`, `dump_orphans_to_file()`, `dump_orphans_to_stdout()`, `cl_syoscb_compare_ooo::get_count_producer()`, `cl_syoscb_compare_iop::get_count_producer()`, `cl_syoscb_compare_io_2hp::primary_loop_do()`, `cl_syoscb_compare_io::secondary_loop_do()`, and `cl_syoscb_compare_iop::secondary_loop_do()`.

13.94.2.4 create_producer_stats()

```
string cl_syoscb_queue_base::create_producer_stats (
    int unsigned offset,
    int unsigned first_column_width ) [protected], [virtual]
```

Returns a table with statistics for all producers in this queue.

Outputs the number of insertions, matches, flushed items and orphans per-producer.

Parameters

<i>offset</i>	The x-offset that should be used when printing producer names
<i>first_column_width</i>	The width of the first column in the table

Returns

A string containing producer stats for all producers in this queue.

Definition at line 597 of file `cl_syoscb_queue_base.svh`.

References `cfg`, `cnt_producer`, `cl_syoscb_cfg::get_producers()`, `cl_syoscb_string_library::pad_str()`, `total_cnt_flushed_producer`, and `total_cnt_producer`.

Referenced by `create_queue_report()`.

13.94.2.5 create_queue_report()

```
string cl_syoscb_queue_base::create_queue_report (
    int unsigned offset,
    int unsigned first_column_width ) [virtual]
```

Queue API: Returns a string with overall queues statistics.

Reports the number of insertions, matches, flushed items and orphans. If `cl_syoscb_cfg::enable_queue_stats` is 1, also includes per-producer statistics (see [create_producer_stats](#))

Parameters

<i>offset</i>	The x-offset that should be used when printing the queue name names
<i>first_column_width</i>	The width of the first column in the table

Returns

A string containing overall queues statistics.

Definition at line 638 of file cl_syoscb_queue_base.svh.

References `cfg`, `create_producer_stats()`, `get_cnt_add_item()`, `get_cnt_flushed_item()`, `get_cnt_matched_item()`, `cl_syoscb_cfg::get_enable_queue_stats()`, `get_size()`, and `cl_syoscb_string_library::pad_str()`.

Referenced by `cl_syoscb::create_queues_stats()`, and `cl_syoscb::intermediate_queue_stat_dump()`.

13.94.2.6 `decr_cnt_producer()`

```
void cl_syoscb_queue_base::decr_cnt_producer (
    string producer ) [protected], [virtual]
```

Decrement the producer counter for a given producer.

Parameters

<i>producer</i>	The producer to decrement the counter for
-----------------	---

Definition at line 380 of file cl_syoscb_queue_base.svh.

References `cnt_producer`.

Referenced by `cl_syoscb_queue_std::delete_item()`, and `cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::delete_item()`.

13.94.2.7 `delete_item()`

```
bit cl_syoscb_queue_base::delete_item (
    cl_syoscb_proxy_item_base proxy_item ) [virtual]
```

Queue API: Deletes the item indicated by the proxy item from the queue.

The basic job of the `delete_item` method is:

1. Delete the element
2. Notify any iterators, moving them as necessary
3. Update the producer counter for the deleted item's producer

Parameters

<i>proxy_item</i>	A proxy item indicating which scoreboard item to delete from the queue
-------------------	--

Returns

if the item was successfully deleted, 0 otherwise

Note

Abstract method. Must be implemented in a subclass

Reimplemented in [cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_std](#), and [cl_syoscb_queue_std](#).

Definition at line 200 of file `cl_syoscb_queue_base.svh`.

Referenced by `cl_syoscb_compare_base::delete()`.

13.94.2.8 delete_iterator()

```
bit cl_syoscb_queue_base::delete_iterator (
    cl_syoscb_queue_iterator_base iterator ) [virtual]
```

Queue API: Deletes an iterator from this queue.

Parameters

<i>iterator</i>	The iterator to delete
-----------------	------------------------

Returns

1 if the iterator was successfully deleted, 0 otherwise

Reimplemented in [cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_std](#), and [cl_syoscb_queue_std](#).

Definition at line 345 of file `cl_syoscb_queue_base.svh`.

Referenced by `cl_scb_test_iterator_unit_tests::check_first()`, `cl_scb_test_iterator_unit_tests::check_flush()`, `cl_scb_test_iterator_unit_tests::check_last()`, `cl_scb_test_iterator_unit_tests::check_names()`, `cl_scb_test_iterator_unit_tests::check_next()`, `cl_scb_test_iterator_unit_tests::check_prev()`, `cl_scb_test_iterator_unit_tests::check_set_queue()`, `dump_orphans_to_file()`, and `dump_orphans_to_stdout()`.

13.94.2.9 dump()

```
void cl_syoscb_queue_base::dump (
    uvm_printer printer = null,
    int fd = UVM_STDOUT ) [virtual]
```

Queue API: Loop over all the items in the shadow queue and dump them.

If a printer has not been passed in the arguments, used [cl_syoscb_cfg::get_printer](#) to lookup a printer for each shadow item (which may be quite inefficient). If `cl_syoscb_cfg::full_scb_type` has been set to XML, the XML printer is used, overriding any specific printers that have been set

Parameters

<i>printer</i>	The printer to use when dumping items. Defaults to null, getting a queue/producer specific printer for each item
<i>fd</i>	File descriptor for where to dump items. Defaults to STDOUT

Definition at line 438 of file `cl_syoscb_queue_base.svh`.

References `cfg`, `cl_syoscb_item::convert2string()`, `cl_syoscb_cfg::get_default_printer()`, `cl_syoscb_cfg::get_↵
enable_c2s_full_scb_dump()`, `cl_syoscb_cfg::get_full_scb_dump()`, `cl_syoscb_cfg::get_full_scb_dump_type()`, `cl_↵
_syoscb_cfg::get_printer()`, `cl_syoscb_item::get_producer()`, `nbr_items_dumped`, `cl_syoscb_printer_config::set_↵
file_descriptor()`, `cl_syoscb_item::set_queue_index()`, and `shadow_items`.

Referenced by `cl_syoscb::dump_join_txt()`, `cl_syoscb::dump_join_xml()`, `cl_syoscb::dump_split_txt()`, and `cl_↵
syoscb::dump_split_xml()`.

13.94.2.10 dump_orphans_to_file()

```
void cl_syoscb_queue_base::dump_orphans_to_file ( ) [protected], [virtual]
```

Dumps orphans remaining in the queue into a logfile.

Assumes that the caller has checked whether [cl_syoscb_cfg::dump_orphans_to_files](#) is set

Definition at line 521 of file `cl_syoscb_queue_base.svh`.

References `cfg`, `create_iterator()`, `delete_iterator()`, `cl_syoscb_queue_iterator_base::first()`, `cl_syoscb_cfg::get_↵
_default_printer()`, `get_dump_extension()`, `cl_syoscb_proxy_item_base::get_item()`, `cl_syoscb_cfg::get_max_↵
print_orphans()`, `cl_syoscb_cfg::get_orphan_dump_file_name()`, `cl_syoscb_cfg::get_orphan_dump_type()`, `cl_↵
syoscb_cfg::get_printer()`, `cl_syoscb_item::get_producer()`, `cl_syoscb_cfg::get_scb_name()`, `cl_syoscb_queue_↵
iterator_base::has_next()`, `cl_syoscb_queue_iterator_base::next()`, `cl_syoscb_queue_iterator_base::next_index()`, `cl_syoscb_queue_iterator_base::previous_index()`, `print_orphan_xml_footer()`, `print_orphan_xml_header()`, `cl_↵
syoscb_printer_config::set_file_descriptor()`, and `cl_syoscb_item::set_queue_index()`.

Referenced by `check_phase()`.

13.94.2.11 dump_orphans_to_stdout()

```
void cl_syoscb_queue_base::dump_orphans_to_stdout ( ) [protected], [virtual]
```

Prints orphans remaining in the queue to stdout.

The number of orphans that are printed depends on [cl_syoscb_cfg::max_print_orphans](#)

Definition at line 484 of file `cl_syoscb_queue_base.svh`.

References `cfg`, `create_iterator()`, `delete_iterator()`, `cl_syoscb_queue_iterator_base::first()`, `cl_syoscb_proxy_↵
item_base::get_item()`, `cl_syoscb_cfg::get_max_print_orphans()`, `cl_syoscb_cfg::get_orphans_as_errors()`, `cl_↵
syoscb_queue_iterator_base::has_next()`, `cl_syoscb_queue_iterator_base::next()`, `cl_syoscb_queue_iterator_↵
base::next_index()`, `cl_syoscb_queue_iterator_base::previous_index()`, and `cl_syoscb_item::set_queue_index()`.

Referenced by `check_phase()`.

13.94.2.12 empty()

```
bit cl_syoscb_queue_base::empty ( ) [virtual]
```

Queue API: Returns whether or not the queue is empty.

Returns

1 if the queue is empty, 0 otherwise

Note

Abstract method. Must be implemented in a subclass

Reimplemented in [cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_std](#), and [cl_syoscb_queue_std](#).

Definition at line 266 of file `cl_syoscb_queue_base.svh`.

Referenced by `check_phase()`, `cl_syoscb_compare_base::check_queues()`, and `cl_syoscb::empty_queues()`.

13.94.2.13 exists_cnt_producer()

```
bit cl_syoscb_queue_base::exists_cnt_producer (
    string producer ) [virtual]
```

Queue API: Check if a given producer exists in the producer counter for this queue

Parameters

<i>producer</i>	The producer to check for existence 1 if the producer exists, 0 otherwise
-----------------	---

Definition at line 395 of file cl_syoscb_queue_base.svh.

References cnt_producer.

Referenced by cl_syoscb_compare_base::count_producers().

13.94.2.14 flush_queue()

```
void cl_syoscb_queue_base::flush_queue ( ) [virtual]
```

Queue API: Deletes all elements from the queue.

Updates the flush counter, sets all producer counts to 0 and resets all iterators.

Definition at line 288 of file cl_syoscb_queue_base.svh.

References cfg, cnt_producer, do_flush_queue(), cl_syoscb_cfg::get_producers(), iter_sem, iterators, and total_↔ cnt_flushed_producer.

Referenced by cl_syoscb::flush_queues().

13.94.2.15 get_cnt_producer()

```
int unsigned cl_syoscb_queue_base::get_cnt_producer (
    string producer ) [virtual]
```

Queue API: Get the producer count for a given producer.

Parameters

<i>producer</i>	The producer to get count for
-----------------	-------------------------------

Returns

The number of items in the queue that were from the given producer

Note

May *ONLY* be called if the producer exists (see [exists_cnt_producer](#))

Definition at line 403 of file cl_syoscb_queue_base.svh.

References cnt_producer.

Referenced by cl_syoscb_compare_base::count_producers().

13.94.2.16 get_dump_extension()

```
string cl_syoscb_queue_base::get_dump_extension (
    t_dump_type dump_type ) [protected], [virtual]
```

Gets the file extension to be used for a dump file.

Parameters

<i>dump_type</i>	The type of dump that should be performed.
------------------	--

Returns

A string with the file extension that should be used for that kind of dump

Definition at line 659 of file cl_syoscb_queue_base.svh.

Referenced by dump_orphans_to_file().

13.94.2.17 get_failed_checks()

```
string cl_syoscb_queue_base::get_failed_checks ( ) [virtual]
```

Queue API: Gets a string containing all queue checks that this queue have failed.

Failed checks include having orphans at the end of simulation, and not having any insertions

Returns

A string containing all failed checks for this queue

Definition at line 577 of file cl_syoscb_queue_base.svh.

References failed_checks.

Referenced by cl_syoscb::get_queue_failed_checks().

13.94.2.18 get_item()

```
cl_syoscb_item cl_syoscb_queue_base::get_item (
    cl_syoscb_proxy_item_base proxy_item ) [virtual]
```

Queue API: Gets the item pointed to by the proxy item from the queue.

If the proxy item does not specify a valid item in the queue, print a UVM_INFO/DEBUG message

Parameters

<i>proxy_item</i>	A proxy item indicating which scoreboard item to delete from the queue
-------------------	--

Returns

The scoreboard item indicated by the proxy item, null if the proxy item did not point to a valid item

Note

Abstract method. Must be implemented in a subclass

Reimplemented in [cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_std](#), and [cl_syoscb_queue_std](#).

Definition at line 248 of file `cl_syoscb_queue_base.svh`.

Referenced by `cl_syoscb_compare_ooo::get_count_producer()`, `cl_syoscb_compare_iop::get_count_producer()`, `cl_syoscb_proxy_item_base::get_item()`, `cl_syoscb_compare_io::primary_loop_do()`, `cl_syoscb_compare_io_2hp::primary_loop_do()`, `cl_syoscb_compare_iop::primary_loop_do()`, `cl_syoscb_compare_io::secondary_loop_do()`, and `cl_syoscb_compare_iop::secondary_loop_do()`.

13.94.2.19 get_iterator()

```
cl_syoscb_queue_iterator_base cl_syoscb_queue_base::get_iterator (
    string name ) [virtual]
```

Queue API: Gets the iterator from this queue with a given name.

If no queue exists with that name, returns null

Parameters

<i>name</i>	The name of the queue to lookup
-------------	---------------------------------

Returns

That iterator, if it exists, or null if no such queue exists

Note

Will raise a UVM_ERROR if multiple iterators with the same name exist

Definition at line 330 of file `cl_syoscb_queue_base.svh`.

References `cl_syoscb_queue_iterator_base`.

Referenced by `cl_syoscb_compare_base::create_primary_iterator()`, `cl_syoscb_compare_io_2hp::primary_loop_do()`, `cl_syoscb_compare_io::secondary_loop_do()`, and `cl_syoscb_compare_iop::secondary_loop_do()`.

13.94.2.20 `get_locator()`

```
cl_syoscb_queue_locator_base cl_syoscb_queue_base::get_locator ( ) [virtual]
```

Queue API: Creates a locator for this queue.

Returns

A locator over this queue

Reimplemented in [cl_syoscb_queue_std](#), [cl_syoscb_queue_std](#), [cl_syoscb_queue_hash_md5](#), and [cl_syoscb_queue_hash_md5](#).

Definition at line 352 of file `cl_syoscb_queue_base.svh`.

Referenced by `cl_syoscb_compare_ooo::secondary_loop_do()`.

13.94.2.21 `get_size()`

```
int unsigned cl_syoscb_queue_base::get_size ( ) [virtual]
```

Queue API: Returns the current size of the queue.

Returns

Number of items currently in the queue

Note

Abstract method. Must be implemented in a subclass

Reimplemented in [cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_std](#), and [cl_syoscb_queue_std](#).

Definition at line 257 of file `cl_syoscb_queue_base.svh`.

Referenced by `cl_syoscb::add_item()`, `cl_scb_test_iterator_unit_tests::check_last()`, `cl_scb_test_iterator_unit_tests::check_next()`, `check_phase()`, `create_queue_report()`, `cl_syoscb_compare_base::dynamic_queue_split_do()`, `cl_syoscb_queue_iterator_std::first()`, `get_cnt_matched_item()`, `cl_syoscb_compare_base::get_queues_item_cnt()`, `cl_syoscb::get_total_queue_size()`, `cl_syoscb_queue_iterator_std::has_next()`, `cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::has_next()`, `cl_syoscb_queue_iterator_std::last()`, and `post_add_item()`.

13.94.2.22 `incr_cnt_producer()`

```
void cl_syoscb_queue_base::incr_cnt_producer (
    string producer ) [protected], [virtual]
```

Increment the producer counter for a given producer.

Parameters

<i>producer</i>	The producer to increment the counter for
-----------------	---

Definition at line 363 of file cl_syoscb_queue_base.svh.

References `cnt_producer`, `total_cnt_flushed_producer`, and `total_cnt_producer`.

Referenced by `post_add_item()`.

13.94.2.23 insert_item()

```
bit cl_syoscb_queue_base::insert_item (
    string producer,
    uvm_sequence_item item,
    int unsigned idx ) [virtual]
```

Queue API: Inserts a `uvm_sequence_item` at index `idx`.

The method works in the same manner as [add_item](#), by doing the following:

1. Insert the a new item as the [add_item\(\)](#) method
2. Notify any iterators

Parameters

<i>producer</i>	The producer of the sequence item
<i>item</i>	The item that should be add to the queue
<i>idx</i>	The index at which the item should be inserted

Returns

1 if the item was successfully inserted, 0 otherwise

Note

Abstract method. Must be implemented in a subclass

Reimplemented in [cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_std](#), and [cl_syoscb_queue_std](#).

Definition at line 281 of file cl_syoscb_queue_base.svh.

13.94.2.24 post_add_item()

```
void cl_syoscb_queue_base::post_add_item (
    cl_syoscb_item item ) [protected], [virtual]
```

Perform some basic bookkeeping that is the same for all sequence items after insertion.

Parameters

<i>item</i>	The scoreboard item that has been inserted into the scoreboard
-------------	--

Definition at line 230 of file `cl_syoscb_queue_base.svh`.

References `cfg`, `cnt_add_item`, `cl_syoscb_cfg::get_full_scb_dump()`, `cl_syoscb_item::get_producer()`, `get_size()`, `incr_cnt_producer()`, `last_inserted_item`, `max_items`, and `shadow_items`.

Referenced by `cl_syoscb_queue_std::add_item()`, `cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::add_item()`, `cl_syoscb_queue_std::insert_item()`, and `cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::insert_item()`.

13.94.2.25 pre_add_item()

```
cl_syoscb_item cl_syoscb_queue_base::pre_add_item (
    string producer,
    uvm_sequence_item item ) [protected], [virtual]
```

Perform some basic bookkeeping that is the same for all sequence items before insertion.

Generates the scoreboard wrapper item

Parameters

<i>producer</i>	The producer of this item
<i>item</i>	The item to be inserted into the scoreboard

Returns

A scoreboard item, wrapping the given sequence item

Definition at line 210 of file `cl_syoscb_queue_base.svh`.

References `cnt_add_item`, `cl_syoscb_item::set_insertion_index()`, `cl_syoscb_item::set_item()`, and `cl_syoscb_item::set_producer()`.

Referenced by `cl_syoscb_queue_std::add_item()`, `cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::add_item()`, `cl_syoscb_queue_std::insert_item()`, and `cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::insert_item()`.

13.94.2.26 print_orphan_xml_footer()

```
void cl_syoscb_queue_base::print_orphan_xml_footer (
    int fd ) [protected], [virtual]
```

Prints the footer for an XML orphan dump.

Parameters

<i>fd</i>	File descriptor for the file to write the header into
-----------	---

Definition at line 688 of file cl_syoscb_queue_base.svh.

Referenced by dump_orphans_to_file().

13.94.2.27 print_orphan_xml_header()

```
void cl_syoscb_queue_base::print_orphan_xml_header (
    int fd ) [protected], [virtual]
```

Prints the header for an XML orphan dump.

Parameters

<i>fd</i>	File descriptor for the file to write the header into
-----------	---

Definition at line 672 of file cl_syoscb_queue_base.svh.

References `cfg`, and `cl_syoscb_cfg::get_scb_name()`.

Referenced by dump_orphans_to_file().

13.94.3 Member Data Documentation**13.94.3.1 failed_checks**

```
string cl_syoscb_queue_base::failed_checks [private]
```

AA for storing queue debug checks during the UVM check phase.

These values are used in [cl_syoscb::report_phase](#) and [cl_syoscb::check_phase](#)

Definition at line 46 of file cl_syoscb_queue_base.svh.

Referenced by `check_phase()`, and `get_failed_checks()`.

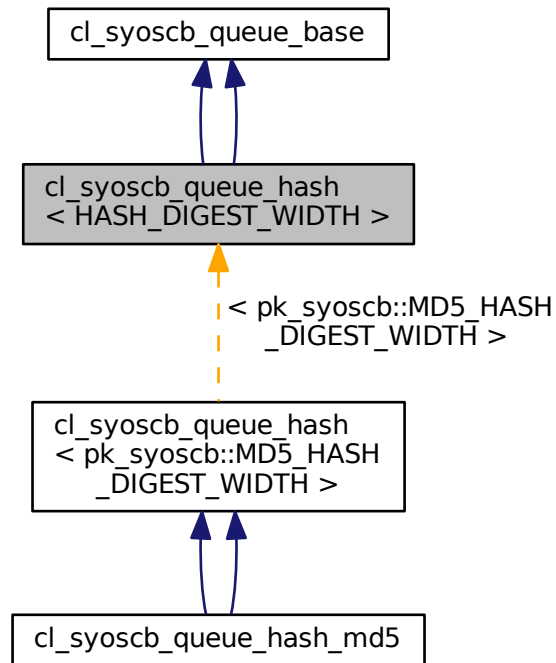
The documentation for this class was generated from the following files:

- cl_syoscb_queue_base.svh
- pk_syoscb.sv

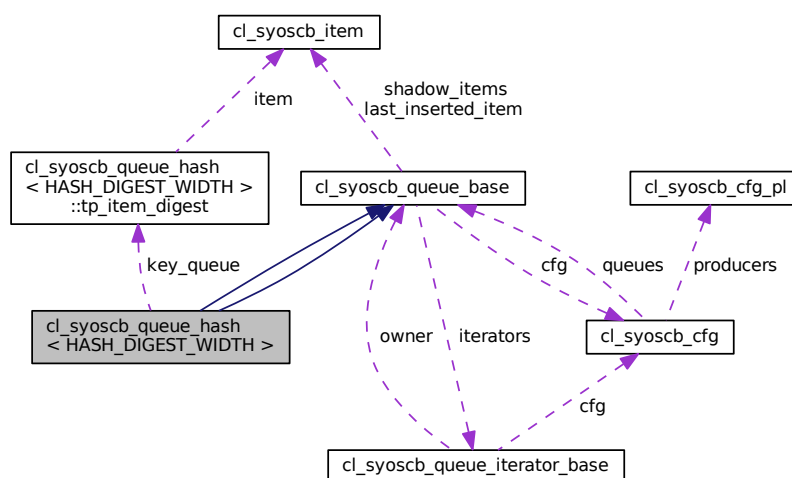
13.95 cl_syoscb_queue_hash< HASH_DIGEST_WIDTH > Class Template Reference

Class which represents the base concept of a hash queue.

Inheritance diagram for cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >:



Collaboration diagram for cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >:



Classes

- struct [packed](#)
Typedef for struct representing whether an option with an iterator was valid.
- struct [tp_item_digest](#)
Typedef for struct used to track items and their digests in the key queue.

Public Types

- typedef [cl_syoscb_hash_base](#)< HASH_DIGEST_WIDTH >::tp_hash_digest [tp_digest](#)
Typedef for hash algorithm digests.
- typedef [tp_item_digest](#) [tp_queue_of_keys](#)[\$]
Typedef for queue of digests and items.
- typedef [cl_syoscb_hash_aa_wrapper](#)< HASH_DIGEST_WIDTH > [tp_aa_hash](#)
Typedef for parameterized AA wrapper.
- typedef struct [cl_syoscb_queue_hash](#)::[packed](#) [tp_return_digest](#)
Typedef for struct representing whether an option with an iterator was valid.
- typedef [cl_syoscb_hash_base](#)< HASH_DIGEST_WIDTH >::tp_hash_digest [tp_digest](#)
Typedef for hash algorithm digests.
- typedef [tp_item_digest](#) [tp_queue_of_keys](#)[\$]
Typedef for queue of digests and items.
- typedef [cl_syoscb_hash_aa_wrapper](#)< HASH_DIGEST_WIDTH > [tp_aa_hash](#)
Typedef for parameterized AA wrapper.
- typedef struct [cl_syoscb_queue_hash](#)::[packed](#) [tp_return_digest](#)
Typedef for struct representing whether an option with an iterator was valid.

Public Member Functions

- virtual bit [add_item](#) (string producer, uvm_sequence_item item)
Queue API: See [cl_syoscb_queue_base::add_item](#) for more details
- virtual bit [delete_item](#) ([cl_syoscb_proxy_item_base](#) proxy_item)
Queue API: See [cl_syoscb_queue_base::delete_item](#) for more details
- virtual [cl_syoscb_item](#) [get_item](#) ([cl_syoscb_proxy_item_base](#) proxy_item)
Queue API: See [cl_syoscb_queue_base::get_item](#) for more details
- virtual int unsigned [get_size](#) ()
Queue API: See [cl_syoscb_queue_base::get_size](#) for more details.
- virtual bit [empty](#) ()
Queue API: See [cl_syoscb_queue_base::empty](#) for more details
- virtual bit [insert_item](#) (string producer, uvm_sequence_item item, int unsigned idx)
Queue API: See [cl_syoscb_queue_base::insert_item](#) for more details
- virtual bit [delete_iterator](#) ([cl_syoscb_queue_iterator_base](#) iterator)
Queue API: See [cl_syoscb_queue_base::delete_iterator](#) for more details
- virtual [tp_queue_of_keys](#) [get_key_queue](#) ()
Get the list of hash values of items in the queue.
- virtual [tp_aa_hash](#) [get_hash](#) ()
Gets the hash AA wrapper used for this queue.
- virtual bit [add_item](#) (string producer, uvm_sequence_item item)
Queue API: Adds a uvm_sequence_item to this queue.
- virtual bit [delete_item](#) ([cl_syoscb_proxy_item_base](#) proxy_item)

Queue API: Deletes the item indicated by the proxy item from the queue.

- virtual `cl_syoscb_item get_item (cl_syoscb_proxy_item_base proxy_item)`

Queue API: Gets the item pointed to by the proxy item from the queue.

- virtual `int unsigned get_size ()`

Queue API: Returns the current size of the queue.

- virtual `bit empty ()`

Queue API: Returns whether or not the queue is empty.

- virtual `bit insert_item (string producer, uvm_sequence_item item, int unsigned idx)`

Queue API: Inserts a `uvm_sequence_item` at index `idx`.

- virtual `bit delete_iterator (cl_syoscb_queue_iterator_base iterator)`

Queue API: Deletes an iterator from this queue.

Protected Member Functions

- virtual `void do_flush_queue ()`

See `cl_syoscb_queue_base::do_flush_queue` for more details.

- virtual `void do_flush_queue ()`

Performs the actual element deletion from the queue when called by `flush_queue`.

Protected Attributes

- `cl_syoscb_hash_base< HASH_DIGEST_WIDTH > hash_algo`

Handle to the implemented hash algorithm.

- `cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH > hash`

Queue implementation with an associative array. Wrapped in a class for performance reasons.

- `tp_item_digest key_queue [$]`

List of hash values of the items in the queue.

- `int unsigned size`

Size of queue, stored here to optimize for speed.

Additional Inherited Members

13.95.1 Detailed Description

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
class cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >
```

Class which represents the base concept of a hash queue.

All hash queues must extend this class and implement the queue API.

Definition at line 3 of file `cl_syoscb_queue_hash.svh`.

13.95.2 Member Function Documentation

13.95.2.1 `add_item()`

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
virtual bit cl\_syoscb\_queue\_hash< HASH_DIGEST_WIDTH >::add_item (
    string producer,
    uvm_sequence_item item ) [virtual]
```

Queue API: Adds a `uvm_sequence_item` to this queue.

The basic job of the `add_item` method is:

1. Create the new [cl_syoscb_item](#) and give it a unique name
2. Set the producer and other metadata of the scoreboard item
3. Wrap the `uvm_sequence_item` inside the scoreboard item
4. Insert the item into the queue and shadow queue
5. Update the producer counter and insert counter

Parameters

<i>producer</i>	The producer of the sequence item
<i>item</i>	The item that should be add to the queue

Returns

1 if the item was successfully added, 0 otherwise

Note

Abstract method. Must be implemented in a subclass

Reimplemented from [cl_syoscb_queue_base](#).

13.95.2.2 `delete_item()`

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
virtual bit cl\_syoscb\_queue\_hash< HASH_DIGEST_WIDTH >::delete_item (
    cl\_syoscb\_proxy\_item\_base proxy_item ) [virtual]
```

Queue API: Deletes the item indicated by the proxy item from the queue.

The basic job of the `delete_item` method is:

1. Delete the element
2. Notify any iterators, moving them as necessary
3. Update the producer counter for the deleted item's producer

Parameters

<i>proxy_item</i>	A proxy item indicating which scoreboard item to delete from the queue
-------------------	--

Returns

if the item was successfully deleted, 0 otherwise

Note

Abstract method. Must be implemented in a subclass

Reimplemented from [cl_syoscb_queue_base](#).

13.95.2.3 delete_iterator()

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
virtual bit cl\_syoscb\_queue\_hash< HASH_DIGEST_WIDTH >::delete_iterator (
    cl\_syoscb\_queue\_iterator\_base iterator ) [virtual]
```

Queue API: Deletes an iterator from this queue.

Parameters

<i>iterator</i>	The iterator to delete
-----------------	------------------------

Returns

1 if the iterator was successfully deleted, 0 otherwise

Reimplemented from [cl_syoscb_queue_base](#).

13.95.2.4 empty()

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
virtual bit cl\_syoscb\_queue\_hash< HASH_DIGEST_WIDTH >::empty ( ) [virtual]
```

Queue API: Returns whether or not the queue is empty.

Returns

1 if the queue is empty, 0 otherwise

Note

Abstract method. Must be implemented in a subclass

Reimplemented from [cl_syoscb_queue_base](#).

13.95.2.5 `get_item()`

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
virtual cl\_syoscb\_item cl\_syoscb\_queue\_hash< HASH_DIGEST_WIDTH >::get_item (
    cl\_syoscb\_proxy\_item\_base proxy_item ) [virtual]
```

Queue API: Gets the item pointed to by the proxy item from the queue.

If the proxy item does not specify a valid item in the queue, print a UVM_INFO/DEBUG message

Parameters

<i>proxy_item</i>	A proxy item indicating which scoreboard item to delete from the queue
-------------------	--

Returns

The scoreboard item indicated by the proxy item, null if the proxy item did not point to a valid item

Note

Abstract method. Must be implemented in a subclass

Reimplemented from [cl_syoscb_queue_base](#).

13.95.2.6 `get_key_queue()`

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
cl\_syoscb\_queue\_hash::tp\_queue\_of\_keys cl\_syoscb\_queue\_hash< HASH_DIGEST_WIDTH >::get_key_↵
queue ( ) [virtual]
```

Get the list of hash values of items in the queue.

Note

If [cl_syoscb_cfg::ordered_next](#) is 0, the key queue has no inherent meaning. An empty queue is returned in this case

Definition at line 325 of file `cl_syoscb_queue_hash.svh`.

Referenced by `cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::get_item_proxy()`, `cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::next()`, and `cl_syoscb_queue_↵
iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::previous()`.

13.95.2.7 `get_size()`

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
virtual int unsigned cl\_syoscb\_queue\_hash< HASH_DIGEST_WIDTH >::get_size ( ) [virtual]
```

Queue API: Returns the current size of the queue.

Returns

Number of items currently in the queue

Note

Abstract method. Must be implemented in a subclass

Reimplemented from [cl_syoscb_queue_base](#).

13.95.2.8 `insert_item()`

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
virtual bit cl\_syoscb\_queue\_hash< HASH_DIGEST_WIDTH >::insert_item (
    string producer,
    uvm_sequence_item item,
    int unsigned idx ) [virtual]
```

Queue API: Inserts a `uvm_sequence_item` at index `idx`.

The method works in the same manner as [add_item](#), by doing the following:

1. Insert the a new item as the [add_item\(\)](#) method
2. Notify any iterators

Parameters

<i>producer</i>	The producer of the sequence item
<i>item</i>	The item that should be add to the queue
<i>idx</i>	The index at which the item should be inserted

Returns

1 if the item was successfully inserted, 0 otherwise

Note

Abstract method. Must be implemented in a subclass

Reimplemented from [cl_syoscb_queue_base](#).

13.95.3 Member Data Documentation

13.95.3.1 key_queue

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
tp_item_digest cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >::key_queue [protected]
```

List of hash values of the items in the queue.

Only used if `cl_syoscb_cfg::ordered_next` is 1.

Definition at line 39 of file `cl_syoscb_queue_hash.svh`.

Referenced by `cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::add_item()`, `cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::delete_item()`, `cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::do_flush_queue()`, `cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::get_key_queue()`, and `cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::insert_item()`.

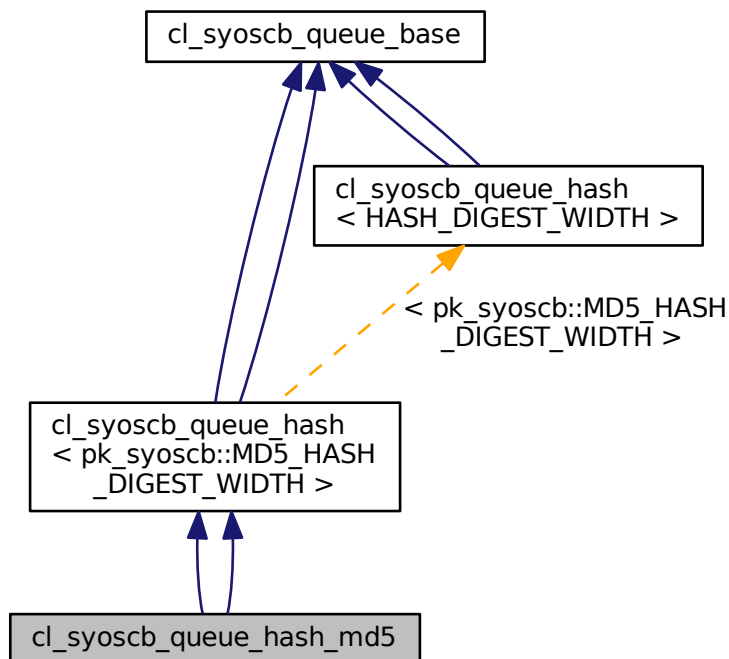
The documentation for this class was generated from the following files:

- `cl_syoscb_queue_hash.svh`
- `pk_syoscb.sv`

13.96 cl_syoscb_queue_hash_md5 Class Reference

MD5 implementation of a hash queue which optimizes the OOO compare.

Inheritance diagram for `cl_syoscb_queue_hash_md5`:



Parameters

<i>name</i>	A name to be used for the iterator. If an iterator with this name already exists, prints a UVM_DEBUG message
-------------	--

Returns

An iterator over this queue, or null if a queue with the requested name already exists

Reimplemented from [cl_syoscb_queue_base](#).

13.96.2.2 get_locator()

```
virtual cl\_syoscb\_queue\_locator\_base cl_syoscb_queue_hash_md5::get_locator ( ) [virtual]
```

Queue API: Creates a locator for this queue.

Returns

A locator over this queue

Reimplemented from [cl_syoscb_queue_base](#).

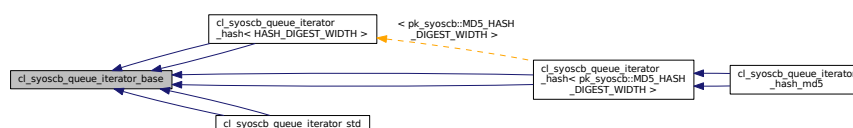
The documentation for this class was generated from the following files:

- cl_syoscb_queue_hash_md5.svh
- pk_syoscb.sv

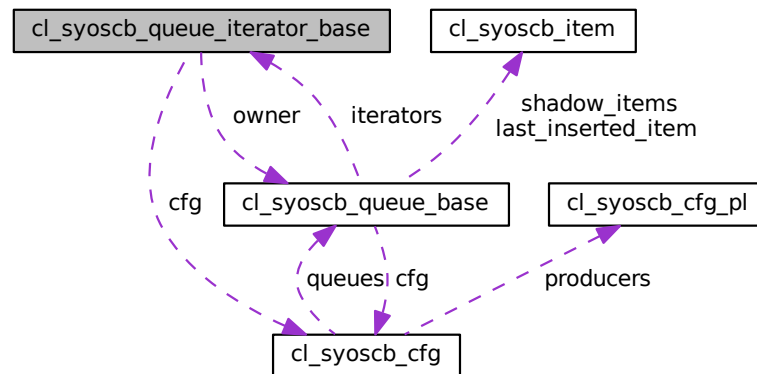
13.97 cl_syoscb_queue_iterator_base Class Reference

Queue iterator base class defining the iterator API used for iterating over queues.

Inheritance diagram for cl_syoscb_queue_iterator_base:



Collaboration diagram for `cl_syoscb_queue_iterator_base`:



Public Member Functions

- virtual `cl_syoscb_proxy_item_base next ()`
Iterator API: Moves the iterator one step forward, returning the next item in the queue.
- virtual bit `has_next ()`
Iterator API: Checks if there are more items in the queue in the forward direction
- virtual int `next_index ()`
Iterator API: Returns the index of the item which would be returned if `next()` was called
- virtual `cl_syoscb_proxy_item_base previous ()`
Iterator API: Moves the iterator one step backward, returning the previous item in the queue.
- virtual bit `has_previous ()`
Iterator API: Checks if there are more items in the queue in the backward direction
- virtual int `previous_index ()`
Iterator API: Returns the index of the item which would be returned if `previous()` was called
- virtual bit `first ()`
Iterator API: Moves the iterator to the first item in the queue.
- virtual bit `last ()`
Iterator API: Moves the iterator to the last item in the queue.
- virtual bit `set_queue (cl_syoscb_queue_base owner)`
Iterator API: Sets the queue over which this iterator is iterating.

Protected Member Functions

- virtual `cl_syoscb_queue_base get_queue ()`
Iterator API: Internal API: Returns the queue over which this iterator is iterating.
- virtual `cl_syoscb_proxy_item_base get_item_proxy ()`
Iterator API: Internal API: Returns a proxy item that can be used to access the element that was just moved past by calling `next` or `previous`.

Protected Attributes

- [cl_syoscb_queue_base](#) owner
The owner of this iterator.
- int unsigned [position](#) = 0
Current position in the queue.
- [cl_syoscb_cfg](#) cfg
Local handle to the SCB cfg.

13.97.1 Detailed Description

Queue iterator base class defining the iterator API used for iterating over queues.

The iterator API is modelled after the Java ListIterator interface <https://docs.oracle.com/javase/8/docs/api/java/ListIterator.html>. To iterate over all elements of a queue, use a while loop of the type

```
void' (iter.first());
while(iter.has_next()) begin
    cl_syoscb_proxy_item_base pib = iter.next();
    //do something
end
```

Internally, the iterator's position is always between elements. Calling [next](#) or [previous](#) will advance or reverse the iterator, returning the item that was moved past

```
items:           queue[0]   queue[1]   queue[2]   ...   queue[n-1]
cursor positions: ^         ^         ^         ^         ^         ^
```

Definition at line 17 of file `cl_syoscb_queue_iterator_base.svh`.

13.97.2 Member Function Documentation

13.97.2.1 first()

```
bit cl_syoscb_queue_iterator_base::first ( ) [virtual]
```

Iterator API: Moves the iterator to the first item in the queue.

Calling [has_previous](#) at this point will always return 1'b0

Returns

1 if successful, 0 if the queue is empty

Reimplemented in [cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_iterator_hash< pk_syoscb::MD5](#), [cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST](#), [cl_syoscb_queue_iterator_std](#), and [cl_syoscb_queue_iterator_std](#).

Definition at line 104 of file `cl_syoscb_queue_iterator_base.svh`.

Referenced by `cl_scb_test_iterator_unit_tests::check_first()`, `cl_scb_test_iterator_unit_tests::check_flush()`, `cl_scb_test_iterator_unit_tests::create_primary_iterator()`, `cl_syoscb_queue_base::dump_orphans_to_file()`, `cl_syoscb_queue_base::dump_orphans_to_stdout()`, `cl_syoscb_compare_ooo::get_count_producer()`, `cl_syoscb_compare_ooo::get_count_producer()`, `cl_syoscb_compare_ooo::primary_loop_do()`, `cl_syoscb_compare_io_2hp::primary_loop_do()`, `cl_syoscb_compare_io_2hp::primary_loop_do()`, `cl_syoscb_compare_io::secondary_loop_do()`, and `cl_syoscb_compare_io::secondary_loop_do()`.

13.97.2.2 `get_item_proxy()`

```
cl_syoscb_proxy_item_base cl_syoscb_queue_iterator_base::get_item_proxy ( ) [protected],
[virtual]
```

Iterator API: Internal API: Returns a proxy item that can be used to access the element that was just moved past by calling [next](#) or [previous](#).

Returns

A proxy item for the element that was moved past.

Reimplemented in [cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_iterator_std](#), and [cl_syoscb_queue_iterator_std](#).

Definition at line 143 of file `cl_syoscb_queue_iterator_base.svh`.

13.97.2.3 `get_queue()`

```
cl_syoscb_queue_base cl_syoscb_queue_iterator_base::get_queue ( ) [protected], [virtual]
```

Iterator API: Internal API: Returns the queue over which this iterator is iterating.

Returns

A handle to the queue. Raises a UVM_FATAL if no queue is associated with the iterator.

Definition at line 119 of file `cl_syoscb_queue_iterator_base.svh`.

References owner.

Referenced by [cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::get_item_proxy\(\)](#), [cl_syoscb_queue_iterator_std::has_next\(\)](#), [cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::has_next\(\)](#), [cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::has_previous\(\)](#), [cl_syoscb_queue_iterator_std::next\(\)](#), [cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::next\(\)](#), [cl_syoscb_queue_iterator_std::previous\(\)](#), and [cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::previous\(\)](#).

13.97.2.4 `has_next()`

```
bit cl_syoscb_queue_iterator_base::has_next ( ) [virtual]
```

Iterator API: Checks if there are more items in the queue in the forward direction

Returns

1 if there are more items in the forward direction, 0 otherwise (either empty queue or past last item)

Reimplemented in [cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_iterator_std](#), and [cl_syoscb_queue_iterator_std](#).

Definition at line 69 of file `cl_syoscb_queue_iterator_base.svh`.

Referenced by [cl_scb_test_iterator_unit_tests::check_flush\(\)](#), [cl_scb_test_iterator_unit_tests::check_next\(\)](#), [cl_scb_test_iterator_unit_tests::check_prev\(\)](#), [cl_syoscb_queue_base::dump_orphans_to_file\(\)](#), [cl_syoscb_queue_base::dump_orphans_to_stdout\(\)](#), [cl_syoscb_compare_ooo::primary_loop_do\(\)](#), [cl_syoscb_compare_iop::primary_loop_do\(\)](#), and [cl_syoscb_compare_iop::secondary_loop_do\(\)](#).

13.97.2.5 has_previous()

```
bit cl_syoscb_queue_iterator_base::has_previous ( ) [virtual]
```

Iterator API: Checks if there are more items in the queue in the backward direction

Returns

1 if there are more items in the backward direction, 0 otherwise (either empty queue or at first item)

Reimplemented in [cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_iterator_std](#), and [cl_syoscb_queue_iterator_std](#).

Definition at line 90 of file `cl_syoscb_queue_iterator_base.svh`.

Referenced by `cl_scb_test_iterator_unit_tests::check_flush()`, and `cl_scb_test_iterator_unit_tests::check_prev()`.

13.97.2.6 last()

```
bit cl_syoscb_queue_iterator_base::last ( ) [virtual]
```

Iterator API: Moves the iterator to the last item in the queue.

Calling [has_next](#) at this point will always return 1'b0.

Returns

1 if succesful, 0 if there is no first item (queue is empty)

Reimplemented in [cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_iterator_std](#), and [cl_syoscb_queue_iterator_std](#).

Definition at line 112 of file `cl_syoscb_queue_iterator_base.svh`.

Referenced by `cl_scb_test_iterator_unit_tests::check_flush()`, and `cl_scb_test_iterator_unit_tests::check_last()`.

13.97.2.7 next()

```
cl_syoscb_proxy_item_base cl_syoscb_queue_iterator_base::next ( ) [virtual]
```

Iterator API: Moves the iterator one step forward, returning the next item in the queue.

Returns

The next item if successful, raises a `uvm_error` if there is no next item

Reimplemented in [cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_iterator_std](#), and [cl_syoscb_queue_iterator_std](#).

Definition at line 62 of file `cl_syoscb_queue_iterator_base.svh`.

Referenced by `cl_scb_test_iterator_unit_tests::check_first()`, `cl_scb_test_iterator_unit_tests::check_flush()`, `cl_scb_test_iterator_unit_tests::check_prev()`, `cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::delete_item()`, `cl_syoscb_queue_base::dump_orphans_to_file()`, `cl_syoscb_queue_base::dump_orphans_to_stdout()`, `cl_syoscb_compare_ooo::get_count_producer()`, `cl_syoscb_compare_iop::get_count_producer()`, `cl_syoscb_compare_ooo::primary_loop_do()`, `cl_syoscb_compare_io::primary_loop_do()`, `cl_syoscb_compare_io_2hp::primary_loop_do()`, `cl_syoscb_compare_iop::primary_loop_do()`, `cl_syoscb_compare_io::secondary_loop_do()`, and `cl_syoscb_compare_iop::secondary_loop_do()`.

13.97.2.8 next_index()

```
int cl_syoscb_queue_iterator_base::next_index ( ) [virtual]
```

Iterator API: Returns the index of the item which would be returned if [next\(\)](#) was called

Returns

The index of the next item, or `queue.size()` if the iterator has reached the }.

Definition at line 76 of file `cl_syoscb_queue_iterator_base.svh`.

References position.

Referenced by `cl_scb_test_iterator_unit_tests::check_flush()`, `cl_scb_test_iterator_unit_tests::check_last()`, `cl_scb_test_iterator_unit_tests::check_next()`, `cl_scb_test_iterator_unit_tests::check_prev()`, `cl_syoscb_queue_base::dump_orphans_to_file()`, `cl_syoscb_queue_base::dump_orphans_to_stdout()`, `cl_syoscb_compare_ooo::primary_loop_do()`, `cl_syoscb_compare_iop::primary_loop_do()`, and `cl_syoscb_compare_iop::secondary_loop_do()`.

13.97.2.9 previous()

```
cl_syoscb_proxy_item_base cl_syoscb_queue_iterator_base::previous ( ) [virtual]
```

Iterator API: Moves the iterator one step backward, returning the previous item in the queue.

Returns

The previous item if successful, raises a `uvm_error` if there is no previous item

Reimplemented in `cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH >`, `cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >`, `cl_syoscb_queue_iterator_std`, and `cl_syoscb_queue_iterator_std`.

Definition at line 82 of file `cl_syoscb_queue_iterator_base.svh`.

13.97.2.10 previous_index()

```
int cl_syoscb_queue_iterator_base::previous_index ( ) [virtual]
```

Iterator API: Returns the index of the item which would be returned if [previous\(\)](#) was called

Returns

The index of the previous item, or -1 if the iterator is pointing to the first item of the queue

Definition at line 97 of file `cl_syoscb_queue_iterator_base.svh`.

References position.

Referenced by `cl_scb_test_iterator_unit_tests::check_first()`, `cl_scb_test_iterator_unit_tests::check_flush()`, `cl_scb_test_iterator_unit_tests::check_next()`, `cl_scb_test_iterator_unit_tests::check_prev()`, `cl_syoscb_queue_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::delete_item()`, `cl_syoscb_queue_base::dump_orphans_to_file()`, `cl_syoscb_queue_base::dump_orphans_to_stdout()`, and `cl_syoscb_compare_iop::secondary_loop_do()`.

13.97.2.11 `set_queue()`

```
bit cl_syoscb_queue_iterator_base::set_queue (
    cl_syoscb_queue_base owner ) [virtual]
```

Iterator API: Sets the queue over which this iterator is iterating.

If a queue has already been associated with this iterator, or the queue type does not match the iterator type, generates a UVM_ERROR message with id ITER_ERROR.

Returns

1 if successful, raises a UVM_ERROR otherwise (a queue is already associated with this iterator, or wrong queue type)

Reimplemented in [cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_iterator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_iterator_std](#), and [cl_syoscb_queue_iterator_std](#).

Definition at line 134 of file `cl_syoscb_queue_iterator_base.svh`.

Referenced by `cl_scb_test_iterator_unit_tests::check_set_queue()`.

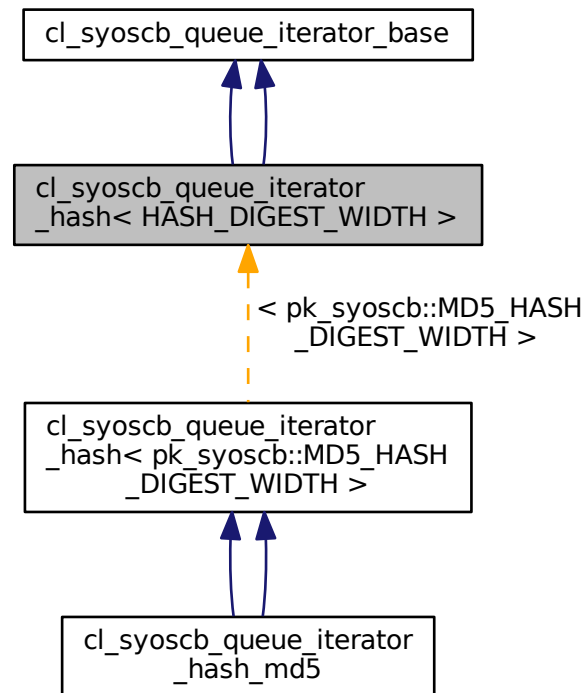
The documentation for this class was generated from the following files:

- `cl_syoscb_queue_iterator_base.svh`
- `pk_syoscb.sv`

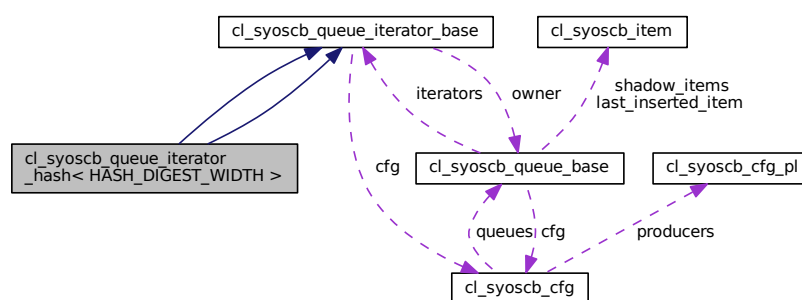
13.98 `cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH >` Class Template Reference

Queue iterator class defining the iterator API used for iterating hash queues.

Inheritance diagram for `cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH >`:



Collaboration diagram for `cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH >`:



Public Member Functions

- virtual `cl_syoscb_proxy_item_base` `next` ()
Iterator API: See `cl_syoscb_queue_iterator_base::next` for details
- virtual bit `has_next` ()
Iterator API: See `cl_syoscb_queue_iterator_base::has_next` for details

- virtual `cl_syoscb_proxy_item_base previous ()`
Iterator API: See `cl_syoscb_queue_iterator_base::previous` for details
- virtual bit `has_previous ()`
Iterator API: See `cl_syoscb_queue_iterator_base::has_previous` for details
- virtual bit `first ()`
Iterator API: See `cl_syoscb_queue_iterator_base::first` for details
- virtual bit `last ()`
Iterator API: See `cl_syoscb_queue_iterator_base::last` for details
- virtual bit `set_queue (cl_syoscb_queue_base owner)`
Iterator API: See `cl_syoscb_queue_iterator_base::set_queue` for details
- virtual `cl_syoscb_proxy_item_base next ()`
Iterator API: Moves the iterator one step forward, returning the next item in the queue.
- virtual bit `has_next ()`
Iterator API: Checks if there are more items in the queue in the forward direction
- virtual `cl_syoscb_proxy_item_base previous ()`
Iterator API: Moves the iterator one step backward, returning the previous item in the queue.
- virtual bit `has_previous ()`
Iterator API: Checks if there are more items in the queue in the backward direction
- virtual bit `first ()`
Iterator API: Moves the iterator to the first item in the queue.
- virtual bit `last ()`
Iterator API: Moves the iterator to the last item in the queue.
- virtual bit `set_queue (cl_syoscb_queue_base owner)`
Iterator API: Sets the queue over which this iterator is iterating.

Protected Member Functions

- virtual `cl_syoscb_proxy_item_base get_item_proxy ()`
Iterator API: See `cl_syoscb_queue_iterator_base::get_item_proxy` for details
- virtual `cl_syoscb_proxy_item_base get_item_proxy ()`
Iterator API: Internal API: Returns a proxy item that can be used to access the element that was just moved past by calling `next` or `previous`.

Protected Attributes

- int unsigned `hash_index = 0`
Field indicating which `cl_syoscb_hash_item` index we're currently looking at.

Private Attributes

- `cl_syoscb_hash_base< HASH_DIGEST_WIDTH >::tp_hash_digest digest`
Holds the value of the most recently accessed hash digest.

13.98.1 Detailed Description

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
class cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH >
```

Queue iterator class defining the iterator API used for iterating hash queues.

Parameters

<code>HASH_DIGEST_WIDTH</code>	Number of bits used in the hash digest for the chosen hash algorithm
--------------------------------	--

Definition at line 3 of file `cl_syoscb_queue_iterator_hash.svh`.

13.98.2 Member Function Documentation

13.98.2.1 `first()`

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
virtual bit cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH >::first ( ) [virtual]
```

Iterator API: Moves the iterator to the first item in the queue.

Calling `has_previous` at this point will always return 1'b0

Returns

1 if successful, 0 if the queue is empty

Reimplemented from `cl_syoscb_queue_iterator_base`.

13.98.2.2 `get_item_proxy()`

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
virtual cl_syoscb_proxy_item_base cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH >::get_↵
item_proxy ( ) [protected], [virtual]
```

Iterator API: Internal API: Returns a proxy item that can be used to access the element that was just moved past by calling `next` or `previous`.

Returns

A proxy item for the element that was moved past.

Reimplemented from `cl_syoscb_queue_iterator_base`.

13.98.2.3 `has_next()`

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
virtual bit cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH >::has_next ( ) [virtual]
```

Iterator API: Checks if there are more items in the queue in the forward direction

Returns

1 if there are more items in the forward direction, 0 otherwise (either empty queue or past last item)

Reimplemented from [cl_syoscb_queue_iterator_base](#).

13.98.2.4 `has_previous()`

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
virtual bit cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH >::has_previous ( ) [virtual]
```

Iterator API: Checks if there are more items in the queue in the backward direction

Returns

1 if there are more items in the backward direction, 0 otherwise (either empty queue or at first item)

Reimplemented from [cl_syoscb_queue_iterator_base](#).

13.98.2.5 `last()`

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
virtual bit cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH >::last ( ) [virtual]
```

Iterator API: Moves the iterator to the last item in the queue.

Calling [has_next](#) at this point will always return 1'b0.

Returns

1 if succesful, 0 if there is no first item (queue is empty)

Reimplemented from [cl_syoscb_queue_iterator_base](#).

13.98.2.6 next()

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
virtual cl\_syoscb\_proxy\_item\_base cl\_syoscb\_queue\_iterator\_hash< HASH_DIGEST_WIDTH >::next ( )
[virtual]
```

Iterator API: Moves the iterator one step forward, returning the next item in the queue.

Returns

The next item if successful, raises a `uvm_error` if there is no next item

Reimplemented from [cl_syoscb_queue_iterator_base](#).

13.98.2.7 previous()

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
virtual cl\_syoscb\_proxy\_item\_base cl\_syoscb\_queue\_iterator\_hash< HASH_DIGEST_WIDTH >::previous
( ) [virtual]
```

Iterator API: Moves the iterator one step backward, returning the previous item in the queue.

Returns

The previous item if successful, raises a `uvm_error` if there is no previous item

Reimplemented from [cl_syoscb_queue_iterator_base](#).

13.98.2.8 set_queue()

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
virtual bit cl\_syoscb\_queue\_iterator\_hash< HASH_DIGEST_WIDTH >::set_queue (
    cl\_syoscb\_queue\_base owner ) [virtual]
```

Iterator API: Sets the queue over which this iterator is iterating.

If a queue has already been associated with this iterator, or the queue type does not match the iterator type, generates a `UVM_ERROR` message with id `ITER_ERROR`.

Returns

1 if successful, raises a `UVM_ERROR` otherwise (a queue is already associated with this iterator, or wrong queue type)

Reimplemented from [cl_syoscb_queue_iterator_base](#).

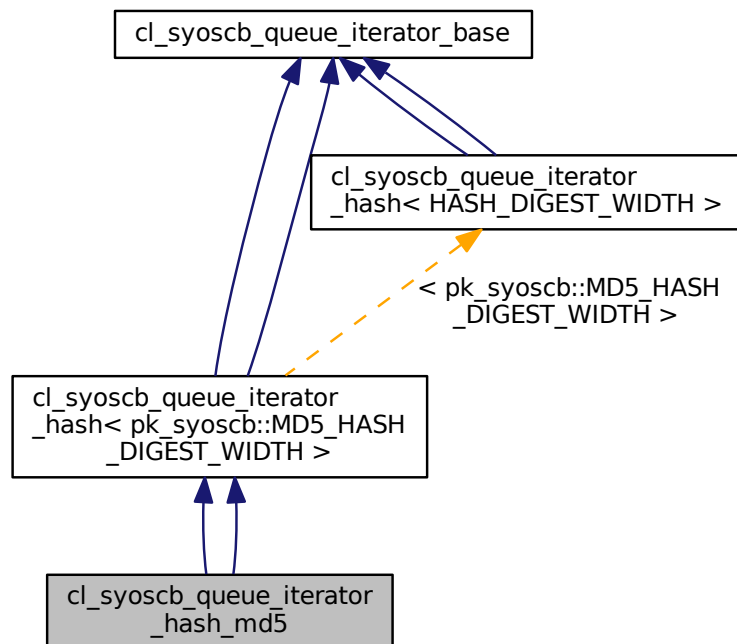
The documentation for this class was generated from the following files:

- `cl_syoscb_queue_iterator_hash.svh`
- `pk_syoscb.sv`

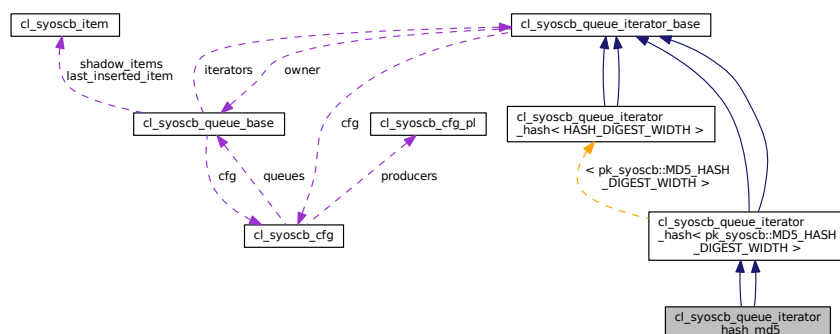
13.99 cl_syoscb_queue_iterator_hash_md5 Class Reference

Queue iterator class defining the iterator API used for iterating md5 hash queues.

Inheritance diagram for cl_syoscb_queue_iterator_hash_md5:



Collaboration diagram for cl_syoscb_queue_iterator_hash_md5:



Additional Inherited Members

13.99.1 Detailed Description

Queue iterator class defining the iterator API used for iterating md5 hash queues.

Definition at line 2 of file `cl_syoscb_queue_iterator_hash_md5.svh`.

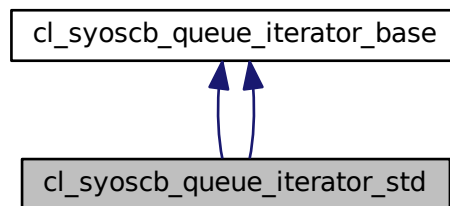
The documentation for this class was generated from the following files:

- `cl_syoscb_queue_iterator_hash_md5.svh`
- `pk_syoscb.sv`

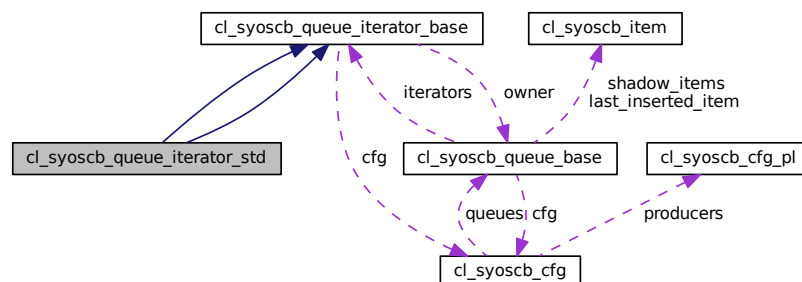
13.100 `cl_syoscb_queue_iterator_std` Class Reference

Queue iterator class for iterating over std queues.

Inheritance diagram for `cl_syoscb_queue_iterator_std`:



Collaboration diagram for `cl_syoscb_queue_iterator_std`:



Public Member Functions

- virtual [cl_syoscb_proxy_item_base next](#) ()
Iterator API: See [cl_syoscb_queue_iterator_base::next](#) for details
- virtual bit [has_next](#) ()
Iterator API: See [cl_syoscb_queue_iterator_base::has_next](#) for details
- virtual [cl_syoscb_proxy_item_base previous](#) ()
Iterator API: See [cl_syoscb_queue_iterator_base::previous](#) for details
- virtual bit [has_previous](#) ()
Iterator API: See [cl_syoscb_queue_iterator_base::has_previous](#) for details
- virtual bit [first](#) ()
Iterator API: See [cl_syoscb_queue_iterator_base::first](#) for details
- virtual bit [last](#) ()
Iterator API: See [cl_syoscb_queue_iterator_base::last](#) for details
- virtual bit [set_queue](#) ([cl_syoscb_queue_base owner](#))
Iterator API: See [cl_syoscb_queue_iterator_base::set_queue](#) for details
- virtual [cl_syoscb_proxy_item_base next](#) ()
Iterator API: Moves the iterator one step forward, returning the next item in the queue.
- virtual bit [has_next](#) ()
Iterator API: Checks if there are more items in the queue in the forward direction
- virtual [cl_syoscb_proxy_item_base previous](#) ()
Iterator API: Moves the iterator one step backward, returning the previous item in the queue.
- virtual bit [has_previous](#) ()
Iterator API: Checks if there are more items in the queue in the backward direction
- virtual bit [first](#) ()
Iterator API: Moves the iterator to the first item in the queue.
- virtual bit [last](#) ()
Iterator API: Moves the iterator to the last item in the queue.
- virtual bit [set_queue](#) ([cl_syoscb_queue_base owner](#))
Iterator API: Sets the queue over which this iterator is iterating.

Protected Member Functions

- virtual [cl_syoscb_proxy_item_base get_item_proxy](#) ()
Iterator API: See [cl_syoscb_queue_iterator_base::get_item_proxy](#) for details
- virtual [cl_syoscb_proxy_item_base get_item_proxy](#) ()
Iterator API: Internal API: Returns a proxy item that can be used to access the element that was just moved past by calling [next](#) or [previous](#).

Additional Inherited Members

13.100.1 Detailed Description

Queue iterator class for iterating over std queues.

Definition at line 2 of file `cl_syoscb_queue_iterator_std.svh`.

13.100.2 Member Function Documentation

13.100.2.1 first()

```
virtual bit cl_syoscb_queue_iterator_std::first ( ) [virtual]
```

Iterator API: Moves the iterator to the first item in the queue.

Calling [has_previous](#) at this point will always return 1'b0

Returns

1 if successful, 0 if the queue is empty

Reimplemented from [cl_syoscb_queue_iterator_base](#).

13.100.2.2 get_item_proxy()

```
virtual cl\_syoscb\_proxy\_item\_base cl_syoscb_queue_iterator_std::get_item_proxy ( ) [protected],  
[virtual]
```

Iterator API: Internal API: Returns a proxy item that can be used to access the element that was just moved past by calling [next](#) or [previous](#).

Returns

A proxy item for the element that was moved past.

Reimplemented from [cl_syoscb_queue_iterator_base](#).

13.100.2.3 has_next()

```
virtual bit cl_syoscb_queue_iterator_std::has_next ( ) [virtual]
```

Iterator API: Checks if there are more items in the queue in the forward direction

Returns

1 if there are more items in the forward direction, 0 otherwise (either empty queue or past last item)

Reimplemented from [cl_syoscb_queue_iterator_base](#).

13.100.2.4 has_previous()

```
virtual bit cl_syoscb_queue_iterator_std::has_previous ( ) [virtual]
```

Iterator API: Checks if there are more items in the queue in the backward direction

Returns

1 if there are more items in the backward direction, 0 otherwise (either empty queue or at first item)

Reimplemented from [cl_syoscb_queue_iterator_base](#).

13.100.2.5 last()

```
virtual bit cl_syoscb_queue_iterator_std::last ( ) [virtual]
```

Iterator API: Moves the iterator to the last item in the queue.

Calling [has_next](#) at this point will always return 1'b0.

Returns

1 if succesful, 0 if there is no first item (queue is empty)

Reimplemented from [cl_syoscb_queue_iterator_base](#).

13.100.2.6 next()

```
virtual cl\_syoscb\_proxy\_item\_base cl_syoscb_queue_iterator_std::next ( ) [virtual]
```

Iterator API: Moves the iterator one step forward, returning the next item in the queue.

Returns

The next item if successful, raises a uvm_error if there is no next item

Reimplemented from [cl_syoscb_queue_iterator_base](#).

13.100.2.7 previous()

```
virtual cl_syoscb_proxy_item_base cl_syoscb_queue_iterator_std::previous ( ) [virtual]
```

Iterator API: Moves the iterator one step backward, returning the previous item in the queue.

Returns

The previous item if successful, raises a uvm_error if there is no previous item

Reimplemented from [cl_syoscb_queue_iterator_base](#).

13.100.2.8 set_queue()

```
virtual bit cl_syoscb_queue_iterator_std::set_queue (
    cl_syoscb_queue_base owner ) [virtual]
```

Iterator API: Sets the queue over which this iterator is iterating.

If a queue has already been associated with this iterator, or the queue type does not match the iterator type, generates a UVM_ERROR message with id ITER_ERROR.

Returns

1 if successful, raises a UVM_ERROR otherwise (a queue is already associated with this iterator, or wrong queue type)

Reimplemented from [cl_syoscb_queue_iterator_base](#).

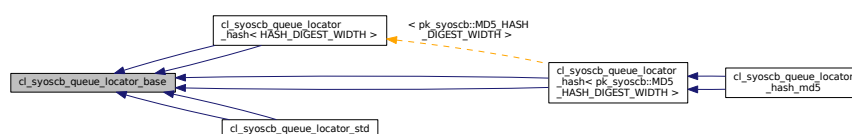
The documentation for this class was generated from the following files:

- cl_syoscb_queue_iterator_std.svh
- pk_syoscb.sv

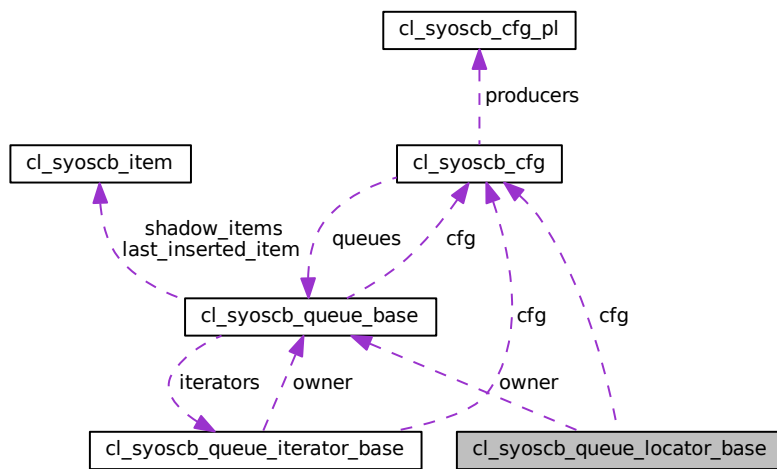
13.101 cl_syoscb_queue_locator_base Class Reference

Locator base class defining the locator API used for searching in queues.

Inheritance diagram for cl_syoscb_queue_locator_base:



Collaboration diagram for cl_syoscb_queue_locator_base:



Public Member Functions

- virtual [cl_syoscb_proxy_item_base search](#) ([cl_syoscb_proxy_item_base](#) proxy_item)
Locator API: Returns the item of the underlying queue which matches the given proxy item
- virtual bit [set_queue](#) ([cl_syoscb_queue_base](#) owner)
Locator API: Sets the queue that this locator is associated with
- virtual [cl_syoscb_queue_base get_queue](#) ()
Locator API: Returns the queue that this locator is associated with

Protected Attributes

- [cl_syoscb_queue_base](#) owner
The queue owning this locator.
- [cl_syoscb_cfg](#) cfg
Local handle to the SCB cfg.

13.101.1 Detailed Description

Locator base class defining the locator API used for searching in queues.

Locators are primarily used with the OOO compare and hash queues, as this allows us to efficiently find an item with a matching digest

Definition at line 4 of file cl_syoscb_queue_locator_base.svh.

13.101.2 Member Function Documentation

13.101.2.1 `search()`

```
cl_syoscb_proxy_item_base cl_syoscb_queue_locator_base::search (
    cl_syoscb_proxy_item_base proxy_item ) [virtual]
```

Locator API: Returns the item of the underlying queue which matches the given proxy item

Parameters

<i>proxy_item</i>	A proxy item indicating what to search for in this queue
-------------------	--

Returns

A proxy item pointing to the matching item in this queue, or null if no match is found

Reimplemented in [cl_syoscb_queue_locator_std](#), [cl_syoscb_queue_locator_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_locator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_locator_hash< HASH_DIGEST_WIDTH >](#), [cl_syoscb_queue_locator_std](#), and [cl_syoscb_queue_locator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >](#).

Definition at line 34 of file `cl_syoscb_queue_locator_base.svh`.

Referenced by `cl_syoscb_compare_ooo::secondary_loop_do()`.

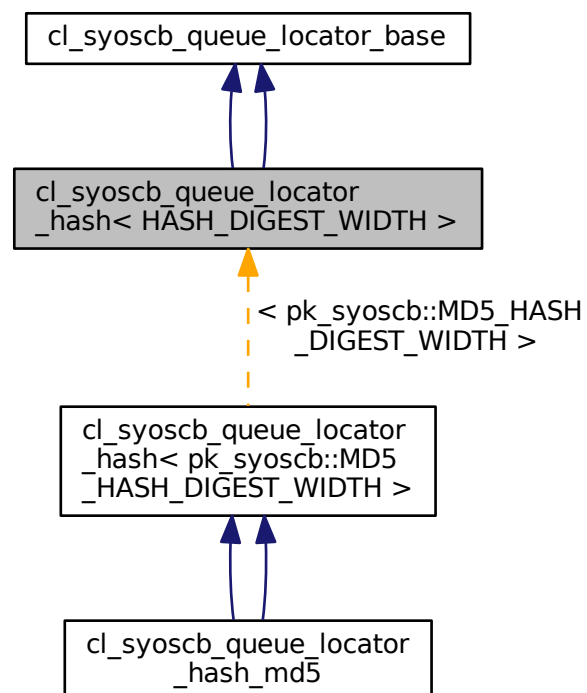
The documentation for this class was generated from the following files:

- `cl_syoscb_queue_locator_base.svh`
- `pk_syoscb.sv`

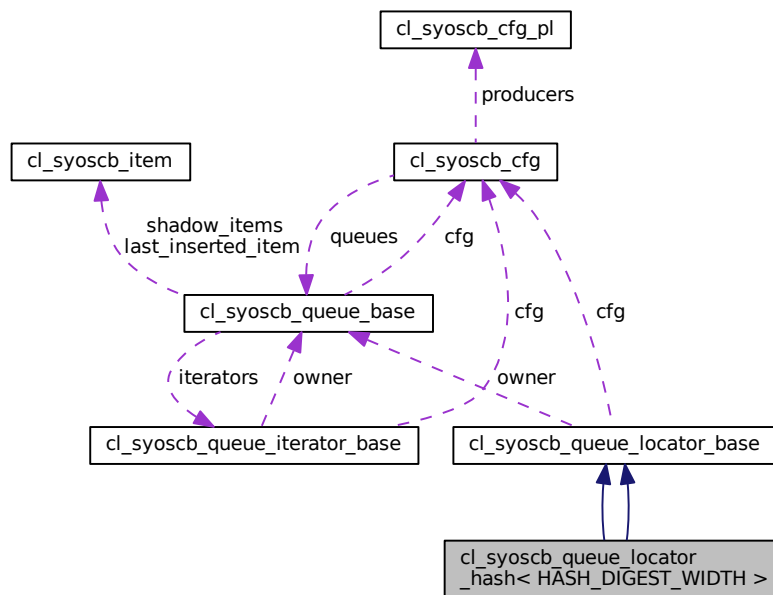
13.102 `cl_syoscb_queue_locator_hash< HASH_DIGEST_WIDTH >` Class Template Reference

Locator class for searching over generic hash queues.

Inheritance diagram for cl_syoscb_queue_locator_hash< HASH_DIGEST_WIDTH >:



Collaboration diagram for `cl_syoscb_queue_locator_hash< HASH_DIGEST_WIDTH >`:



Public Member Functions

- virtual `cl_syoscb_proxy_item_base` `search` (`cl_syoscb_proxy_item_base` proxy_item)
- Locator API:** See `cl_syoscb_queue_locator_base::search` for details
- virtual bit `validate_match` (`cl_syoscb_proxy_item_base` primary_proxy, `cl_syoscb_proxy_item_base` secondary_proxy)

Validates that a sequence item found in a secondary hash queue matches the sequence item being searched for.
- virtual void `validate_no_match` (`cl_syoscb_proxy_item_base` primary_proxy)

Validates that no sequence item in this secondary hash queue matches the primary sequence item being searched for.
- virtual `cl_syoscb_proxy_item_base` `search` (`cl_syoscb_proxy_item_base` proxy_item)

Locator API: Returns the item of the underlying queue which matches the given proxy item

Additional Inherited Members

13.102.1 Detailed Description

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
class cl_syoscb_queue_locator_hash< HASH_DIGEST_WIDTH >
```

Locator class for searching over generic hash queues.

Definition at line 2 of file `cl_syoscb_queue_locator_hash.svh`.

13.102.2 Member Function Documentation

13.102.2.1 search()

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
virtual cl\_syoscb\_proxy\_item\_base cl\_syoscb\_queue\_locator\_hash< HASH_DIGEST_WIDTH >::search (
    cl\_syoscb\_proxy\_item\_base proxy_item ) [virtual]
```

Locator API: Returns the item of the underlying queue which matches the given proxy item

Parameters

<i>proxy_item</i>	A proxy item indicating what to search for in this queue
-------------------	--

Returns

A proxy item pointing to the matching item in this queue, or null if no match is found

Reimplemented from [cl_syoscb_queue_locator_base](#).

13.102.2.2 validate_match()

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
bit cl\_syoscb\_queue\_locator\_hash< HASH_DIGEST_WIDTH >::validate_match (
    cl\_syoscb\_proxy\_item\_base primary_proxy,
    cl\_syoscb\_proxy\_item\_base secondary_proxy ) [virtual]
```

Validates that a sequence item found in a secondary hash queue matches the sequence item being searched for.

Raises a UVM_ERROR if the items do not match

Parameters

<i>primary_proxy</i>	The proxy item from the primary queue
<i>secondary_proxy</i>	The proxy item found in the secondary queue

Returns

1 is the two items match represented by proxy items match, 0 otherwise

Definition at line 101 of file [cl_syoscb_queue_locator_hash.svh](#).

Referenced by [cl_syoscb_queue_locator_hash](#)< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::search().

13.102.2.3 validate_no_match()

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
void cl_syoscb_queue_locator_hash< HASH_DIGEST_WIDTH >::validate_no_match (
    cl_syoscb_proxy_item_base primary_proxy ) [virtual]
```

Validates that no sequence item in this secondary hash queue matches the primary sequence item being searched for.

Raises a UVM_ERROR if a match is actually found

Parameters

<i>primary_proxy</i>	The proxy item from the primary queue
----------------------	---------------------------------------

Definition at line 159 of file cl_syoscb_queue_locator_hash.svh.

Referenced by cl_syoscb_queue_locator_hash< pk_syoscb::MD5_HASH_DIGEST_WIDTH >::search().

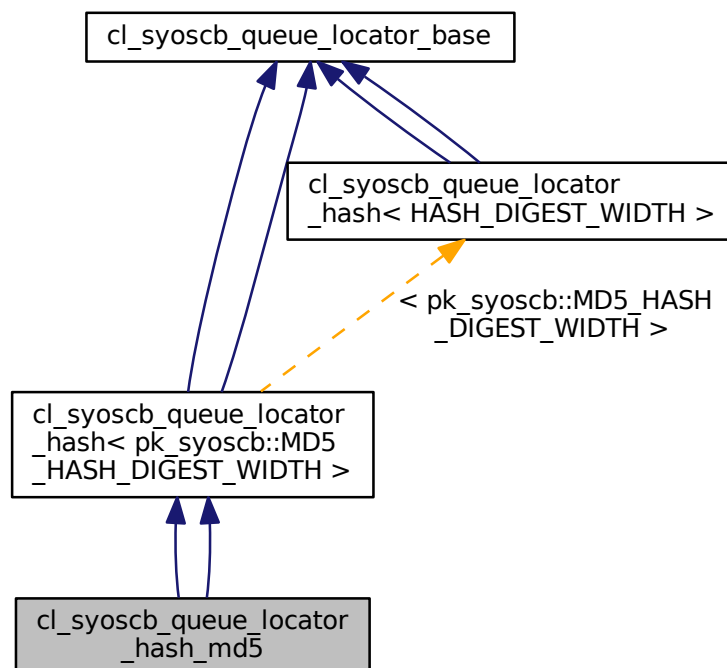
The documentation for this class was generated from the following files:

- cl_syoscb_queue_locator_hash.svh
- pk_syoscb.sv

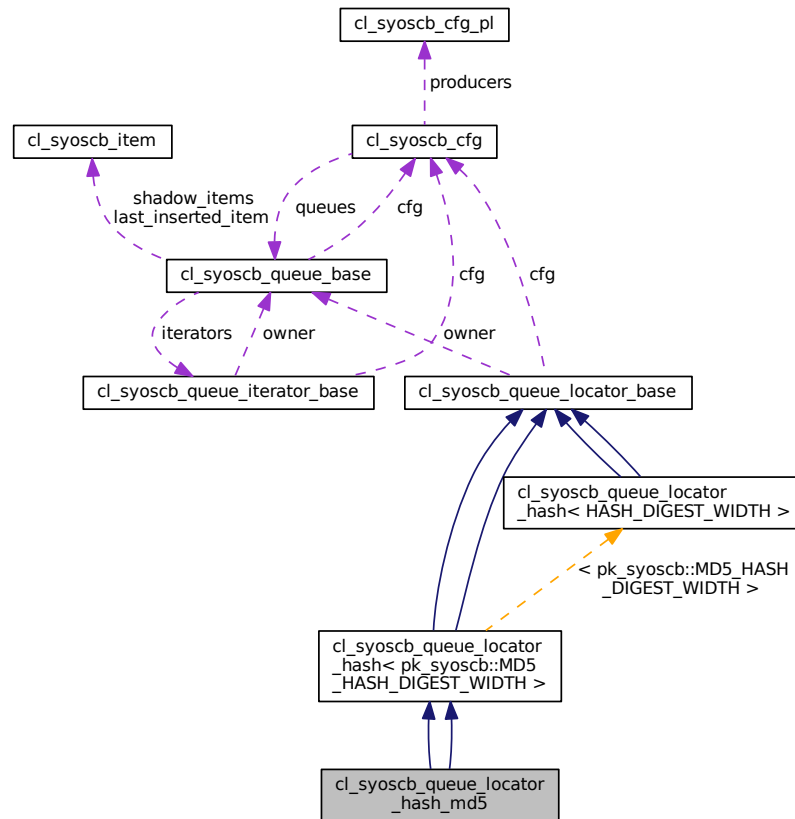
13.103 cl_syoscb_queue_locator_hash_md5 Class Reference

Locator class for searching over hash queues using md5 as the hash algorithm.

Inheritance diagram for cl_syoscb_queue_locator_hash_md5:



Collaboration diagram for cl_syoscb_queue_locator_hash_md5:



Additional Inherited Members

13.103.1 Detailed Description

Locator class for searching over hash queues using md5 as the hash algorithm.

Definition at line 2 of file `cl_syoscb_queue_locator_hash_md5.svh`.

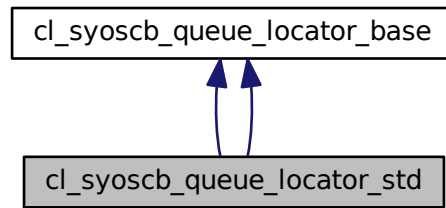
The documentation for this class was generated from the following files:

- `cl_syoscb_queue_locator_hash_md5.svh`
- `pk_syoscb.sv`

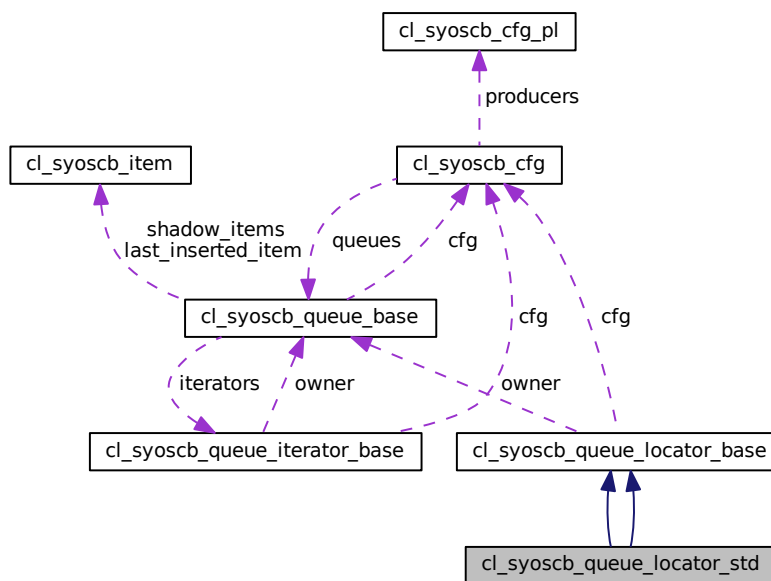
13.104 cl_syoscb_queue_locator_std Class Reference

Locator class for searching over std queues.

Inheritance diagram for `cl_syoscb_queue_locator_std`:



Collaboration diagram for `cl_syoscb_queue_locator_std`:



Public Member Functions

- virtual [cl_syoscb_proxy_item_base](#) `search` ([cl_syoscb_proxy_item_base](#) proxy_item)
Locator API: See [cl_syoscb_queue_locator_base::search](#) for details
- virtual bit [compare_items](#) ([cl_syoscb_item](#) primary_item, [cl_syoscb_item](#) sec_item, [uvm_comparer](#) comparer)
Compare two scoreboard items and check if they're equal.
- virtual [cl_syoscb_proxy_item_base](#) `search` ([cl_syoscb_proxy_item_base](#) proxy_item)
Locator API: Returns the item of the underlying queue which matches the given proxy item

Additional Inherited Members

13.104.1 Detailed Description

Locator class for searching over std queues.

Definition at line 2 of file cl_syoscb_queue_locator_std.svh.

13.104.2 Member Function Documentation

13.104.2.1 compare_items()

```
bit cl_syoscb_queue_locator_std::compare_items (
    cl_syoscb_item primary_item,
    cl_syoscb_item sec_item,
    uvm_comparer comparer ) [virtual]
```

Compare two scoreboard items and check if they're equal.

Used as parameter to queue.find_first_index when searching std-queues

Parameters

<i>primary_item</i>	The item from the primary queue
<i>sec_item</i>	The item from the secondary queue
<i>comparer</i>	The comparer to use when comparing the two items

Returns

1 if the two items are equal, 0 otherwise

Definition at line 79 of file cl_syoscb_queue_locator_std.svh.

Referenced by search().

13.104.2.2 search()

```
virtual cl_syoscb_proxy_item_base cl_syoscb_queue_locator_std::search (
    cl_syoscb_proxy_item_base proxy_item ) [virtual]
```

Locator API: Returns the item of the underlying queue which matches the given proxy item

Parameters

<i>proxy_item</i>	A proxy item indicating what to search for in this queue
-------------------	--

Returns

A proxy item pointing to the matching item in this queue, or null if no match is found

Reimplemented from [cl_syoscb_queue_locator_base](#).

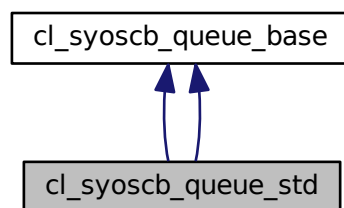
The documentation for this class was generated from the following files:

- cl_syoscb_queue_locator_std.svh
- pk_syoscb.sv

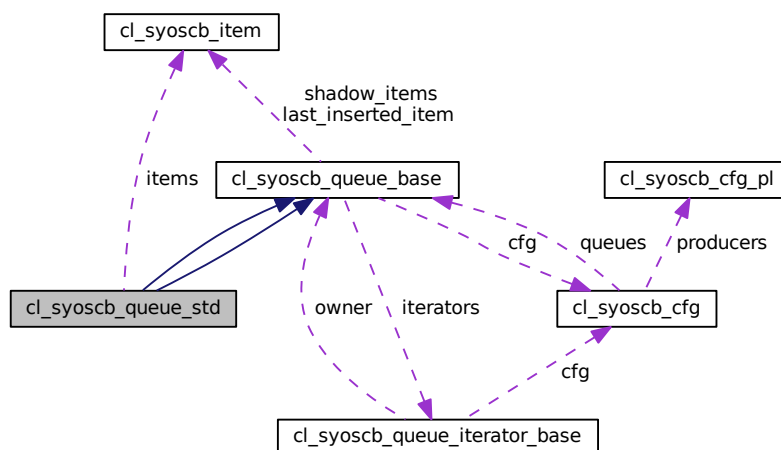
13.105 cl_syoscb_queue_std Class Reference

Standard implementation of a queue.

Inheritance diagram for cl_syoscb_queue_std:



Collaboration diagram for cl_syoscb_queue_std:



Public Member Functions

- virtual bit [add_item](#) (string producer, uvm_sequence_item item)
Queue API: See [cl_syoscb_queue_base::add_item](#) for more details
- virtual bit [delete_item](#) (cl_syoscb_proxy_item_base proxy_item)
Queue API: See [cl_syoscb_queue_base::delete_item](#) for more details
- virtual [cl_syoscb_item](#) [get_item](#) (cl_syoscb_proxy_item_base proxy_item)
Queue API: See [cl_syoscb_queue_base::get_item](#) for more details
- virtual int unsigned [get_size](#) ()
Queue API: See [cl_syoscb_queue_base::get_size](#) for more details
- virtual bit [empty](#) ()
Queue API: See [cl_syoscb_queue_base::empty](#) for more details
- virtual bit [insert_item](#) (string producer, uvm_sequence_item item, int unsigned idx)
Queue API: See [cl_syoscb_queue_base::insert_item](#) for more details
- virtual [cl_syoscb_queue_iterator_base](#) [create_iterator](#) (string name="")
Queue API: See [cl_syoscb_queue_base::create_iterator](#) for more details
- virtual bit [delete_iterator](#) (cl_syoscb_queue_iterator_base iterator)
Queue API: See [cl_syoscb_queue_base::delete_iterator](#) for more details
- virtual [cl_syoscb_queue_locator_base](#) [get_locator](#) ()
Queue API: See [cl_syoscb_queue_base::get_locator](#) for more details
- virtual bit [add_item](#) (string producer, uvm_sequence_item item)
Queue API: Adds a *uvm_sequence_item* to this queue.
- virtual bit [delete_item](#) (cl_syoscb_proxy_item_base proxy_item)
Queue API: Deletes the item indicated by the proxy item from the queue.
- virtual [cl_syoscb_item](#) [get_item](#) (cl_syoscb_proxy_item_base proxy_item)
Queue API: Gets the item pointed to by the proxy item from the queue.
- virtual int unsigned [get_size](#) ()
Queue API: Returns the current size of the queue.
- virtual bit [empty](#) ()
Queue API: Returns whether or not the queue is empty.
- virtual bit [insert_item](#) (string producer, uvm_sequence_item item, int unsigned idx)
Queue API: Inserts a *uvm_sequence_item* at index *idx*.
- virtual [cl_syoscb_queue_iterator_base](#) [create_iterator](#) (string name="")
Queue API: Creates an iterator for this queue.
- virtual bit [delete_iterator](#) (cl_syoscb_queue_iterator_base iterator)
Queue API: Deletes an iterator from this queue.
- virtual [cl_syoscb_queue_locator_base](#) [get_locator](#) ()
Queue API: Creates a locator for this queue.

Protected Member Functions

- virtual void [do_flush_queue](#) ()
See [cl_syoscb_queue_base::do_flush_queue](#) for more details.
- virtual void [do_flush_queue](#) ()
Performs the actual element deletion from the queue when called by [flush_queue](#).

Private Attributes

- [cl_syoscb_item](#) items [\$]
Simple queue implementation with a SV queue.

Additional Inherited Members

13.105.1 Detailed Description

Standard implementation of a queue.

Uses a normal SystemVerilog queue as implementation. The class implements the queue API as defined by the queue base class.

Definition at line 4 of file `cl_syoscb_queue_std.svh`.

13.105.2 Member Function Documentation

13.105.2.1 `add_item()`

```
virtual bit cl_syoscb_queue_std::add_item (
    string producer,
    uvm_sequence_item item ) [virtual]
```

Queue API: Adds a `uvm_sequence_item` to this queue.

The basic job of the `add_item` method is:

1. Create the new `cl_syoscb_item` and give it a unique name
2. Set the producer and other metadata of the scoreboard item
3. Wrap the `uvm_sequence_item` inside the scoreboard item
4. Insert the item into the queue and shadow queue
5. Update the producer counter and insert counter

Parameters

<i>producer</i>	The producer of the sequence item
<i>item</i>	The item that should be add to the queue

Returns

1 if the item was successfully added, 0 otherwise

Note

Abstract method. Must be implemented in a subclass

Reimplemented from `cl_syoscb_queue_base`.

13.105.2.2 create_iterator()

```
virtual cl\_syoscb\_queue\_iterator\_base cl_syoscb_queue_std::create_iterator (
    string name = "" ) [virtual]
```

Queue API: Creates an iterator for this queue.

Iterators are by default named "[name]_iter[X]", where [name] is the name of the queue, and [X] is the number of iterators that have previously been created for this queue

Parameters

<i>name</i>	A name to be used for the iterator. If an iterator with this name already exists, prints a UVM_DEBUG message
-------------	--

Returns

An iterator over this queue, or null if a queue with the requested name already exists

Reimplemented from [cl_syoscb_queue_base](#).

13.105.2.3 delete_item()

```
virtual bit cl_syoscb_queue_std::delete_item (
    cl\_syoscb\_proxy\_item\_base proxy_item ) [virtual]
```

Queue API: Deletes the item indicated by the proxy item from the queue.

The basic job of the delete_item method is:

1. Delete the element
2. Notify any iterators, moving them as necessary
3. Update the producer counter for the deleted item's producer

Parameters

<i>proxy_item</i>	A proxy item indicating which scoreboard item to delete from the queue
-------------------	--

Returns

if the item was successfully deleted, 0 otherwise

Note

Abstract method. Must be implemented in a subclass

Reimplemented from [cl_syoscb_queue_base](#).

13.105.2.4 delete_iterator()

```
virtual bit cl_syoscb_queue_std::delete_iterator (
    cl_syoscb_queue_iterator_base iterator ) [virtual]
```

Queue API: Deletes an iterator from this queue.

Parameters

<i>iterator</i>	The iterator to delete
-----------------	------------------------

Returns

1 if the iterator was successfully deleted, 0 otherwise

Reimplemented from [cl_syoscb_queue_base](#).

13.105.2.5 empty()

```
virtual bit cl_syoscb_queue_std::empty ( ) [virtual]
```

Queue API: Returns whether or not the queue is empty.

Returns

1 if the queue is empty, 0 otherwise

Note

Abstract method. Must be implemented in a subclass

Reimplemented from [cl_syoscb_queue_base](#).

13.105.2.6 get_item()

```
virtual cl_syoscb_item cl_syoscb_queue_std::get_item (
    cl_syoscb_proxy_item_base proxy_item ) [virtual]
```

Queue API: Gets the item pointed to by the proxy item from the queue.

If the proxy item does not specify a valid item in the queue, print a UVM_INFO/DEBUG message

Parameters

<i>proxy_item</i>	A proxy item indicating which scoreboard item to delete from the queue
-------------------	--

Returns

The scoreboard item indicated by the proxy item, null if the proxy item did not point to a valid item

Note

Abstract method. Must be implemented in a subclass

Reimplemented from [cl_syoscb_queue_base](#).

13.105.2.7 get_locator()

```
virtual cl\_syoscb\_queue\_locator\_base cl_syoscb_queue_std::get_locator ( ) [virtual]
```

Queue API: Creates a locator for this queue.

Returns

A locator over this queue

Reimplemented from [cl_syoscb_queue_base](#).

13.105.2.8 get_size()

```
virtual int unsigned cl_syoscb_queue_std::get_size ( ) [virtual]
```

Queue API: Returns the current size of the queue.

Returns

Number of items currently in the queue

Note

Abstract method. Must be implemented in a subclass

Reimplemented from [cl_syoscb_queue_base](#).

13.105.2.9 insert_item()

```
virtual bit cl_syoscb_queue_std::insert_item (
    string producer,
    uvm_sequence_item item,
    int unsigned idx ) [virtual]
```

Queue API: Inserts a `uvm_sequence_item` at index `idx`.

The method works in the same manner as [add_item](#), by doing the following:

1. Insert the a new item as the [add_item\(\)](#) method
2. Notify any iterators

Parameters

<i>producer</i>	The producer of the sequence item
<i>item</i>	The item that should be add to the queue
<i>idx</i>	The index at which the item should be inserted

Returns

1 if the item was successfully inserted, 0 otherwise

Note

Abstract method. Must be implemented in a subclass

Reimplemented from [cl_syoscb_queue_base](#).

The documentation for this class was generated from the following files:

- [cl_syoscb_queue_std.svh](#)
- [pk_syoscb.sv](#)

13.106 cl_syoscb_string_library Class Reference

A utility class holding a number of static methods for performing string manipulation.

Inherits [uvm_object](#), and [uvm_object](#).

Static Public Member Functions

- static string [pad_str](#) (string str, int unsigned max_length, string expand=" ", bit side=0b0)
Pads the input string with another string until it reaches a given length.
- static string [scb_separator_str](#) (int unsigned pre_length)
Creates a new separator string for scoreboard stat tables.
- static string [scb_header_str](#) (string hn, int unsigned pre_length, bit side, string col_names[]=(" Inserts ", " Matches ", " Flushed ", " Orphans "))
Creates a new header string for a scoreboard stat table.
- static void [split_string](#) (string in, byte delim[], output string out[])
Splits the string 'in' by any of the delimiter strings in 'delim', returning the result in 'out'.
- static int [merge_string_arrays](#) (string inputs[\$][], string concat="|", output string result)
Takes a queue of string arrays, merging these into a single string.
- static string [generate_cmp_table_header](#) (int table_width, string header_text)
Generates the header section of a comparison table.
- static int [generate_cmp_table_body](#) ([cl_syoscb_item](#) items[], [cl_syoscb_cfg](#) cfg, output string result)
Generates the body of a comparison table.
- static string [generate_cmp_table_footer](#) (int table_width, [uvm_comparer](#) comparer)
Generates the footer section of a comparison table.
- static string [sprint_item](#) ([cl_syoscb_item](#) item, [cl_syoscb_cfg](#) cfg)
Utility function for printing sequence items using a table printer. This function sprints the given seq. item. using the uvm_default_table_printer. The value of the configuration object's default_printer_verbosity bit is used to control.

13.106.1 Detailed Description

A utility class holding a number of static methods for performing string manipulation.

Definition at line 2 of file cl_syoscb_string_library.svh.

13.106.2 Member Function Documentation

13.106.2.1 generate_cmp_table_body()

```
int cl_syoscb_string_library::generate_cmp_table_body (
    cl_syoscb_item items[],
    cl_syoscb_cfg cfg,
    output string result ) [static]
```

Generates the body of a comparison table.

Primarily used for inspecting mismatches.

Parameters

<i>items</i>	An array of all cl_syoscb_items that must be included in the table
<i>cfg</i>	The configuration object for the scoreboard
<i>result</i>	Handle to a string in which the result is returned

Returns

The width of the comparison table

Definition at line 210 of file cl_syoscb_string_library.svh.

References merge_string_arrays(), split_string(), and sprint_item().

Referenced by cl_syoscb_compare_base::generate_mismatch_table(), cl_syoscb_queue_locator_hash< pk_↵
syoscb::MD5_HASH_DIGEST_WIDTH >::validate_match(), and cl_syoscb_queue_locator_hash< pk_syoscb::↵
MD5_HASH_DIGEST_WIDTH >::validate_no_match().

13.106.2.2 generate_cmp_table_footer()

```
string cl_syoscb_string_library::generate_cmp_table_footer (
    int table_width,
    uvm_comparer comparer ) [static]
```

Generates the footer section of a comparison table.

Parameters

<i>table_width</i>	The width of the comparison table
<i>comparer</i>	The UVM comparer used to compare seq. items

Definition at line 249 of file `cl_syoscb_string_library.svh`.

Referenced by `cl_syoscb_compare_base::generate_miscomp_table()`, and `cl_syoscb_queue_locator_hash< pk_↵ syoscb::MD5_HASH_DIGEST_WIDTH >::validate_match()`.

13.106.2.3 `generate_cmp_table_header()`

```
string cl_syoscb_string_library::generate_cmp_table_header (
    int table_width,
    string header_text ) [static]
```

Generates the header section of a comparison table.

Parameters

<i>table_width</i>	The width of the comparison table
<i>header_text</i>	The text to be included in the header

Returns

A string containing the header

Definition at line 229 of file `cl_syoscb_string_library.svh`.

References `pad_str()`, and `split_string()`.

Referenced by `cl_syoscb_compare_base::generate_miscomp_table()`, `cl_syoscb_queue_locator_hash< pk_↵ syoscb::MD5_HASH_DIGEST_WIDTH >::validate_match()`, and `cl_syoscb_queue_locator_hash< pk_↵ syoscb::MD5_HASH_DIGEST_WIDTH >::validate_no_match()`.

13.106.2.4 `merge_string_arrays()`

```
int cl_syoscb_string_library::merge_string_arrays (
    string inputs[$][],
    string concat = "|",
    output string result ) [static]
```

Takes a queue of string arrays, merging these into a single string.

The output consists of the *i*'th lines of all entries concatenated together. Each corresponding line is joined with a line break `"\n"`. Primarily intended to be used for merging item printouts previously split by [split_string\(\)](#)

Parameters

<i>inputs</i>	A queue of string arrays. inputs[x] is a string array, and inputs [x][y] is the yth line of that string
<i>concat</i>	A concatenator to be used between strings. Defaults to " "
<i>result</i>	A string handle where the result is returned

Returns

The width of the resulting table

Definition at line 148 of file cl_syoscb_string_library.svh.

Referenced by generate_cmp_table_body().

13.106.2.5 pad_str()

```
string cl_syoscb_string_library::pad_str (
    string str,
    int unsigned max_length,
    string expand = " ",
    bit side = 0b0 ) [static]
```

Pads the input string with another string until it reaches a given length.

Parameters

<i>str</i>	The input string to pad
<i>max_length</i>	The length to pad it to
<i>expand</i>	The character(s) to insert on the left/right of the original string until <i>max_length</i> is reached
<i>side</i>	Which side of the string to insert padding on. If 1, inserts on the right, if 0, inserts on the left

Returns

The padded string

Definition at line 44 of file cl_syoscb_string_library.svh.

Referenced by cl_syoscb_queue_base::create_producer_stats(), cl_syoscb_queue_base::create_queue_report(), cl_syoscb::create_total_stats(), cl_syoscb_base::create_total_stats(), cl_syoscb::dump_join_txt(), cl_syoscb↵::dump_split_txt(), cl_syoscb::dump_split_xml(), generate_cmp_table_header(), scb_header_str(), and scb↵ separator_str().

13.106.2.6 scb_header_str()

```
string cl_syoscb_string_library::scb_header_str (
    string hn,
```

```

        int unsigned pre_length,
        bit side,
        string col_names[] = ( " Inserts " , " Matches " , " Flushed " , " Orphans " ) )
[static]

```

Creates a new header string for a scoreboard stat table.

Parameters

<i>hn</i>	The name of the table
<i>pre_length</i>	The width of the first column of the table
<i>side</i>	Whether to pad the table name with spaces on the right (1) or left (0)
<i>col_names</i>	The names of the columns in the table. Must have exactly 4 entries, each of which should be 10 characters wide

Definition at line 74 of file `cl_syoscb_string_library.svh`.

References `pad_str()`, and `scb_separator_str()`.

Referenced by `cl_syoscb::create_report()`, `cl_syoscb::intermediate_queue_stat_dump()`, and `cl_syoscb::base::report_phase()`.

13.106.2.7 scb_separator_str()

```

string cl_syoscb_string_library::scb_separator_str (
    int unsigned pre_length ) [static]

```

Creates a new separator string for scoreboard stat tables.

Parameters

<i>pre_length</i>	The width of the first column of the table
-------------------	--

Returns

The separator string

Definition at line 59 of file `cl_syoscb_string_library.svh`.

References `pad_str()`.

Referenced by `cl_syoscb::create_queues_stats()`, `cl_syoscb::base::create_report()`, `cl_syoscb::create_report()`, `cl_syoscb::base::create_scb_stats()`, `cl_syoscb::intermediate_queue_stat_dump()`, `cl_syoscb::base::report_phase()`, and `scb_header_str()`.

13.106.2.8 split_string()

```
void cl_syoscb_string_library::split_string (
    string in,
    byte delim[],
    output string out[] ) [static]
```

Splits the string 'in' by any of the delimiter strings in 'delim', returning the result in 'out'.

Example: in="Hello, world..", delim={"", " ", "."} => out={"Hello", "world"}

Parameters

<i>in</i>	The input string to be split
<i>delim</i>	An array of possible delimiter characters to be used in splitting the string
<i>out</i>	A handle to an array in which the split strings will be placed.

Definition at line 99 of file cl_syoscb_string_library.svh.

Referenced by generate_cmp_table_body(), and generate_cmp_table_header().

13.106.2.9 sprint_item()

```
string cl_syoscb_string_library::sprint_item (
    cl_syoscb_item item,
    cl_syoscb_cfg cfg ) [static]
```

Utility function for printing sequence items using a table printer. This function sprints the given seq. item. using the uvm_default_table_printer. The value of the configuration object's default_printer_verbosity bit is used to control.

how many array elements are included in long arrays. (See [cl_syoscb_cfg::default_printer_verbosity](#))

Parameters

<i>item</i>	The sequence item to sprint
<i>cfg</i>	The configuration object for the current scoreboard

Definition at line 303 of file cl_syoscb_string_library.svh.

Referenced by generate_cmp_table_body(), cl_syoscb_compare_io_2hp::primary_loop_do(), cl_syoscb_compare_io::primary_loop_do(), cl_syoscb_compare_iop::primary_loop_do(), cl_syoscb_compare_io::secondary_loop_do(), and cl_syoscb_compare_iop::secondary_loop_do().

The documentation for this class was generated from the following files:

- cl_syoscb_string_library.svh
- pk_syoscb.sv

13.107 cl_syoscb_subscriber Class Reference

Generic subscriber for the scoreboard.

Inherits `uvm_subscriber< uvm_sequence_item >`, and `uvm_subscriber< uvm_sequence_item >`.

Public Member Functions

- void `write` (`uvm_sequence_item t`)
Implementation of the write method which must be implemented when extending `uvm_subscriber`.
- virtual string `get_queue_name` ()
Subscriber API: Returns the name of the queue which this subscriber is connected to.
- virtual void `set_queue_name` (string qn)
Subscriber API: Sets the name of the queue which this subscriber is connected to.
- virtual string `get_producer` ()
Subscriber API: Returns the name of the producer which this subscriber is connected to.
- virtual void `set_producer` (string p)
Subscriber API: Sets the name of the producer which this subscriber is connected to.
- virtual void `set_mutexed_add_item_enable` (bit maie)
Subscriber API: Controls whether items should be added in a mutexed fashion or not.

Private Attributes

- string `queue_name`
The name of the queue this subscriber writes data to.
- string `producer`
The name of the producer that this is subscribed to.
- bit `mutexed_add_item_enable` = 0b0
Whether to use mutexed add_item calls (1) or non-mutexed (0)

13.107.1 Detailed Description

Generic subscriber for the scoreboard.

It provides the write method for UVM monitors and utilizes the function based API of the scoreboard to insert the items received through the write method.

Definition at line 4 of file `cl_syoscb_subscriber.svh`.

13.107.2 Member Function Documentation

13.107.2.1 set_mutexed_add_item_enable()

```
void cl_syoscb_subscriber::set_mutexed_add_item_enable (
    bit maie ) [virtual]
```

Subscriber API: Controls whether items should be added in a mutexed fashion or not.

Must be called during [cl_syoscb::build_phase](#)

Definition at line 98 of file cl_syoscb_subscriber.svh.

References [mutexed_add_item_enable](#).

Referenced by [cl_syoscb::build_phase\(\)](#).

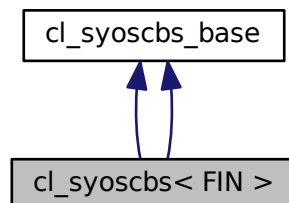
The documentation for this class was generated from the following files:

- cl_syoscb_subscriber.svh
- pk_syoscb.sv

13.108 cl_syoscbs< FIN > Class Template Reference

Default implementation of a scoreboard wrapper.

Inheritance diagram for cl_syoscbs< FIN >:



Additional Inherited Members

13.108.1 Detailed Description

```
template<typename FIN = int>
class cl_syoscbs< FIN >
```

Default implementation of a scoreboard wrapper.

FIN: The type of input transactions. Output transactions will be of type `uvm_sequence_item`

Definition at line 3 of file `cl_syoscbs.svh`.

13.108.2 Member Function Documentation

13.108.2.1 build_phase() [1/2]

```
template<typename FIN = int>
void cl_syoscbs< FIN >::build_phase (
    uvm_phase phase ) [virtual]
```

UVM build phase.

Receives a `cl_syoscbs_cfg` object, creates wrapped scoreboards and their configuration objects, forwards configuration objects to each wrapped scoreboard.

Reimplemented from `cl_syoscbs_base`.

Definition at line 68 of file `cl_syoscbs.svh`.

References `cl_syoscbs_base::cfg`, `cl_syoscbs_base::create_filters()`, `cl_syoscbs_cfg::get_cfg()`, and `cl_syoscbs_base::scbs`.

13.108.2.2 build_phase() [2/2]

```
template<typename FIN = int>
virtual void cl_syoscbs< FIN >::build_phase (
    uvm_phase phase ) [virtual]
```

UVM build phase.

Receives a `cl_syoscbs_cfg` object, creates wrapped scoreboards and their configuration objects, forwards configuration objects to each wrapped scoreboard.

Reimplemented from `cl_syoscbs_base`.

13.108.2.3 get_filter_trfm() [1/2]

```
template<typename FIN = int>
virtual cl_syoscbs::tp_wrapper_filter_trfm cl_syoscbs< FIN >::get_filter_trfm (
    string queue_name,
    string producer_name,
    int unsigned idx ) [virtual]
```

Gets a handle to a filter transform This convenience wrapper gets a filter transform and typecasts it to the correct type for the user.

Parameters

<i>queue_name</i>	The name of the queue to connect the filter to
<i>producer_name</i>	The name of the producer that produced data going into this filter
<i>idx</i>	The index of the scoreboard in which this queue exists

Returns

A filter transform object, if all parameters are valid. If the parameters do not specify a valid filter, returns null and prints a UVM_INFO/DEBUG message

Definition at line 47 of file cl_syoscbs.svh.

References cl_syoscbs_base::get_filter_trfm_base().

13.108.2.4 get_filter_trfm() [2/2]

```
template<typename FIN = int>
virtual cl_syoscbs::tp_wrapper_filter_trfm cl_syoscbs< FIN >::get_filter_trfm (
    string queue_name,
    string producer_name,
    int unsigned idx ) [virtual]
```

Gets a handle to a filter transform This convenience wrapper gets a filter transform and typecasts it to the correct type for the user.

Parameters

<i>queue_name</i>	The name of the queue to connect the filter to
<i>producer_name</i>	The name of the producer that produced data going into this filter
<i>idx</i>	The index of the scoreboard in which this queue exists

Returns

A filter transform object, if all parameters are valid. If the parameters do not specify a valid filter, returns null and prints a UVM_INFO/DEBUG message

Definition at line 48 of file pk_syoscb.sv.

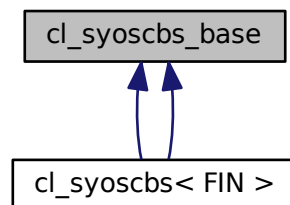
The documentation for this class was generated from the following files:

- cl_syoscbs.svh
- pk_syoscb.sv

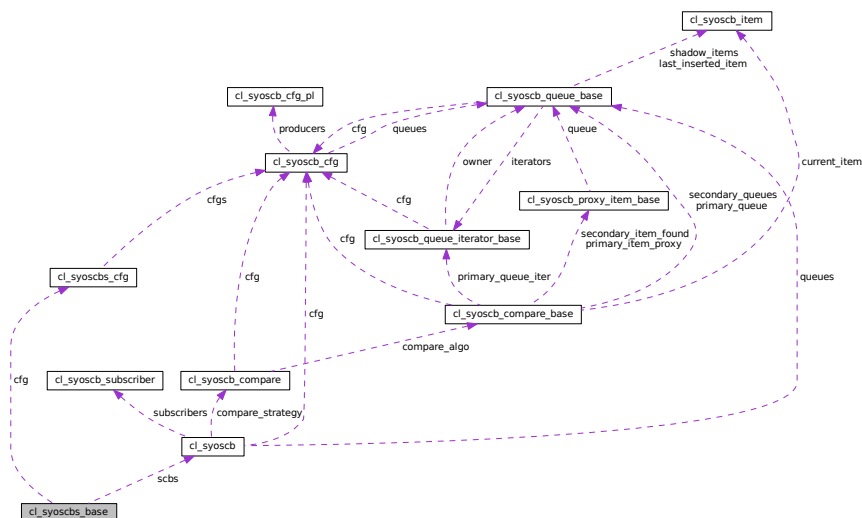
13.109 cl_syoscbs_base Class Reference

Base class for a wrapper around multiple SyoSil Scoreboards.

Inheritance diagram for cl_syoscbs_base:



Collaboration diagram for cl_syoscbs_base:



Public Member Functions

- virtual void [build_phase](#) (uvm_phase phase)
UVM build phase.
- virtual void [connect_phase](#) (uvm_phase phase)
UVM connect phase. syoscbs_base only calls super.connect_phase.
- virtual void [report_phase](#) (uvm_phase phase)
UVM report phase.
- virtual [cl_syoscbs_cfg](#) [get_cfg](#) ()
Gets the configuration object associated with this scoreboard wrapper.

- virtual [cl_syoscb get_scb](#) (int unsigned idx)
Scoreboard Wrapper API: Get a handle to a scoreboard inside this wrapper
- virtual void [flush_queues_all](#) ()
Scoreboard Wrapper API: Flush all queues of all scoreboards.
- virtual void [flush_queues_by_index](#) (int unsigned idxs[]={}, string queue_names[]={})
Scoreboard Wrapper API: Flush the queues indicated by queue_names of the scoreboards with indexes idxs.
- virtual void [flush_queues_by_name](#) (string scb_names[]={}, string queue_names[]={})
Scoreboard Wrapper API: Flush the queues indicated by queue_names of the scoreboards with names scb_names.
- virtual void [compare_control_all](#) (bit cc)
Scoreboard Wrapper API: Disable or enable the compare in all scoreboards.
- virtual void [compare_control_by_index](#) (int unsigned idxs[]={}, bit cc)
Scoreboard Wrapper API: Disable or enable the compare in scoreboards with given indexes.
- virtual void [compare_control_by_name](#) (string scb_names[]={}, bit cc)
Scoreboard Wrapper API: Disable or enable the compare in scoreboards with given names.
- virtual string [create_report](#) (int unsigned offset, int unsigned first_column_width)
Scoreboard Wrapper API: Creates a summary report once simulation has finished.
- virtual uvm_component [get_filter_trfm_base](#) (string queue_name, string producer_name, int unsigned idx)
Scoreboard Wrapper API: Gets a handle to a filter transform as a uvm_component.
- virtual void [do_print](#) (uvm_printer printer)
Implementation of UVM do_print-hmethod Prints information of all wrapped scoreboards, as well as all filter transforms.

Protected Member Functions

- virtual string [create_scb_stats](#) (int unsigned offset, int unsigned first_column_width)
Returns a string containing the tables with statistics of the different scoreboards.
- virtual void [create_filters](#) (int unsigned idx, [cl_syoscb_cfg](#) cfg)
Create all filter transforms for the given scoreboard.
- virtual void [connect_filters](#) (int unsigned idx, [cl_syoscb_cfg](#) cfg)
Connects all filter transforms with their respective subscribers in the scoreboard.
- virtual void [create_filter](#) (string queue_name, string producer_name, int unsigned idx)
Creates a filter for given scoreboard/queue name/producer combination.
- virtual void [connect_filter_and_subscriber](#) (string queue_name, string producer_name, int unsigned idx)
Connects a filter's output to a scoreboard's subscriber.
- virtual string [create_total_stats](#) (int unsigned offset, int unsigned first_column_width)
Returns a table with summed scoreboard statistics for all wrapped scoreboards.
- virtual string [get_scb_failed_checks](#) ()
Gets information on whether any of the wrapped scoreboards failed to pass error checks.

Protected Attributes

- [cl_syoscb scbs](#) []
Array holding handles to all scoreboards.
- [cl_syoscb_cfg](#) cfg
Handle to scoreboard wrapper configuration object.
- uvm_component [fts](#) [][string][string]
Array holding handles to filter transforms, used to transform inputs of one type to outputs of another type, for feeding into the wrapped scoreboards.

13.109.1 Detailed Description

Base class for a wrapper around multiple SyoSil Scoreboards.

An implementation is found in [cl_syoscbs_base](#)

Definition at line 3 of file cl_syoscbs_base.svh.

13.109.2 Member Function Documentation

13.109.2.1 build_phase()

```
void cl_syoscbs_base::build_phase (
    uvm_phase phase ) [virtual]
```

UVM build phase.

Receives a [cl_syoscbs_cfg](#) object, creates wrapped scoreboards and their configuration objects, forwards configuration objects to each wrapped scoreboard.

Reimplemented in [cl_syoscbs< FIN >](#), and [cl_syoscbs< FIN >](#).

Definition at line 87 of file cl_syoscbs_base.svh.

References [cl_syoscbs_cfg::get_cfg\(\)](#), [cl_syoscbs_cfg::get_no_scbs\(\)](#), [cl_syoscbs_cfg::get_print_cfg\(\)](#), [cl_syoscbs_cfg::get_scb_name\(\)](#), [cl_syoscbs_cfg::get_scbs_name\(\)](#), and [cl_syoscbs_cfg::set_scbs_name\(\)](#).

13.109.2.2 compare_control_all()

```
void cl_syoscbs_base::compare_control_all (
    bit cc ) [virtual]
```

Scoreboard Wrapper API: Disable or enable the compare in all scoreboards.

Parameters

<code>cc</code>	Compare control bit. If 0b1, enables compare in all scoreboards. If 10b0, disables compare
-----------------	--

Definition at line 288 of file cl_syoscbs_base.svh.

References [compare_control_by_index\(\)](#).

13.109.2.3 compare_control_by_index()

```
void cl_syoscbs_base::compare_control_by_index (
    int unsigned idxs[] = {},
    bit cc ) [virtual]
```

Scoreboard Wrapper API: Disable or enable the compare in scoreboards with given indexes.

If no indexes are specified, all scoreboards are affected.

Parameters

<i>idxs</i>	The indexes of the scoreboards to enable/disable compare control for
<i>cc</i>	Compare control bit. If 0b1, enables compare in all scoreboards. If 10b0, disables compare

Definition at line 296 of file `cl_syoscbs_base.svh`.

Referenced by `compare_control_all()`.

13.109.2.4 compare_control_by_name()

```
void cl_syoscbs_base::compare_control_by_name (
    string scb_names[] = {},
    bit cc ) [virtual]
```

Scoreboard Wrapper API: Disable or enable the compare in scoreboards with given names.

If no names are specified, all scoreboards are affected.

Parameters

<i>scb_names</i>	The names of the scoreboards to enable/disable compare control for
<i>cc</i>	Compare control bit. If 0b1, enables compare in all scoreboards. If 10b0, disables compare

Definition at line 312 of file `cl_syoscbs_base.svh`.

13.109.2.5 connect_filter_and_subscriber()

```
void cl_syoscbs_base::connect_filter_and_subscriber (
    string queue_name,
    string producer_name,
    int unsigned idx ) [protected], [virtual]
```

Connects a filter's output to a scoreboard's subscriber.

Parameters

<i>queue_name</i>	The name of the queue to connect the filter to
<i>producer_name</i>	The name of the producer that produced data going into this filter
<i>fts_idx</i>	The index of the scoreboard in which this queue exists

Note

Abstract method, will throw UVM_FATAL if called. Must override in a child class

Definition at line 479 of file cl_syoscbs_base.svh.

Referenced by connect_filters().

13.109.2.6 connect_filters()

```
void cl_syoscbs_base::connect_filters (
    int unsigned idx,
    cl_syoscb_cfg cfg ) [protected], [virtual]
```

Connects all filter transforms with their respective subscribers in the scoreboard.

Should be called in the UVM connect phase

Parameters

<i>idx</i>	Index of the scoreboard for which all filters should be connected
<i>cfg</i>	The configuration object for that scoreboard

Definition at line 451 of file cl_syoscbs_base.svh.

References `cfg`, `connect_filter_and_subscriber()`, and `cl_syoscb_cfg_pl::list`.

Referenced by `cl_syoscbs< FIN >::connect_phase()`.

13.109.2.7 create_filter()

```
void cl_syoscbs_base::create_filter (
    string queue_name,
    string producer_name,
    int unsigned idx ) [protected], [virtual]
```

Creates a filter for given scoreboard/queue name/producer combination.

Parameters

<i>queue_name</i>	The name of the queue to connect the filter to
<i>producer_name</i>	The name of the producer that produced data going into this filter
<i>idx</i>	The index of the scoreboard in which this queue exists

Note

Abstract method. Must override in a child class to create filters of the correct type

Definition at line 468 of file `cl_syoscbs_base.svh`.

Referenced by `create_filters()`.

13.109.2.8 `create_filters()`

```
void cl_syoscbs_base::create_filters (
    int unsigned idx,
    cl_syoscb_cfg cfg ) [protected], [virtual]
```

Create all filter transforms for the given scoreboard.

Should be called in the UVM build phase

Parameters

<i>idx</i>	Index of the scoreboard to create filters for
<i>cfg</i>	The configuration object for that scoreboard

Definition at line 434 of file `cl_syoscbs_base.svh`.

References `cfg`, `create_filter()`, and `cl_syoscb_cfg_pl::list`.

Referenced by `cl_syoscbs< FIN >::build_phase()`.

13.109.2.9 `create_report()`

```
string cl_syoscbs_base::create_report (
    int unsigned offset,
    int unsigned first_column_width ) [virtual]
```

Scoreboard Wrapper API: Creates a summary report once simulation has finished.

The report contains insert/match/flush/orphan statistics for the wrapped scoreboards. If the `cl_syoscb_cfg::gen↔_enable_scb_stats` configuration knob is active then the report of the different queues in each scoreboard is also included. At the end of the report is a table with the statistics of all scoreboards.

Parameters

<i>offset</i>	Horizontal offset at which text should start
<i>first_column_width</i>	The width of the first column in the output table

Returns

A string containing the entire report, ready to print

Definition at line 347 of file cl_syoscbs_base.svh.

References [create_scb_stats\(\)](#), [create_total_stats\(\)](#), and [cl_syoscb_string_library::scb_separator_str\(\)](#).

Referenced by [report_phase\(\)](#).

13.109.2.10 create_scb_stats()

```
string cl_syoscbs_base::create_scb_stats (
    int unsigned offset,
    int unsigned first_column_width ) [protected], [virtual]
```

Returns a string containing the tables with statistics of the different scoreboards.

If the [cl_syoscbs_cfg::enable_scb_stats](#) configuration knob is active for a given scoreboard, the report of the individual queues of that scoreboard is also included.

Parameters

<i>offset</i>	Horizontal offset at which text should start. Depends on the level of nested calls (see cl_syoscbs_base::report_phase implementation)
<i>first_column_width</i>	The width of the first column in the output table

Returns

A string containing the table

Definition at line 396 of file cl_syoscbs_base.svh.

References [cl_syoscb::create_report_contents\(\)](#), [cl_syoscb::create_total_stats\(\)](#), [cl_syoscbs_cfg::get_enable_scb_stats\(\)](#), and [cl_syoscb_string_library::scb_separator_str\(\)](#).

Referenced by [create_report\(\)](#).

13.109.2.11 create_total_stats()

```
string cl_syoscbs_base::create_total_stats (
    int unsigned offset,
    int unsigned first_column_width ) [protected], [virtual]
```

Returns a table with summed scoreboard statistics for all wrapped scoreboards.

Parameters

<i>offset</i>	Horizontal offset at which text should start
<i>first_column_width</i>	The width of the first column in the output table

Returns

A string containing the table

Definition at line 363 of file `cl_syoscbs_base.svh`.

References `cl_syoscbs::get_total_cnt_add_items()`, `cl_syoscbs::get_total_cnt_flushed_items()`, `cl_syoscbs::get_total_queue_size()`, and `cl_syoscbs_string_library::pad_str()`.

Referenced by `create_report()`.

13.109.2.12 do_print()

```
void cl_syoscbs_base::do_print (
    uvm_printer printer ) [virtual]
```

Implementation of UVM `do_print`-hmethod Prints information of all wrapped scoreboards, as well as all filter transforms.

Parameters

<i>printer</i>	The UVM printer to use
----------------	------------------------

Definition at line 489 of file `cl_syoscbs_base.svh`.

References `fts`, and `scbs`.

13.109.2.13 flush_queues_by_index()

```
void cl_syoscbs_base::flush_queues_by_index (
    int unsigned idxs[] = {},
    string queue_names[] = {} ) [virtual]
```

Scoreboard Wrapper API: Flush the queues indicated by `queue_names` of the scoreboards with indexes `idxs`.

If no indexes are specified, all scoreboards will be affected by the flush. If no queue names are specified all queues are flushed.

Parameters

<i>idxs</i>	indexes of the scoreboards to flush
<i>queue_names</i>	Names of the queues under those scoreboards to flush

Definition at line 225 of file cl_syoscbs_base.svh.

Referenced by flush_queues_all().

13.109.2.14 flush_queues_by_name()

```
void cl_syoscbs_base::flush_queues_by_name (
    string scb_names[] = {},
    string queue_names[] = {} ) [virtual]
```

Scoreboard Wrapper API: Flush the queues indicated by queue_names of the scoreboards with names scb_names.

If no scoreboard names are specified all the scoreboards will be affected by the flush. If no queue names are specified all queues are flushed.

Parameters

<i>scb_names</i>	Names of the scoreboards to flush
<i>queue_names</i>	Names of the queues under those scoreboards to flush

Definition at line 259 of file cl_syoscbs_base.svh.

13.109.2.15 get_filter_trfm_base()

```
uvm_component cl_syoscbs_base::get_filter_trfm_base (
    string queue_name,
    string producer_name,
    int unsigned idx ) [virtual]
```

Scoreboard Wrapper API: Gets a handle to a filter transform as a uvm_component.

The end user must typecast this uvm_component to the correct type, based on the kind of filter transforms that is implemented

Parameters

<i>queue_name</i>	The name of the queue to connect the filter to
<i>producer_name</i>	The name of the producer that produced data going into this filter
<i>fts_idx</i>	The index of the scoreboard in which this queue exists

Returns

A uvm_component which represents a filter, if all parameters are valid. If the parameters do not specify a valid filter, returns null and prints a UVM_INFO/DEBUG message

Definition at line 194 of file cl_syoscbs_base.svh.

Referenced by `cl_syoscbs< FIN >::get_filter_trfm()`.

13.109.2.16 `get_scb()`

```
cl_syoscb cl_syoscbs_base::get_scb (
    int unsigned idx ) [virtual]
```

Scoreboard Wrapper API: Get a handle to a scoreboard inside this wrapper

Parameters

<i>idx</i>	The index of that scoreboard
------------	------------------------------

Returns

A handle to scoreboard [*idx*]. If *idx* \geq number of scoreboards, throws a `uvm_fatal` error

Definition at line 175 of file `cl_syoscbs_base.svh`.

13.109.2.17 `get_scb_failed_checks()`

```
string cl_syoscbs_base::get_scb_failed_checks ( ) [protected], [virtual]
```

Gets information on whether any of the wrapped scoreboards failed to pass error checks.

These error checks include orphan checking and no-insertion checks.

Returns

A string combining the error checks of all queues.

Definition at line 417 of file `cl_syoscbs_base.svh`.

References `cl_syoscb::get_failed_checks()`.

Referenced by `report_phase()`.

13.109.2.18 `report_phase()`

```
void cl_syoscbs_base::report_phase (
    uvm_phase phase ) [virtual]
```

UVM `report_phase`.

Prints the status of all scoreboard instances.

Definition at line 128 of file `cl_syoscbs_base.svh`.

References `create_report()`, `cl_syoscbs_cfg::get_disable_report()`, `cl_syoscbs_cfg::get_max_length_producer()`, `cl_syoscbs_cfg::get_max_length_queue_name()`, `cl_syoscbs_cfg::get_max_length_scb_name()`, `get_scb_failed_checks()`, `cl_syoscb_string_library::scb_header_str()`, and `cl_syoscb_string_library::scb_separator_str()`.

13.109.3 Member Data Documentation

13.109.3.1 fts

```
uvm_component cl_syoscbs_base::fts [protected]
```

Array holding handles to filter transforms, used to transform inputs of one type to outputs of another type, for feeding into the wrapped scoreboards.

Declared as type `uvm_component` for flexibility. See example of implementation in [cl_syoscbs_base](#). AA is indexed by [scb_idx][queue_name][producer_name]

Definition at line 18 of file `cl_syoscbs_base.svh`.

Referenced by `cl_syoscbs< FIN >::connect_phase()`, and `do_print()`.

The documentation for this class was generated from the following files:

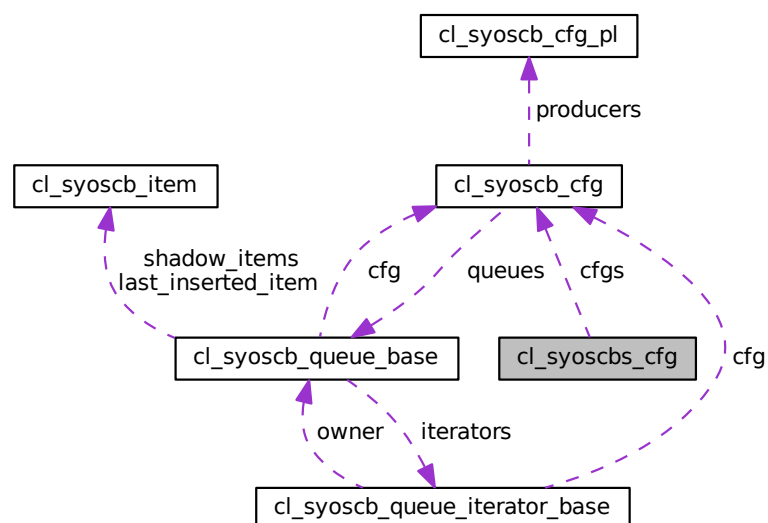
- `cl_syoscbs_base.svh`
- `pk_syoscb.sv`

13.110 cl_syoscbs_cfg Class Reference

Configuration object for the [cl_syoscbs_base](#) scoreboard wrapper.

Inherits `uvm_object`, and `uvm_object`.

Collaboration diagram for `cl_syoscbs_cfg`:



Public Member Functions

- virtual void `init` (string `scbs_name`="", int unsigned `no_scbs`, string `scb_names`[], string `queues`[], string `producers`[])
- Configuration API: Initializes the scoreboard wrapper and all contained scoreboards.*
- virtual void `set_cfg` (cl_syoscb_cfg `cfg`, int unsigned `idx`)
- Configuration API: Sets the configuration object for the scoreboard at a given index*
- virtual `cl_syoscb_cfg` `get_cfg` (int unsigned `idx`)
- Configuration API: Returns the configuration object of the scoreboard with a given index*
- virtual void `set_no_scbs` (int unsigned `no_scbs`)
- Configuration API: Sets the number of scoreboards that should be wrapped.*
- virtual int unsigned `get_no_scbs` ()
- Configuration API: Returns the number of scoreboards wrapped by this wrapper.*
- virtual void `set_scbs_name` (string `scbs_name`)
- Configuration API: Sets the name of this scoreboard wrapper*
- virtual string `get_scbs_name` ()
- Configuration API: Returns the name of this scoreboard wrapper*
- virtual void `set_scb_names` (string `scb_names`[], int unsigned `idxs`[]={})
- Configuration API: Sets the scoreboard name of all scoreboards under this wrapper.*
- virtual void `get_scb_names` (output string `scb_names`[], input int unsigned `idxs`[]={})
- Configuration API: Returns the names of some or all scoreboards wrapped by this wrapper.*
- virtual void `set_queues` (string `queues`[], int unsigned `idxs`[]={})
- Configuration API: Sets the legal queue names for the scoreboards indicated by the idxs argument.*
- virtual void `get_queues` (output string `queues`[], input int unsigned `idx`)
- Configuration API: Returns the names of the queues for the scoreboard with a given index*
- virtual void `set_producers` (string `producer`, string `queues`[]={}, int unsigned `idxs`[]={})
- Configuration API: Sets the producer for the specified queues of the scoreboards with given indexes If no queues are specified, the producer is set for all the queues.*
- virtual void `set_queue_type` (t_scb_queue_type `queue_types`[], int unsigned `idxs`[]={})
- Configuration API: Sets the queue types for the given scoreboards inside the wrapper.*
- virtual void `set_compare_type` (t_scb_compare_type `compare_types`[], int unsigned `idxs`[]={})
- Configuration API: Sets the compare strategy for the given scoreboards inside the wrapper*
- virtual int `get_scb_index_by_name` (string `scb_name`)
- Configuration API: Gets the index of a scoreboard with a given name*
- virtual void `set_scb_trigger_greediness` (int unsigned `idxs`[]={}, t_scb_compare_greed `tg`[])
- Configuration API: Sets trigger greediness status for all or a subset of the scoreboards.*
- virtual void `get_scb_trigger_greediness` (output t_scb_compare_greed `tg`[], input int unsigned `idxs`[]={})
- Configuration API: Gets the trigger greediness status for all or a subset of the scoreboards.*
- virtual void `set_scb_end_greediness` (int unsigned `idxs`[]={}, t_scb_compare_greed `eg`[])
- Configuration API: Sets the end greediness status for all or a subset of the scoreboards.*
- virtual void `get_scb_end_greediness` (output t_scb_compare_greed `eg`[], input int unsigned `idxs`[]={})
- Configuration API: Gets the end greediness status for all or a subset of the scoreboards.*
- virtual void `set_disable_report` (bit `dr`)
- Configuration API: Sets the value of the `disable_report` member variable*
- virtual bit `get_disable_report` ()
- Configuration API: Returns the value of the `disable_report` member variable*
- virtual void `set_enable_scb_stats` (input int unsigned `idxs`[]={}, bit `ess`)
- Configuration API: Sets the value of the `enable_scb_stats` flag for all or a subset of scoreboards*
- virtual bit `get_enable_scb_stats` (int unsigned `idx`)
- Configuration API: Returns the value of the `enable_scb_stats` member variable for the scoreboard at the given index*
- virtual int unsigned `get_max_length_scb_name` ()

- Returns the length of the longest scoreboard name that is wrapped by this.*
 - virtual int unsigned [get_max_length_queue_name](#) ()
- Returns the length of the longest queue name that is wrapped by this.*
 - virtual int unsigned [get_max_length_producer](#) ()
- Returns the length of the producer name with maximum length.*
 - virtual void [set_print_cfg](#) (bit pc)
- Configuration API:** Sets the value of the [print_cfg](#) member variable
 - virtual bit [get_print_cfg](#) ()
- Gets the value of the [print_cfg](#) member variable.*
 - virtual void [do_print](#) (uvm_printer printer)
- Custom do_print implementation. Print only the wrapped configuration objects which have [print_cfg](#) == 1.*

Protected Member Functions

- virtual bit [is_scb_names_unique](#) (input string scb_name)

Checks if a given name is not yet used by a scoreboard under this wrapper.

Private Attributes

- [cl_syoscbs_cfg](#) [cfgs](#) []

Array holding handles to all the UVM scoreboard configurations.
- string [scbs_name](#)

Scoreboard wrapper name.
- int unsigned [no_scbs](#)

Number of scoreboards.
- bit [disable_report](#)

Whether to disable report printing in the report_phase.
- bit [enable_scb_stats](#) []

Enable/disable the printing of scb statistics per queue by each scb.
- bit [print_cfg](#) = 0b1

Whether to print scoreboard wrapper configuration in the UVM build_phase.

13.110.1 Detailed Description

Configuration object for the [cl_syoscbs_base](#) scoreboard wrapper.

Definition at line 2 of file cl_syoscbs_cfg.svh.

13.110.2 Member Function Documentation

13.110.2.1 [get_cfg\(\)](#)

```
cl_syoscbs_cfg cl_syoscbs_cfg::get_cfg (
    int unsigned idx ) [virtual]
```

Configuration API: Returns the configuration object of the scoreboard with a given index

Parameters

<i>idx</i>	The index of the scoreboard configuration to retrieve
------------	---

Returns

That scoreboard configuration, or null if none could be found

Note

If the index is invalid, throws a UVM_FATAL

Definition at line 147 of file cl_syoscbs_cfg.svh.

Referenced by cl_syoscbs< FIN >::build_phase(), cl_syoscbs_base::build_phase(), and cl_syoscbs< FIN >::connect_phase().

13.110.2.2 get_queues()

```
void cl_syoscbs_cfg::get_queues (
    output string queues[],
    input int unsigned idx ) [virtual]
```

Configuration API: Returns the names of the queues for the scoreboard with a given index

Parameters

<i>queues</i>	Handle to an array where queue names are returned. Should not point to an existing array, as a new array is allocated
<i>ids</i>	The index of the scoreboard to get queue names for

Note

If *idx* >= the number of scoreboards, a UVM_FATAL is issued

Definition at line 306 of file cl_syoscbs_cfg.svh.

References cl_syoscb_cfg::get_queues().

13.110.2.3 get_scb_end_greediness()

```
void cl_syoscbs_cfg::get_scb_end_greediness (
    output t_scb_compare_greed eg[],
    input int unsigned idxs[] = {} ) [virtual]
```

Configuration API: Gets the end greediness status for all or a subset of the scoreboards.

Parameters

<i>eg</i>	The } greediness levels of the requested scoreboards. If <i>idxs</i> is empty, <i>eg</i> [<i>i</i>] is the end greediness of <i>scb</i> [<i>i</i>]. Otherwise, <i>eg</i> [<i>i</i>] is the end greediness of <i>scb</i> [<i>idxs</i> [<i>i</i>]]
<i>idxs</i>	The indexes of the scoreboards for which to get the } greed level. If empty, all greed levels are returned.

Definition at line 469 of file *cl_syoscbs_cfg.svh*.

13.110.2.4 get_scb_index_by_name()

```
int cl_syoscbs_cfg::get_scb_index_by_name (
    string scb_name ) [virtual]
```

Configuration API: Gets the index of a scoreboard with a given name

Parameters

<i>scb_name</i>	The name of the scoreboard to find the index of
-----------------	---

Returns

The index of that scoreboard, -1 if the name did not match any scoreboard

Definition at line 402 of file *cl_syoscbs_cfg.svh*.

13.110.2.5 get_scb_names()

```
void cl_syoscbs_cfg::get_scb_names (
    output string scb_names[],
    input int unsigned idxs[] = {} ) [virtual]
```

Configuration API: Returns the names of some or all scoreboards wrapped by this wrapper.

If *idxs* is empty, all names are returned. Otherwise, only the names at the requested indexes are returned.

Parameters

<i>scb_names</i>	Handle to a string array where scoreboard names are returned. Should not point to an existing array, as a new array is allocated
<i>idxs</i>	The indexes of the scoreboard names that should be returned. If empty, all names are returned such that <i>scb_names</i> [<i>i</i>] corresponds to <i>scb</i> [<i>i</i>]. Otherwise, <i>scb_names</i> [<i>i</i>] = <i>scbs</i> [<i>idxs</i> [<i>i</i>]]

Definition at line 270 of file *cl_syoscbs_cfg.svh*.

Referenced by *get_max_length_scb_name()*, and *is_scb_names_unique()*.

13.110.2.6 `get_scb_trigger_greediness()`

```
void cl_syoscbs_cfg::get_scb_trigger_greediness (
    output t_scb_compare_greed tg[],
    input int unsigned idxs[] = {} ) [virtual]
```

Configuration API: Gets the trigger greediness status for all or a subset of the scoreboards.

Parameters

<i>tg</i>	The trigger greediness levels of the requested scoreboards. If <i>idxs</i> is empty, <i>tg[i]</i> is the trigger greediness of <i>scb[i]</i> . Otherwise, <i>tg[i]</i> is the trigger greediness of <i>scb[idxs[i]]</i>
<i>idxs</i>	The indexes of the scoreboards for which to get the trigger greed level. If empty, all greed levels are returned.

Definition at line 432 of file `cl_syoscbs_cfg.svh`.

13.110.2.7 `init()`

```
void cl_syoscbs_cfg::init (
    string scbs_name = "",
    int unsigned no_scbs,
    string scb_names[],
    string queues[],
    string producers[] ) [virtual]
```

Configuration API: Initializes the scoreboard wrapper and all contained scoreboards.

See [set_scb_names](#) for important restrictions on the values of parameter `scb_names`

Parameters

<i>scbs_name</i>	The name of the scoreboard wrapper
<i>no_scbs</i>	Number of scoreboards to wrap
<i>scb_names</i>	Names of the scoreboards that should be wrapped.
<i>queues</i>	Names of the queues that should be created in <i>all</i> scoreboards given by <i>scb_names</i>
<i>producers</i>	Names of the producers that should be created for <i>all</i> queues in <i>all</i> scoreboards

Definition at line 106 of file `cl_syoscbs_cfg.svh`.

References `set_no_scbs()`, `set_producers()`, `set_queues()`, `set_scb_names()`, and `set_scbs_name()`.

13.110.2.8 `is_scb_names_unique()`

```
bit cl_syoscbs_cfg::is_scb_names_unique (
    input string scb_name ) [protected], [virtual]
```

Checks if a given name is not yet used by a scoreboard under this wrapper.

Parameters

<i>scb_name</i>	The name that should be checked against all other scoreboard names
-----------------	--

Definition at line 586 of file cl_syoscbs_cfg.svh.

References `get_scb_names()`.

13.110.2.9 set_cfg()

```
void cl_syoscbs_cfg::set_cfg (
    cl_syoscb_cfg cfg,
    int unsigned idx ) [virtual]
```

Configuration API: Sets the configuration object for the scoreboard at a given index

Parameters

<i>cfg</i>	The scoreboard configuration to set
<i>idx</i>	The index of the scoreboard config to set

Note

If the index is invalid, throws a UVM_FATAL

Definition at line 126 of file cl_syoscbs_cfg.svh.

References `cl_syoscb_cfg::set_disable_report()`, and `cl_syoscb_cfg::set_print_cfg()`.

Referenced by `set_no_scbs()`.

13.110.2.10 set_compare_type()

```
void cl_syoscbs_cfg::set_compare_type (
    t_scb_compare_type compare_types[],
    int unsigned idxs[] = {} ) [virtual]
```

Configuration API: Sets the compare strategy for the given scoreboards inside the wrapper

Parameters

<i>compare_types</i>	The compare type that should be used for a scoreboard.
<i>idxs</i>	The indexes of the scoreboards that should have their queue type set. If <i>idxs</i> is empty, <i>compare_types[i]</i> is applied to <i>scb[i]</i> . Otherwise, <i>compare_types[i]</i> is applied to <i>scb[idxs[i]]</i>

Definition at line 380 of file `cl_syoscbs_cfg.svh`.

13.110.2.11 `set_enable_scb_stats()`

```
void cl_syoscbs_cfg::set_enable_scb_stats (
    input int unsigned idxs[] = {},
    bit ess ) [virtual]
```

Configuration API: Sets the value of the [enable_scb_stats](#) flag for all or a subset of scoreboards

Parameters

<i>idxs</i>	The indexes of the scoreboards to set the value of the flag for. If empty, the value is set for all scoreboards.
<i>ess</i>	The value to set the flag to

Definition at line 498 of file `cl_syoscbs_cfg.svh`.

13.110.2.12 `set_no_scbs()`

```
void cl_syoscbs_cfg::set_no_scbs (
    int unsigned no_scbs ) [virtual]
```

Configuration API: Sets the number of scoreboards that should be wrapped.

Creates an empty scoreboard configuration for each scoreboard. If this has previously been called, previously existing scoreboard configs are preserved. If the new number of scoreboards is greater than the old, additional configs are created. If the new number of scoreboards is smaller than the old, some of the old configs are discarded.

Parameters

<i>no_scbs</i>	The number of scoreboards
----------------	---------------------------

Definition at line 164 of file `cl_syoscbs_cfg.svh`.

References `cfgs`, `enable_scb_stats`, `no_scbs`, and `set_cfg()`.

Referenced by `init()`.

13.110.2.13 `set_producers()`

```
void cl_syoscbs_cfg::set_producers (
    string producer,
    string queues[] = {},
    int unsigned idxs[] = {} ) [virtual]
```


Configuration API: Sets the producer for the specified queues of the scoreboards with given indexes. If no queues are specified, the producer is set for all the queues.

If no indices are specified, the producer is set for the queues of all scoreboards.

Parameters

<i>producer</i>	The name of the producer that should be associated with some queues
<i>queues</i>	The names of the queues that the producer can generate for

Note

If `idx >=` the number of scoreboards, a UVM_FATAL is issued

Definition at line 323 of file `cl_syoscbs_cfg.svh`.

Referenced by `init()`.

13.110.2.14 `set_queue_type()`

```
void cl_syoscbs_cfg::set_queue_type (
    t_scb_queue_type queue_types[],
    int unsigned idxs[] = {} ) [virtual]
```

Configuration API: Sets the queue types for the given scoreboards inside the wrapper.

Parameters

<i>queue_types</i>	The queue type that should be used for a scoreboard.
<i>idxs</i>	The indexes of the scoreboards that should have their queue type set. If <code>idxs</code> is empty, <code>queue_types[i]</code> is applied to <code>scb[i]</code> . Otherwise, <code>queue_types[i]</code> is applied to <code>scb[idxs[i]]</code>

Definition at line 364 of file `cl_syoscbs_cfg.svh`.

13.110.2.15 `set_queues()`

```
void cl_syoscbs_cfg::set_queues (
    string queues[],
    int unsigned idxs[] = {} ) [virtual]
```

Configuration API: Sets the legal queue names for the scoreboards indicated by the `idxs` argument.

If `idxs` is empty, the given queue names are set for all scoreboards

Parameters

<i>queues</i>	The queue names that should be used for the given scoreboards
<i>idxs</i>	The indexes of the scoreboards that should have these names. If empty, all scoreboards get these queue names. or for the scoreboards specified in the <code>idxs</code> argument.

Definition at line 283 of file cl_syoscbs_cfg.svh.

Referenced by init().

13.110.2.16 set_scb_end_greediness()

```
void cl_syoscbs_cfg::set_scb_end_greediness (
    int unsigned idxs[] = {},
    t_scb_compare_greed eg[] ) [virtual]
```

Configuration API: Sets the end greediness status for all or a subset of the scoreboards.

Parameters

<i>idxs</i>	The indexes of the scoreboards that should have their } greed level set. If idxs is empty, eg[i] is applied to scb[i]. Otherwise, eg[i] is applied to scb[idxs[i]]
<i>eg</i>	The } greed level that should be used for a scoreboard.

Definition at line 451 of file cl_syoscbs_cfg.svh.

13.110.2.17 set_scb_names()

```
void cl_syoscbs_cfg::set_scb_names (
    string scb_names[],
    int unsigned idxs[] = {} ) [virtual]
```

Configuration API: Sets the scoreboard name of all scoreboards under this wrapper.

- If the 'names' and 'idxs' arguments are empty, scoreboards are given auto-generated name: (scb[x])
- If the 'names' argument has exactly one entry and 'idxs' is empty, scoreboards are named: (<names[0]>[x])
- If the 'names' argument and 'idxs' argument both have the same number of entries, scoreboards are given names based on the idxs: scb[idxs[i]].name = scb_names[i]

Note

If multiple SCB names are passed, these must be unique. Otherwise, a UVM_FATAL is issued
If the parameters do not follow one of the three structures presented, a UVM_FATAL is issued

Parameters

<i>scb_names</i>	The names that scoreboards should be assigned.
<i>idxs</i>	The indexes at which a given scoreboard name should be given.

Definition at line 205 of file cl_syoscbs_cfg.svh.

Referenced by `init()`.

13.110.2.18 `set_scb_trigger_greediness()`

```
void cl_syoscbs_cfg::set_scb_trigger_greediness (
    int unsigned idxs[] = {},
    t_scb_compare_greed tg[] ) [virtual]
```

Configuration API: Sets trigger greediness status for all or a subset of the scoreboards.

Parameters

<i>idxs</i>	The indexes of the scoreboards that should have their trigger greed level set. If <i>idxs</i> is empty, <i>tg</i> [<i>i</i>] is applied to <i>scb</i> [<i>i</i>]. Otherwise, <i>tg</i> [<i>i</i>] is applied to <i>scb</i> [<i>idxs</i> [<i>i</i>]]
<i>tg</i>	The trigger greed level that should be used for a scoreboard.

Definition at line 414 of file `cl_syoscbs_cfg.svh`.

13.110.3 Member Data Documentation

13.110.3.1 `disable_report`

```
bit cl_syoscbs_cfg::disable_report [private]
```

Whether to disable report printing in the `report_phase`.

- 0 => Reports are enabled
- 1 => Reports are disabled

Definition at line 15 of file `cl_syoscbs_cfg.svh`.

Referenced by `get_disable_report()`.

13.110.3.2 `print_cfg`

```
bit cl_syoscbs_cfg::print_cfg = 0b1 [private]
```

Whether to print scoreboard wrapper configuration in the `UVM build_phase`.

- 0 => Disable print of scb wrapper configuration
- 1 => Enable print of scb wrapper configuration

Definition at line 23 of file `cl_syoscbs_cfg.svh`.

Referenced by `get_print_cfg()`.

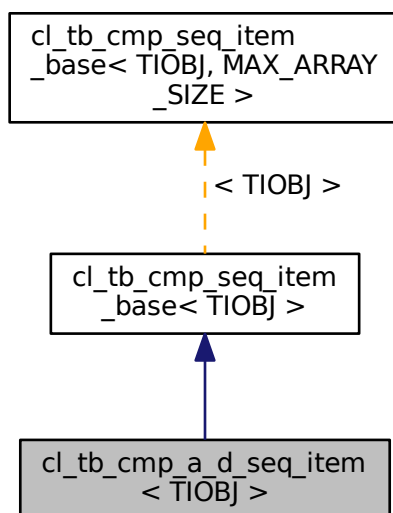
The documentation for this class was generated from the following files:

- `cl_syoscbs_cfg.svh`
- `pk_syoscb.sv`

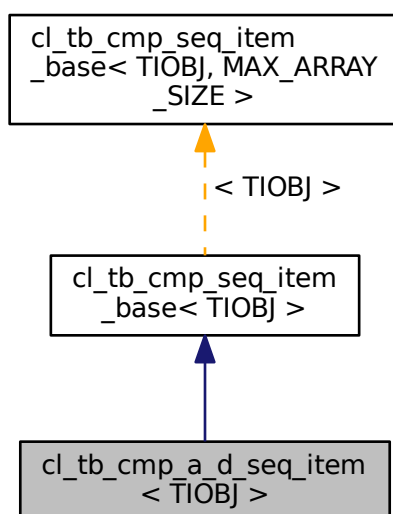
13.111 cl_tb_cmp_a_d_seq_item< TIOBJ > Class Template Reference

An "a" type item which used a manual do_compare implementation instead of field macros.

Inheritance diagram for cl_tb_cmp_a_d_seq_item< TIOBJ >:



Collaboration diagram for cl_tb_cmp_a_d_seq_item< TIOBJ >:



13.111.1 Detailed Description

```
template<typename TIOBJ = cl_tb_seq_item>
class cl_tb_cmp_a_d_seq_item< TIOBJ >
```

An "a" type item which used a manual do_compare implementation instead of field macros.

Definition at line 2 of file cl_tb_cmp_a_d_seq_item.svh.

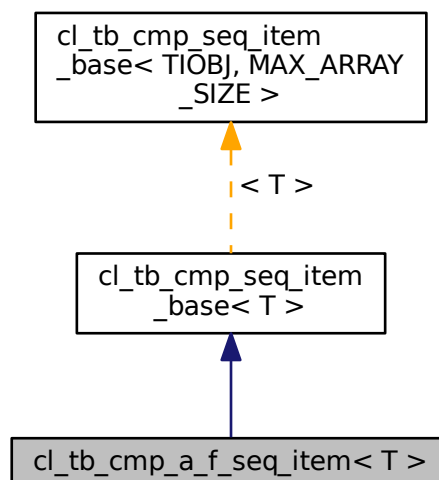
The documentation for this class was generated from the following file:

- cl_tb_cmp_a_d_seq_item.svh

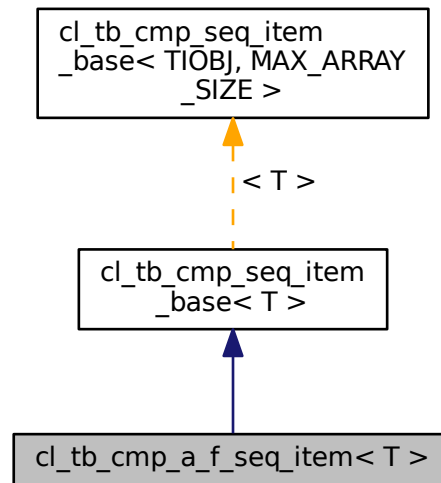
13.112 cl_tb_cmp_a_f_seq_item< T > Class Template Reference

An "a" type item which used a field macros instead of manually implementing do_compare.

Inheritance diagram for cl_tb_cmp_a_f_seq_item< T >:



Collaboration diagram for cl_tb_cmp_a_f_seq_item< T >:



13.112.1 Detailed Description

```
template<typename T = cl_tb_seq_item>
class cl_tb_cmp_a_f_seq_item< T >
```

An "a" type item which used a field macros instead of manually implementing do_compare.

Definition at line 2 of file cl_tb_cmp_a_f_seq_item.svh.

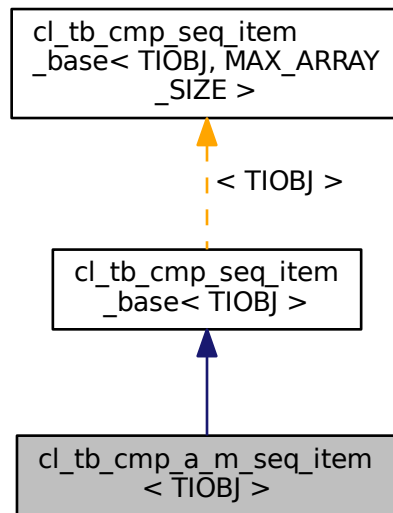
The documentation for this class was generated from the following file:

- cl_tb_cmp_a_f_seq_item.svh

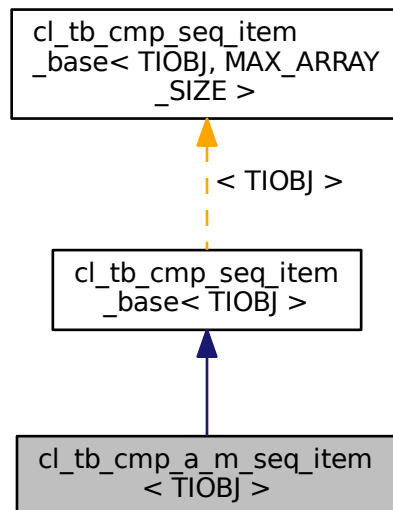
13.113 cl_tb_cmp_a_m_seq_item< TIOBJ > Class Template Reference

A "b" type item which used a mix of do_compare implementation and field macros.

Inheritance diagram for `cl_tb_cmp_a_m_seq_item< TIOBJ >`:



Collaboration diagram for `cl_tb_cmp_a_m_seq_item< TIOBJ >`:



13.113.1 Detailed Description


```
template<typename TIOBJ = cl_tb_seq_item>
class cl_tb_cmp_a_m_seq_item< TIOBJ >
```

A "b" type item which used a mix of do_compare implementation and field macros.

Definition at line 2 of file cl_tb_cmp_b_m_seq_item.svh.

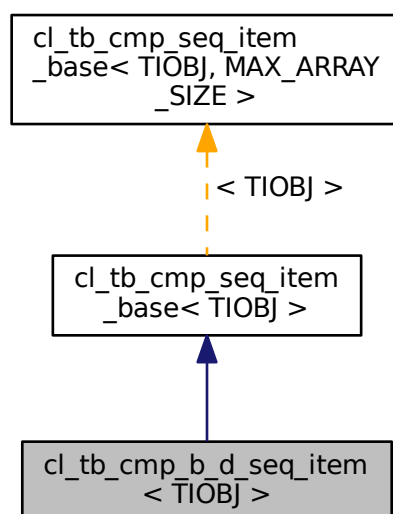
The documentation for this class was generated from the following file:

- cl_tb_cmp_b_m_seq_item.svh

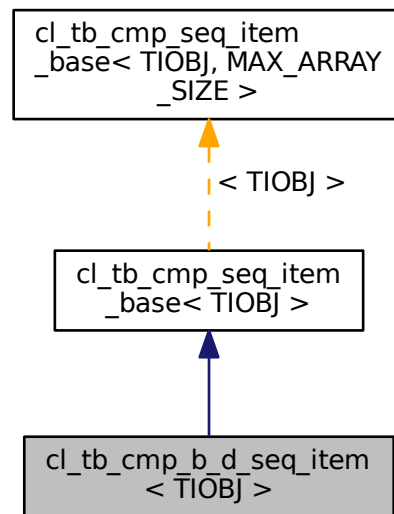
13.114 cl_tb_cmp_b_d_seq_item< TIOBJ > Class Template Reference

A "b" type item which used a manual do_compare implementation instead of field macros.

Inheritance diagram for cl_tb_cmp_b_d_seq_item< TIOBJ >:



Collaboration diagram for `cl_tb_cmp_b_d_seq_item< TIOBJ >`:



13.114.1 Detailed Description

```

template<typename TIOBJ = cl_tb_seq_item>
class cl_tb_cmp_b_d_seq_item< TIOBJ >

```

A "b" type item which used a manual `do_compare` implementation instead of field macros.

Definition at line 2 of file `cl_tb_cmp_b_d_seq_item.svh`.

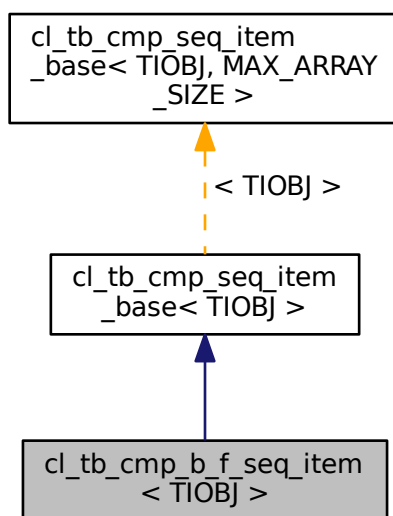
The documentation for this class was generated from the following file:

- `cl_tb_cmp_b_d_seq_item.svh`

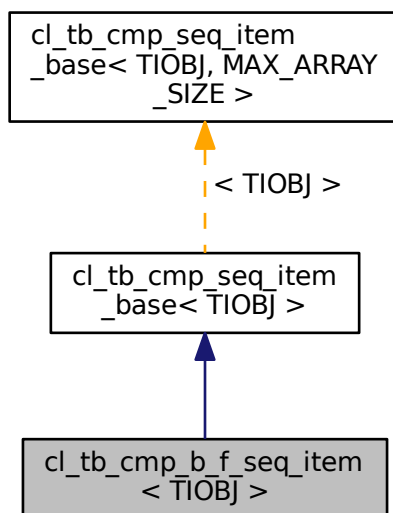
13.115 `cl_tb_cmp_b_f_seq_item< TIOBJ >` Class Template Reference

A "b" type item which used a field macros instead of manually implementing `do_compare`.

Inheritance diagram for cl_tb_cmp_b_f_seq_item< TIOBJ >:



Collaboration diagram for cl_tb_cmp_b_f_seq_item< TIOBJ >:



13.115.1 Detailed Description

```
template<typename TIOBJ = cl_tb_seq_item>
class cl_tb_cmp_b_f_seq_item< TIOBJ >
```

A "b" type item which used a field macros instead of manually implementing do_compare.

Definition at line 2 of file cl_tb_cmp_b_f_seq_item.svh.

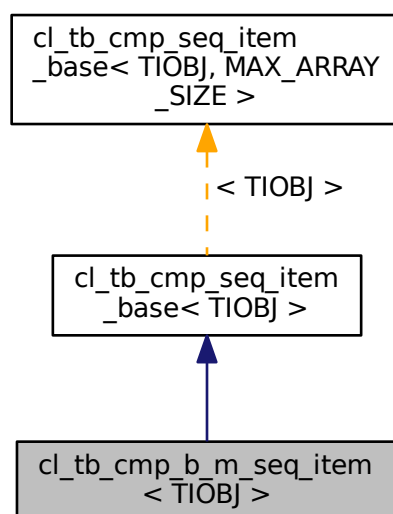
The documentation for this class was generated from the following file:

- cl_tb_cmp_b_f_seq_item.svh

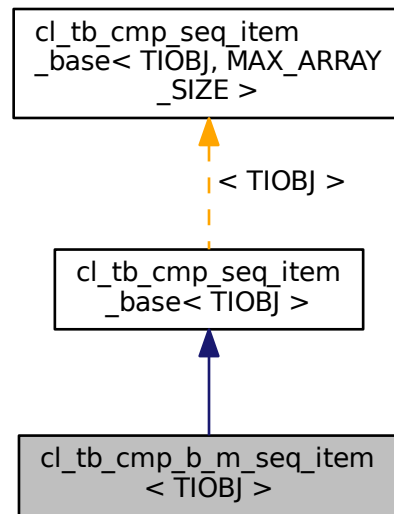
13.116 cl_tb_cmp_b_m_seq_item< TIOBJ > Class Template Reference

An "a" type item which used a mix of do_compare implementation and field macros.

Inheritance diagram for cl_tb_cmp_b_m_seq_item< TIOBJ >:



Collaboration diagram for cl_tb_cmp_b_m_seq_item< TIOBJ >:



13.116.1 Detailed Description

```
template<typename TIOBJ = cl_tb_seq_item>
class cl_tb_cmp_b_m_seq_item< TIOBJ >
```

An "a" type item which used a mix of `do_compare` implementation and field macros.

Definition at line 2 of file `cl_tb_cmp_a_m_seq_item.svh`.

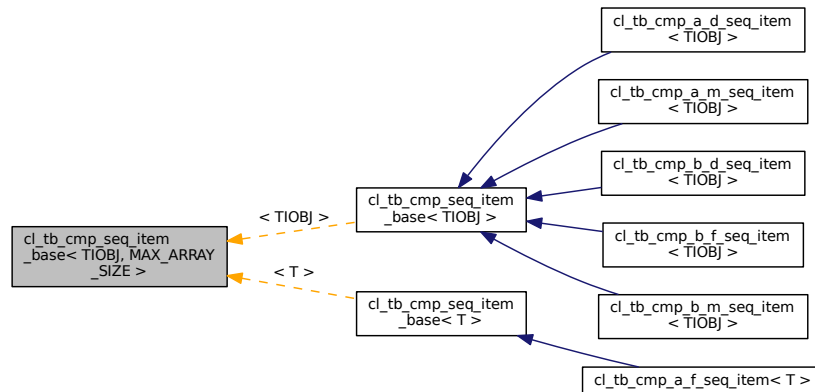
The documentation for this class was generated from the following file:

- `cl_tb_cmp_a_m_seq_item.svh`

13.117 cl_tb_cmp_seq_item_base< TIOBJ, MAX_ARRAY_SIZE > Class Template Reference

A sequence item to be used in cmp-tests extending from [cl_scb_test_cmp_base](#).

Inheritance diagram for `cl_tb_cmp_seq_item_base< TIOBJ, MAX_ARRAY_SIZE >`:



13.117.1 Detailed Description

```
template<typename TIOBJ = cl_tb_seq_item, int unsigned MAX_ARRAY_SIZE = 5>
class cl_tb_cmp_seq_item_base< TIOBJ, MAX_ARRAY_SIZE >
```

A sequence item to be used in cmp-tests extending from [cl_scb_test_cmp_base](#).

Parameters

<i>TIOBJ</i>	the typename of objects that this class should contain
<i>MAX_ARRAY_SIZE</i>	The maximum size of arrays in the object

Definition at line 4 of file `cl_tb_cmp_seq_item_base.svh`.

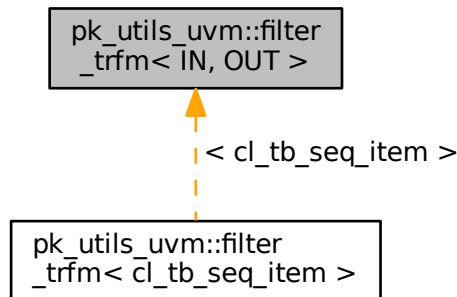
The documentation for this class was generated from the following file:

- `cl_tb_cmp_seq_item_base.svh`

13.118 `pk_utils_uvm::filter_trfm< IN, OUT >` Class Template Reference

Base class for a filter transformation.

Inheritance diagram for pk_utils_uvm::filter_trfm< IN, OUT >:



Public Member Functions

- virtual void [transform](#) (IN t, output OUT items[])
Transforms the item of type IN to one or more items of type OUT.
- virtual bit [evaluate](#) (IN t)
Evaluates whether the input should be transformed and forwarded, or whether it should be discarded.
- void [write](#) (IN t)
This filter transform's write-implementation from uvm_subscriber When items are written to the filter transform, they are first evaluated with [evaluate](#) to decide whether a transformation should occur.

Public Attributes

- uvm_analysis_port< OUT > [ap](#)
Analysis port where transformed items are output.

13.118.1 Detailed Description

```
template<typename IN = int, typename OUT = uvm_sequence_item>
class pk_utils_uvm::filter_trfm< IN, OUT >
```

Base class for a filter transformation.

If type IN is a subtype of uvm_sequence_item, this filter transforms simply performs an identity transform by upcasting the input item to a uvm_sequence_item. If another transformation is desired, extend the class and override the [evaluate](#) and [transform](#) methods.

Parameters

<i>IN</i>	Input type of objects to transform
<i>OUT</i>	Output type of transformed objects

Definition at line 18 of file pk_utils_uvm.sv.

13.118.2 Member Function Documentation

13.118.2.1 evaluate()

```
template<typename IN = int, typename OUT = uvm_sequence_item>
virtual bit pk_utils_uvm::filter_trfm< IN, OUT >::evaluate (
    IN t ) [virtual]
```

Evaluates whether the input should be transformed and forwarded, or whether it should be discarded.

If the method returns 0, the input item is discarded and may not be retrievable

Returns

1 if the item should be transformed and forwarded, 0 otherwise.

Definition at line 49 of file pk_utils_uvm.sv.

13.118.2.2 transform()

```
template<typename IN = int, typename OUT = uvm_sequence_item>
virtual void pk_utils_uvm::filter_trfm< IN, OUT >::transform (
    IN t,
    output OUT items[] ) [virtual]
```

Transforms the item of type IN to one or more items of type OUT.

Parameters

<i>t</i>	The input object which should be transformed
<i>items</i>	A reference to an array where output items will be returned. If the handle is to an existing array, that array will be lost.

Definition at line 40 of file pk_utils_uvm.sv.

13.118.2.3 write()

```
template<typename IN = int, typename OUT = uvm_sequence_item>
void pk_utils_uvm::filter_trfm< IN, OUT >::write (
    IN t )
```


This filter transform's write-implementation from `uvm_subscriber`. When items are written to the filter transform, they are first evaluated with `evaluate` to decide whether a transformation should occur.

If true, they are transformed with `transform`, and all output items are then written out on `ap`.

Parameters

<code>t</code>	The item written to this filter transform
----------------	---

Definition at line 58 of file `pk_utils_uvm.sv`.

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

- `pk_utils_uvm.sv`

13.119 `cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >::packed` Struct Reference

Typedef for struct representing whether an option with an iterator was valid.

13.119.1 Detailed Description

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
struct cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >::packed
```

Typedef for struct representing whether an option with an iterator was valid.

Definition at line 22 of file `cl_syoscb_queue_hash.svh`.

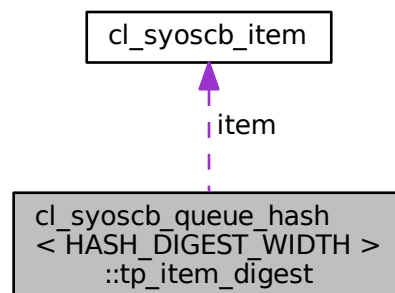
The documentation for this struct was generated from the following files:

- `cl_syoscb_queue_hash.svh`
- `pk_syoscb.sv`

13.120 `cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >::tp_item_digest` Struct Reference

Typedef for struct used to track items and their digests in the key queue.

Collaboration diagram for `cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >::tp_item_digest`:



13.120.1 Detailed Description

```
template<int unsigned HASH_DIGEST_WIDTH = 1>
struct cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >::tp_item_digest
```

Typedef for struct used to track items and their digests in the key queue.

Only used when [cl_syoscb_cfg.ordered_next=1](#)

Definition at line 10 of file [cl_syoscb_queue_hash.svh](#).

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

- [cl_syoscb_queue_hash.svh](#)
- [pk_syoscb.sv](#)

13.121 [pk_syoscb::uvm_xml_printer](#) Class Reference

An XML printer for [cl_syoscb_items](#).

Inherits [uvm_printer](#).

Public Member Functions

- virtual string [format_syoscb_item](#) (int unsigned idx)
Formats a [cl_syoscb_item](#) and all of its children.

13.121.1 Detailed Description

An XML printer for [cl_syoscb_items](#).

Definition at line 3 of file [pk_syoscb.sv](#).

13.121.2 Member Function Documentation

13.121.2.1 [format_syoscb_item\(\)](#)

```
string uvm_xml_printer::format_syoscb_item (
    int unsigned idx ) [virtual]
```

Formats a [cl_syoscb_item](#) and all of its children.

It is assumed that the [cl_syoscb_item](#) is at position `m_rows[idx]`

Parameters

<i>idx</i>	The id at which this cl_syoscb_item is placed (should always be 0)
------------	--

Returns

The XML formatted string

Definition at line 264 of file pk_syoscb.sv.

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

- pk_syoscb.sv

13.122 uvm_xml_printer Class Reference

An XML printer for [cl_syoscb_items](#).

Inherits [uvm_printer](#).

Public Member Functions

- virtual string [format_syoscb_item](#) (int unsigned idx)
Formats a [cl_syoscb_item](#) and all of its children.
- virtual int unsigned [format_object](#) (int unsigned idx, ref string result)
Formats a sequence item/object, and recursively formats all children of this seq item.
- virtual int unsigned [format_primitive](#) (int unsigned idx, ref string result)
Formats a primitive value.
- virtual int unsigned [format_array](#) (int unsigned idx, ref string result)
Formats an array and all of its children.
- virtual bit [is_primitive](#) (uvm_printer_row_info row)
Checks whether an element is a SystemVerilog primitive.
- virtual bit [is_object](#) (uvm_printer_row_info row)
Checks whether an element is an object.
- virtual bit [is_array](#) (uvm_printer_row_info row)
Checks whether an element is an array Arrays are recognized as dynamic, associative or static arrays.
- virtual void [increase_indent](#) (int unsigned steps=1)
Increases the indentation used by a set amount of steps The size of each step is controlled by knobs.indent.
- virtual void [decrease_indent](#) (int unsigned steps=1)
Decreases the indentation used by a set amount of steps The size of each step is controlled by knobs.indent.
- virtual string [get_indent](#) ()
Gets an indentation string consisting of this.indent_level spaces.

13.122.1 Detailed Description

An XML printer for [cl_syoscb_items](#).

Definition at line 2 of file uvm_xml_printer.svh.

13.122.2 Member Function Documentation

13.122.2.1 `format_array()`

```
int unsigned uvm_xml_printer::format_array (
    int unsigned idx,
    ref string result ) [virtual]
```

Formats an array and all of its children.

Parameters

<i>idx</i>	The position in <code>m_rows</code> where the array is located
<i>result</i>	The result string being built

Definition at line 351 of file `pk_syoscb.sv`.

13.122.2.2 `format_object()`

```
int unsigned uvm_xml_printer::format_object (
    int unsigned idx,
    ref string result ) [virtual]
```

Formats a sequence item/object, and recursively formats all children of this seq item.

Parameters

<i>idx</i>	The position in <code>m_rows</code> where this sequence item is located
<i>result</i>	The result string being built

Definition at line 305 of file `pk_syoscb.sv`.

Referenced by `format_syoscb_item()`.

13.122.2.3 `format_primitive()`

```
int unsigned uvm_xml_printer::format_primitive (
    int unsigned idx,
    ref string result ) [virtual]
```

Formats a primitive value.

Parameters

<i>idx</i>	The position in m_rows where the primitive is located
<i>result</i>	The result being built

Definition at line 397 of file pk_syoscb.sv.

13.122.2.4 format_syoscb_item()

```
string uvm_xml_printer::format_syoscb_item (
    int unsigned idx ) [virtual]
```

Formats a [cl_syoscb_item](#) and all of its children.

It is assumed that the [cl_syoscb_item](#) is at position m_rows[idx]

Parameters

<i>idx</i>	The id at which this cl_syoscb_item is placed (should always be 0)
------------	--

Returns

The XML formatted string

Definition at line 263 of file uvm_xml_printer.svh.

References [format_object\(\)](#).

13.122.2.5 is_array()

```
bit uvm_xml_printer::is_array (
    uvm_printer_row_info row ) [virtual]
```

Checks whether an element is an array Arrays are recognized as dynamic, associative or static arrays.

The UVM printer interprets queues as dynamic arrays

Parameters

<i>type_name</i>	The type name field of the currently parsed element
<i>value</i>	The value field of the currently parsed element

Returns

0b1 if the current element is an array, 10b0 otherwise

Definition at line 423 of file pk_syoscb.sv.

13.122.2.6 is_object()

```
bit uvm_xml_printer::is_object (
    uvm_printer_row_info row ) [virtual]
```

Checks whether an element is an object.

Parameters

<i>size</i>	The size field of the currently parsed element.
<i>val</i>	The value field of the currently parsed element

Returns

1 if the current element is an object, 0 otherwise

Definition at line 438 of file pk_syoscb.sv.

13.122.2.7 is_primitive()

```
bit uvm_xml_printer::is_primitive (
    uvm_printer_row_info row ) [virtual]
```

Checks whether an element is a SystemVerilog primitive.

Here, a "primitive" is one that maps to the UVM printer representations: integral, real, string

Parameters

<i>type_name</i>	The type name field of the currently parsed element
------------------	---

Returns

1 if the current element is a primitive, 0 otherwise

Definition at line 410 of file pk_syoscb.sv.

The documentation for this class was generated from the following files:

- uvm_xml_printer.svh
- pk_syoscb.sv

Index

- add_item
 - cl_syoscb, 106
 - cl_syoscb_hash_item, 196
 - cl_syoscb_queue_base, 218
 - cl_syoscb_queue_hash, 234
 - cl_syoscb_queue_std, 270
- add_item_mutexed
 - cl_syoscb, 107
- build_phase
 - cl_syoscb, 107
 - cl_syoscbcs, 283
 - cl_syoscbcs_base, 287
- check_first
 - cl_scb_test_iterator_unit_tests, 77
- check_last
 - cl_scb_test_iterator_unit_tests, 77
- check_names
 - cl_scb_test_iterator_unit_tests, 78
- check_next
 - cl_scb_test_iterator_unit_tests, 78
- check_phase
 - cl_syoscb, 108
 - cl_syoscb_queue_base, 219
- check_prev
 - cl_scb_test_iterator_unit_tests, 78
- check_queues
 - cl_syoscb_compare_base, 157
- check_set_queue
 - cl_scb_test_iterator_unit_tests, 79
- cl_scb_test_base, 49
- cl_scb_test_benchmark, 49
- cl_scb_test_cmp_base< ATYPE, suffix >, 50
- cl_scb_test_cmp_io< ATYPE, suffix >, 51
- cl_scb_test_cmp_ooo< ATYPE, suffix >, 52
- cl_scb_test_copy_cfg, 54
- cl_scb_test_double_scb, 55
- cl_scb_test_io_2hp_md5_simple, 56
- cl_scb_test_io_2hp_std_sbs_print, 57
- cl_scb_test_io_2hp_std_simple, 58
- cl_scb_test_io_md5_disable_compare, 59
- cl_scb_test_io_md5_dump_orphans, 59
- cl_scb_test_io_md5_simple, 60
- cl_scb_test_io_std_comparer_printer, 61
- cl_scb_test_io_std_comparer_report, 61
- cl_scb_test_io_std_disable_compare, 62
- cl_scb_test_io_std_dump, 62
- cl_scb_test_io_std_dump_default, 63
- cl_scb_test_io_std_dump_max_size, 64
- cl_scb_test_io_std_dump_max_size_less, 64
- cl_scb_test_io_std_dump_mixed, 65
- cl_scb_test_io_std_dump_simple, 66
- cl_scb_test_io_std_dump_xml_join, 67
- cl_scb_test_io_std_dump_xml_split, 68
- cl_scb_test_io_std_insert_item, 69
- cl_scb_test_io_std_insert_item_md5, 69
- cl_scb_test_io_std_intermediate_dump, 70
- cl_scb_test_io_std_sbs_print, 71
- cl_scb_test_io_std_simple, 72
- cl_scb_test_io_std_simple_mutexed, 72
- cl_scb_test_io_std_simple_real, 73
- cl_scb_test_io_std_tlm_gp_test, 73
- cl_scb_test_io_std_tlm_mutexed, 74
- cl_scb_test_iop_md5_simple, 74
- cl_scb_test_iop_std_msw, 74
- cl_scb_test_iop_std_sbs_print, 75
- cl_scb_test_iterator_correctness, 76
- cl_scb_test_iterator_unit_tests, 76
 - check_first, 77
 - check_last, 77
 - check_names, 78
 - check_next, 78
 - check_prev, 78
 - check_set_queue, 79
- cl_scb_test_iterator_unit_tests_md5, 80
- cl_scb_test_md5, 81
- cl_scb_test_md5_hash_collisions, 81
- cl_scb_test_ooo_heavy_base, 81
- cl_scb_test_ooo_io_md5_simple, 82
- cl_scb_test_ooo_io_std_simple, 83
- cl_scb_test_ooo_md5_duplets, 84
- cl_scb_test_ooo_md5_gp, 84
- cl_scb_test_ooo_md5_heavy, 85
- cl_scb_test_ooo_md5_simple, 85
- cl_scb_test_ooo_md5_tlm, 86
- cl_scb_test_ooo_md5_validate, 86
- cl_scb_test_ooo_std_dump_orphans, 87
- cl_scb_test_ooo_std_dump_orphans_abort, 87
- cl_scb_test_ooo_std_dump_orphans_xml, 88
- cl_scb_test_ooo_std_gp, 89
- cl_scb_test_ooo_std_heavy, 89
- cl_scb_test_ooo_std_max_search_window, 90
- cl_scb_test_ooo_std_primary_multiple, 90
- cl_scb_test_ooo_std_simple, 91
- cl_scb_test_ooo_std_tlm, 91
- cl_scb_test_ooo_std_tlm_filter_trfm, 92
- cl_scb_test_ooo_std_trigger_greed, 92
- cl_scb_test_queue_find_vs_search, 93

- cl_scb_test_rnd, 93
- cl_scb_test_uvm_xml_printer, 94
- cl_scb_test_uvm_xml_printer_break, 95
- cl_scbs_test_base< FIN, MON, FT >, 96
- cl_scbs_test_filter_trfm_param, 97
- cl_scbs_test_io_custom_filter_trfm, 98
- cl_scbs_test_io_std_base, 98
- cl_scbs_test_io_std_cc, 100
- cl_scbs_test_ooo_std_base, 101
- cl_scbs_test_ooo_std_flush, 102
- cl_syoscb, 104
 - add_item, 106
 - add_item_mutexed, 107
 - build_phase, 107
 - check_phase, 108
 - compare_control, 108
 - config_validation, 108
 - create_queues_stats, 109
 - create_report, 109
 - create_report_contents, 110
 - create_total_stats, 110
 - dump_join_txt, 111
 - dump_join_xml, 111
 - dump_split_txt, 111
 - dump_split_xml, 111
 - dump_txt, 112
 - dump_xml, 112
 - empty_queues, 112
 - end_of_elaboration_phase, 113
 - flush_queues, 113
 - get_failed_checks, 113
 - get_queue_failed_checks, 114
 - get_subscriber, 114
 - insert_queues, 115
 - intermediate_queue_stat_dump, 115
 - override_queue_type, 116
 - pre_abort, 116
 - print_header, 116
- cl_syoscb_cfg, 117
 - comparers, 139
 - default_comparer, 139
 - default_enable_comparer_report, 140
 - default_printer, 140
 - default_printer_verbosity, 140
 - disable_clone, 141
 - disable_compare_after_error, 141
 - disable_report, 141
 - dump_orphans_to_files, 142
 - dynamic_primary_queue, 123
 - enable_c2s_full_scb_dump, 142
 - enable_comparer_report, 142
 - enable_no_insert_check, 143
 - enable_queue_stats, 143
 - end_greediness, 143
 - exist_producer, 123
 - exist_queue, 124
 - full_scb_dump, 144
 - full_scb_dump_split, 144
 - full_scb_dump_type, 144
 - full_scb_max_queue_size, 145
 - get_comparer, 124
 - get_enable_comparer_report, 125
 - get_enable_queue_stats, 125
 - get_full_scb_max_queue_size, 126
 - get_max_queue_size, 126
 - get_max_search_window, 127
 - get_primary_queue, 127
 - get_printer, 127
 - get_printer_verbosity, 128
 - get_producer, 128
 - get_producers, 129
 - get_queue, 129
 - get_queue_stat_interval, 130
 - get_queues, 130
 - hash_compare_check, 145
 - init, 131
 - max_print_orphans, 145
 - max_queue_size, 146
 - max_search_window, 146
 - mutexed_add_item_enable, 146
 - ordered_next, 147
 - orphan_dump_type, 147
 - primary_queue, 147
 - print_cfg, 148
 - print_orphans_as_errors, 148
 - printer_verbosity, 148
 - printers, 149
 - producers, 149
 - queue_stat_interval, 149
 - scb_stat_interval, 150
 - set_comparer, 131
 - set_default_enable_comparer_report, 132
 - set_default_printer_verbosity, 132
 - set_dump_orphans_to_files, 132
 - set_enable_comparer_report, 132
 - set_enable_queue_stats, 133
 - set_full_scb_dump_split, 133
 - set_full_scb_max_queue_size, 134
 - set_max_queue_size, 134
 - set_max_search_window, 135
 - set_primary_queue, 135
 - set_printer, 136
 - set_printer_verbosity, 136
 - set_producer, 137
 - set_queue, 137
 - set_queue_stat_interval, 138
 - set_queues, 138
 - set_scb_stat_interval, 138
 - size_queues, 139
 - trigger_greediness, 150
- cl_syoscb_cfg_pl, 151
 - exists, 151
- cl_syoscb_compare, 152
 - compare_control, 153
 - compare_trigger, 153
 - extract_phase, 153

- cl_syoscb_compare_base, 154
 - check_queues, 157
 - compare_control, 157
 - compare_do_greed, 157
 - compare_init, 158
 - compare_main, 158
 - compare_trigger, 159
 - count_producers, 159
 - delete, 160
 - do_split, 164
 - dynamic_queue_split_do, 160
 - generate_miscomp_table, 160
 - get_count_producer, 161
 - get_primary_queue_name, 161
 - get_queues_item_cnt, 162
 - primary_loop_do, 162
 - primary_loop_init, 162
 - secondary_item_found, 164
 - secondary_loop_do, 163
 - set_cfg, 163
 - split_queues, 163
 - static_queue_split_do, 164
- cl_syoscb_compare_io, 165
 - count_producers, 166, 167
 - primary_loop_do, 167
 - secondary_loop_do, 168
- cl_syoscb_compare_io_2hp, 169
 - compare_do, 170
 - primary_loop_do, 170, 171
- cl_syoscb_compare_iop, 172
 - compare_init, 173
 - get_count_producer, 174
 - primary_loop_do, 174, 175
 - secondary_loop_do, 175, 176
- cl_syoscb_compare_ooo, 177
 - get_count_producer, 178
 - primary_loop_do, 178, 179
 - secondary_loop_do, 179
- cl_syoscb_comparer_config, 180
 - copy_comparer, 181
 - do_help_pack, 181
 - do_help_unpack, 181
 - get_miscompares_from_comparer, 182
 - get_show_max, 182
 - get_verbosity, 182
 - set_show_max, 183
 - set_verbosity, 183
- cl_syoscb_hash_aa_wrapper
 - delete, 185
 - exists, 186
 - first, 186
 - get_hash_item, 187
 - get_item, 187
 - get_size, 188
 - insert, 188
 - last, 189
 - next, 189
 - prev, 190
 - size, 190
- cl_syoscb_hash_aa_wrapper< HASH_DIGEST_WIDTH, TH >, 184
- cl_syoscb_hash_base
 - do_hash, 193
 - hash, 193
 - hash_str, 194
 - packer, 194
- cl_syoscb_hash_base< HASH_DIGEST_WIDTH >, 191
- cl_syoscb_hash_item, 195
 - add_item, 196
 - delete_item, 197
 - get_item, 197
- cl_syoscb_hash_md5, 198
 - do_hash, 199, 200
- cl_syoscb_hash_packer, 200
- cl_syoscb_item, 201
 - convert2string, 202
 - queue_index, 203
 - set_producer, 203
- cl_syoscb_md5_packer, 204
- cl_syoscb_printer_config, 205
 - copy_printer, 206
 - do_help_pack, 206
 - do_help_unpack, 207
 - get_file_descriptor, 207
 - get_printer_of_type, 208
 - get_printer_type, 208
 - set_file_descriptor, 208
 - set_printer_begin_elements, 209
 - set_printer_end_elements, 209
- cl_syoscb_proxy_item_base, 210
 - get_item, 211
 - get_queue, 211
 - set_queue, 211
- cl_syoscb_proxy_item_hash
 - idx, 213
- cl_syoscb_proxy_item_hash< HASH_DIGEST_WIDTH, TH >, 212
- cl_syoscb_proxy_item_std, 214
- cl_syoscb_queue_base, 215
 - add_item, 218
 - check_phase, 219
 - create_iterator, 219
 - create_producer_stats, 220
 - create_queue_report, 220
 - decr_cnt_producer, 221
 - delete_item, 221
 - delete_iterator, 222
 - dump, 222
 - dump_orphans_to_file, 223
 - dump_orphans_to_stdout, 223
 - empty, 224
 - exists_cnt_producer, 224
 - failed_checks, 231
 - flush_queue, 225
 - get_cnt_producer, 225

- get_dump_extension, 225
- get_failed_checks, 226
- get_item, 226
- get_iterator, 227
- get_locator, 227
- get_size, 228
- incr_cnt_producer, 228
- insert_item, 229
- post_add_item, 229
- pre_add_item, 230
- print_orphan_xml_footer, 230
- print_orphan_xml_header, 231
- cl_syoscb_queue_hash
 - add_item, 234
 - delete_item, 235
 - delete_iterator, 236
 - empty, 236
 - get_item, 236
 - get_key_queue, 237
 - get_size, 237
 - insert_item, 238
 - key_queue, 239
- cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >, 232
- cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >↔
 - ::packed, 319
- cl_syoscb_queue_hash< HASH_DIGEST_WIDTH >↔
 - ::tp_item_digest, 319
- cl_syoscb_queue_hash_md5, 239
 - create_iterator, 240
 - get_locator, 241
- cl_syoscb_queue_iterator_base, 241
 - first, 243
 - get_item_proxy, 243
 - get_queue, 244
 - has_next, 244
 - has_previous, 244
 - last, 245
 - next, 245
 - next_index, 245
 - previous, 246
 - previous_index, 246
 - set_queue, 246
- cl_syoscb_queue_iterator_hash
 - first, 250
 - get_item_proxy, 250
 - has_next, 250
 - has_previous, 251
 - last, 251
 - next, 251
 - previous, 252
 - set_queue, 252
- cl_syoscb_queue_iterator_hash< HASH_DIGEST_WIDTH >↔
 - IDTH >, 247
- cl_syoscb_queue_iterator_hash_md5, 253
- cl_syoscb_queue_iterator_std, 254
 - first, 256
 - get_item_proxy, 256
 - has_next, 256
 - has_previous, 256
 - last, 257
 - next, 257
 - previous, 257
 - set_queue, 258
- cl_syoscb_queue_locator_base, 258
 - search, 259
- cl_syoscb_queue_locator_hash
 - search, 263
 - validate_match, 263
 - validate_no_match, 263
- cl_syoscb_queue_locator_hash< HASH_DIGEST_WIDTH >↔
 - IDTH >, 260
- cl_syoscb_queue_locator_hash_md5, 264
- cl_syoscb_queue_locator_std, 265
 - compare_items, 267
 - search, 267
- cl_syoscb_queue_std, 268
 - add_item, 270
 - create_iterator, 270
 - delete_item, 271
 - delete_iterator, 271
 - empty, 272
 - get_item, 272
 - get_locator, 273
 - get_size, 273
 - insert_item, 273
- cl_syoscb_string_library, 274
 - generate_cmp_table_body, 275
 - generate_cmp_table_footer, 275
 - generate_cmp_table_header, 276
 - merge_string_arrays, 276
 - pad_str, 277
 - scb_header_str, 277
 - scb_separator_str, 278
 - split_string, 278
 - sprint_item, 279
- cl_syoscb_subscriber, 280
 - set_mutexed_add_item_enable, 280
- cl_syoscbcs
 - build_phase, 283
 - get_filter_trfm, 283, 284
- cl_syoscbcs< FIN >, 281
- cl_syoscbcs_base, 285
 - build_phase, 287
 - compare_control_all, 287
 - compare_control_by_index, 287
 - compare_control_by_name, 288
 - connect_filter_and_subscriber, 288
 - connect_filters, 289
 - create_filter, 289
 - create_filters, 290
 - create_report, 290
 - create_scb_stats, 291
 - create_total_stats, 291
 - do_print, 292
 - flush_queues_by_index, 292

- flush_queues_by_name, 293
- fts, 295
- get_filter_trfm_base, 293
- get_scb, 294
- get_scb_failed_checks, 294
- report_phase, 294
- cl_syoscbs_cfg, 295
 - disable_report, 306
 - get_cfg, 297
 - get_queues, 298
 - get_scb_end_greediness, 298
 - get_scb_index_by_name, 299
 - get_scb_names, 299
 - get_scb_trigger_greediness, 299
 - init, 300
 - is_scb_names_unique, 300
 - print_cfg, 306
 - set_cfg, 301
 - set_compare_type, 301
 - set_enable_scb_stats, 302
 - set_no_scbs, 302
 - set_producers, 302
 - set_queue_type, 304
 - set_queues, 304
 - set_scb_end_greediness, 305
 - set_scb_names, 305
 - set_scb_trigger_greediness, 306
- cl_tb_cmp_a_d_seq_item< TIOBJ >, 307
- cl_tb_cmp_a_f_seq_item< T >, 308
- cl_tb_cmp_a_m_seq_item< TIOBJ >, 309
- cl_tb_cmp_b_d_seq_item< TIOBJ >, 311
- cl_tb_cmp_b_f_seq_item< TIOBJ >, 312
- cl_tb_cmp_b_m_seq_item< TIOBJ >, 314
- cl_tb_cmp_seq_item_base< TIOBJ, MAX_ARRAY_SIZE >, 315
- compare_control
 - cl_syoscb, 108
 - cl_syoscb_compare, 153
 - cl_syoscb_compare_base, 157
- compare_control_all
 - cl_syoscbs_base, 287
- compare_control_by_index
 - cl_syoscbs_base, 287
- compare_control_by_name
 - cl_syoscbs_base, 288
- compare_do
 - cl_syoscb_compare_io_2hp, 170
- compare_do_greed
 - cl_syoscb_compare_base, 157
- compare_init
 - cl_syoscb_compare_base, 158
 - cl_syoscb_compare_iop, 173
- compare_items
 - cl_syoscb_queue_locator_std, 267
- compare_main
 - cl_syoscb_compare_base, 158
- compare_trigger
 - cl_syoscb_compare, 153
- cl_syoscb_compare_base, 159
- comparers
 - cl_syoscb_cfg, 139
- config_validation
 - cl_syoscb, 108
- connect_filter_and_subscriber
 - cl_syoscbs_base, 288
- connect_filters
 - cl_syoscbs_base, 289
- convert2string
 - cl_syoscb_item, 202
- copy_comparer
 - cl_syoscb_comparer_config, 181
- copy_printer
 - cl_syoscb_printer_config, 206
- count_producers
 - cl_syoscb_compare_base, 159
 - cl_syoscb_compare_io, 166, 167
- create_filter
 - cl_syoscbs_base, 289
- create_filters
 - cl_syoscbs_base, 290
- create_iterator
 - cl_syoscb_queue_base, 219
 - cl_syoscb_queue_hash_md5, 240
 - cl_syoscb_queue_std, 270
- create_producer_stats
 - cl_syoscb_queue_base, 220
- create_queue_report
 - cl_syoscb_queue_base, 220
- create_queues_stats
 - cl_syoscb, 109
- create_report
 - cl_syoscb, 109
 - cl_syoscbs_base, 290
- create_report_contents
 - cl_syoscb, 110
- create_scb_stats
 - cl_syoscbs_base, 291
- create_total_stats
 - cl_syoscb, 110
 - cl_syoscbs_base, 291
- decr_cnt_producer
 - cl_syoscb_queue_base, 221
- default_comparer
 - cl_syoscb_cfg, 139
- default_enable_comparer_report
 - cl_syoscb_cfg, 140
- default_printer
 - cl_syoscb_cfg, 140
- default_printer_verbosity
 - cl_syoscb_cfg, 140
- delete
 - cl_syoscb_compare_base, 160
 - cl_syoscb_hash_aa_wrapper, 185
- delete_item
 - cl_syoscb_hash_item, 197
 - cl_syoscb_queue_base, 221

- cl_syoscb_queue_hash, 235
 - cl_syoscb_queue_std, 271
- delete_iterator
 - cl_syoscb_queue_base, 222
 - cl_syoscb_queue_hash, 236
 - cl_syoscb_queue_std, 271
- disable_clone
 - cl_syoscb_cfg, 141
- disable_compare_after_error
 - cl_syoscb_cfg, 141
- disable_report
 - cl_syoscb_cfg, 141
 - cl_syoscb_cfg, 306
- do_hash
 - cl_syoscb_hash_base, 193
 - cl_syoscb_hash_md5, 199, 200
- do_help_pack
 - cl_syoscb_comparer_config, 181
 - cl_syoscb_printer_config, 206
- do_help_unpack
 - cl_syoscb_comparer_config, 181
 - cl_syoscb_printer_config, 207
- do_print
 - cl_syoscb_base, 292
- do_split
 - cl_syoscb_compare_base, 164
- dump
 - cl_syoscb_queue_base, 222
- dump_join_txt
 - cl_syoscb, 111
- dump_join_xml
 - cl_syoscb, 111
- dump_orphans_to_file
 - cl_syoscb_queue_base, 223
- dump_orphans_to_files
 - cl_syoscb_cfg, 142
- dump_orphans_to_stdout
 - cl_syoscb_queue_base, 223
- dump_split_txt
 - cl_syoscb, 111
- dump_split_xml
 - cl_syoscb, 111
- dump_txt
 - cl_syoscb, 112
- dump_xml
 - cl_syoscb, 112
- dynamic_primary_queue
 - cl_syoscb_cfg, 123
- dynamic_queue_split_do
 - cl_syoscb_compare_base, 160
- empty
 - cl_syoscb_queue_base, 224
 - cl_syoscb_queue_hash, 236
 - cl_syoscb_queue_std, 272
- empty_queues
 - cl_syoscb, 112
- enable_c2s_full_scb_dump
 - cl_syoscb_cfg, 142
- enable_comparer_report
 - cl_syoscb_cfg, 142
- enable_no_insert_check
 - cl_syoscb_cfg, 143
- enable_queue_stats
 - cl_syoscb_cfg, 143
- end_greediness
 - cl_syoscb_cfg, 143
- end_of_elaboration_phase
 - cl_syoscb, 113
- evaluate
 - pk_utils_uvm::filter_trfm, 318
- exist_producer
 - cl_syoscb_cfg, 123
- exist_queue
 - cl_syoscb_cfg, 124
- exists
 - cl_syoscb_cfg_pl, 151
 - cl_syoscb_hash_aa_wrapper, 186
- exists_cnt_producer
 - cl_syoscb_queue_base, 224
- extract_phase
 - cl_syoscb_compare, 153
- failed_checks
 - cl_syoscb_queue_base, 231
- first
 - cl_syoscb_hash_aa_wrapper, 186
 - cl_syoscb_queue_iterator_base, 243
 - cl_syoscb_queue_iterator_hash, 250
 - cl_syoscb_queue_iterator_std, 256
- flush_queue
 - cl_syoscb_queue_base, 225
- flush_queues
 - cl_syoscb, 113
- flush_queues_by_index
 - cl_syoscb_base, 292
- flush_queues_by_name
 - cl_syoscb_base, 293
- format_array
 - uvm_xml_printer, 322
- format_object
 - uvm_xml_printer, 322
- format_primitive
 - uvm_xml_printer, 322
- format_syoscb_item
 - pk_syoscb::uvm_xml_printer, 320
 - uvm_xml_printer, 323
- fts
 - cl_syoscb_base, 295
- full_scb_dump
 - cl_syoscb_cfg, 144
- full_scb_dump_split
 - cl_syoscb_cfg, 144
- full_scb_dump_type
 - cl_syoscb_cfg, 144
- full_scb_max_queue_size
 - cl_syoscb_cfg, 145

- generate_cmp_table_body
 - cl_syoscb_string_library, 275
- generate_cmp_table_footer
 - cl_syoscb_string_library, 275
- generate_cmp_table_header
 - cl_syoscb_string_library, 276
- generate_miscomp_table
 - cl_syoscb_compare_base, 160
- get_cfg
 - cl_syoscb_cfg, 297
- get_cnt_producer
 - cl_syoscb_queue_base, 225
- get_comparer
 - cl_syoscb_cfg, 124
- get_count_producer
 - cl_syoscb_compare_base, 161
 - cl_syoscb_compare_iop, 174
 - cl_syoscb_compare_ooo, 178
- get_dump_extension
 - cl_syoscb_queue_base, 225
- get_enable_comparer_report
 - cl_syoscb_cfg, 125
- get_enable_queue_stats
 - cl_syoscb_cfg, 125
- get_failed_checks
 - cl_syoscb, 113
 - cl_syoscb_queue_base, 226
- get_file_descriptor
 - cl_syoscb_printer_config, 207
- get_filter_trfm
 - cl_syoscb, 283, 284
- get_filter_trfm_base
 - cl_syoscb_base, 293
- get_full_scb_max_queue_size
 - cl_syoscb_cfg, 126
- get_hash_item
 - cl_syoscb_hash_aa_wrapper, 187
- get_item
 - cl_syoscb_hash_aa_wrapper, 187
 - cl_syoscb_hash_item, 197
 - cl_syoscb_proxy_item_base, 211
 - cl_syoscb_queue_base, 226
 - cl_syoscb_queue_hash, 236
 - cl_syoscb_queue_std, 272
- get_item_proxy
 - cl_syoscb_queue_iterator_base, 243
 - cl_syoscb_queue_iterator_hash, 250
 - cl_syoscb_queue_iterator_std, 256
- get_iterator
 - cl_syoscb_queue_base, 227
- get_key_queue
 - cl_syoscb_queue_hash, 237
- get_locator
 - cl_syoscb_queue_base, 227
 - cl_syoscb_queue_hash_md5, 241
 - cl_syoscb_queue_std, 273
- get_max_queue_size
 - cl_syoscb_cfg, 126
- get_max_search_window
 - cl_syoscb_cfg, 127
- get_miscompares_from_comparer
 - cl_syoscb_comparer_config, 182
- get_primary_queue
 - cl_syoscb_cfg, 127
- get_primary_queue_name
 - cl_syoscb_compare_base, 161
- get_printer
 - cl_syoscb_cfg, 127
- get_printer_of_type
 - cl_syoscb_printer_config, 208
- get_printer_type
 - cl_syoscb_printer_config, 208
- get_printer_verbosity
 - cl_syoscb_cfg, 128
- get_producer
 - cl_syoscb_cfg, 128
- get_producers
 - cl_syoscb_cfg, 129
- get_queue
 - cl_syoscb_cfg, 129
 - cl_syoscb_proxy_item_base, 211
 - cl_syoscb_queue_iterator_base, 244
- get_queue_failed_checks
 - cl_syoscb, 114
- get_queue_stat_interval
 - cl_syoscb_cfg, 130
- get_queues
 - cl_syoscb_cfg, 130
 - cl_syoscb_cfg, 298
- get_queues_item_cnt
 - cl_syoscb_compare_base, 162
- get_scb
 - cl_syoscb_base, 294
- get_scb_end_greediness
 - cl_syoscb_cfg, 298
- get_scb_failed_checks
 - cl_syoscb_base, 294
- get_scb_index_by_name
 - cl_syoscb_cfg, 299
- get_scb_names
 - cl_syoscb_cfg, 299
- get_scb_trigger_greediness
 - cl_syoscb_cfg, 299
- get_show_max
 - cl_syoscb_comparer_config, 182
- get_size
 - cl_syoscb_hash_aa_wrapper, 188
 - cl_syoscb_queue_base, 228
 - cl_syoscb_queue_hash, 237
 - cl_syoscb_queue_std, 273
- get_subscriber
 - cl_syoscb, 114
- get_verbosity
 - cl_syoscb_comparer_config, 182
- has_next
 - cl_syoscb_queue_iterator_base, 244

- cl_syoscb_queue_iterator_hash, 250
 - cl_syoscb_queue_iterator_std, 256
- has_previous
 - cl_syoscb_queue_iterator_base, 244
 - cl_syoscb_queue_iterator_hash, 251
 - cl_syoscb_queue_iterator_std, 256
- hash
 - cl_syoscb_hash_base, 193
- hash_compare_check
 - cl_syoscb_cfg, 145
- hash_str
 - cl_syoscb_hash_base, 194
- idx
 - cl_syoscb_proxy_item_hash, 213
- incr_cnt_producer
 - cl_syoscb_queue_base, 228
- init
 - cl_syoscb_cfg, 131
 - cl_syoscb_cfg, 300
- insert
 - cl_syoscb_hash_aa_wrapper, 188
- insert_item
 - cl_syoscb_queue_base, 229
 - cl_syoscb_queue_hash, 238
 - cl_syoscb_queue_std, 273
- insert_queues
 - cl_syoscb, 115
- intermediate_queue_stat_dump
 - cl_syoscb, 115
- is_array
 - uvm_xml_printer, 323
- is_object
 - uvm_xml_printer, 324
- is_primitive
 - uvm_xml_printer, 324
- is_scb_names_unique
 - cl_syoscb_cfg, 300
- key_queue
 - cl_syoscb_queue_hash, 239
- last
 - cl_syoscb_hash_aa_wrapper, 189
 - cl_syoscb_queue_iterator_base, 245
 - cl_syoscb_queue_iterator_hash, 251
 - cl_syoscb_queue_iterator_std, 257
- max_print_orphans
 - cl_syoscb_cfg, 145
- max_queue_size
 - cl_syoscb_cfg, 146
- max_search_window
 - cl_syoscb_cfg, 146
- merge_string_arrays
 - cl_syoscb_string_library, 276
- mutexed_add_item_enable
 - cl_syoscb_cfg, 146
- next
 - cl_syoscb_hash_aa_wrapper, 189
 - cl_syoscb_queue_iterator_base, 245
 - cl_syoscb_queue_iterator_hash, 251
 - cl_syoscb_queue_iterator_std, 257
- next_index
 - cl_syoscb_queue_iterator_base, 245
- ordered_next
 - cl_syoscb_cfg, 147
- orphan_dump_type
 - cl_syoscb_cfg, 147
- override_queue_type
 - cl_syoscb, 116
- packer
 - cl_syoscb_hash_base, 194
- pad_str
 - cl_syoscb_string_library, 277
- pk_syoscb::uvm_xml_printer, 320
 - format_syoscb_item, 320
- pk_utils_uvm::filter_trfm
 - evaluate, 318
 - transform, 318
 - write, 318
- pk_utils_uvm::filter_trfm< IN, OUT >, 316
- post_add_item
 - cl_syoscb_queue_base, 229
- pre_abort
 - cl_syoscb, 116
- pre_add_item
 - cl_syoscb_queue_base, 230
- prev
 - cl_syoscb_hash_aa_wrapper, 190
- previous
 - cl_syoscb_queue_iterator_base, 246
 - cl_syoscb_queue_iterator_hash, 252
 - cl_syoscb_queue_iterator_std, 257
- previous_index
 - cl_syoscb_queue_iterator_base, 246
- primary_loop_do
 - cl_syoscb_compare_base, 162
 - cl_syoscb_compare_io, 167
 - cl_syoscb_compare_io_2hp, 170, 171
 - cl_syoscb_compare_iop, 174, 175
 - cl_syoscb_compare_ooo, 178, 179
- primary_loop_init
 - cl_syoscb_compare_base, 162
- primary_queue
 - cl_syoscb_cfg, 147
- print_cfg
 - cl_syoscb_cfg, 148
 - cl_syoscb_cfg, 306
- print_header
 - cl_syoscb, 116
- print_orphan_xml_footer
 - cl_syoscb_queue_base, 230
- print_orphan_xml_header
 - cl_syoscb_queue_base, 231
- print_orphans_as_errors

- cl_syoscb_cfg, 148
- printer_verbosity
 - cl_syoscb_cfg, 148
- printers
 - cl_syoscb_cfg, 149
- producers
 - cl_syoscb_cfg, 149
- queue_index
 - cl_syoscb_item, 203
- queue_stat_interval
 - cl_syoscb_cfg, 149
- report_phase
 - cl_syoscbcs_base, 294
- scb_header_str
 - cl_syoscb_string_library, 277
- scb_separator_str
 - cl_syoscb_string_library, 278
- scb_stat_interval
 - cl_syoscb_cfg, 150
- search
 - cl_syoscb_queue_locator_base, 259
 - cl_syoscb_queue_locator_hash, 263
 - cl_syoscb_queue_locator_std, 267
- secondary_item_found
 - cl_syoscb_compare_base, 164
- secondary_loop_do
 - cl_syoscb_compare_base, 163
 - cl_syoscb_compare_io, 168
 - cl_syoscb_compare_iop, 175, 176
 - cl_syoscb_compare_ooo, 179
- set_cfg
 - cl_syoscb_compare_base, 163
 - cl_syoscbcs_cfg, 301
- set_compare_type
 - cl_syoscbcs_cfg, 301
- set_comparer
 - cl_syoscb_cfg, 131
- set_default_enable_comparer_report
 - cl_syoscb_cfg, 132
- set_default_printer_verbosity
 - cl_syoscb_cfg, 132
- set_dump_orphans_to_files
 - cl_syoscb_cfg, 132
- set_enable_comparer_report
 - cl_syoscb_cfg, 132
- set_enable_queue_stats
 - cl_syoscb_cfg, 133
- set_enable_scb_stats
 - cl_syoscbcs_cfg, 302
- set_file_descriptor
 - cl_syoscb_printer_config, 208
- set_full_scb_dump_split
 - cl_syoscb_cfg, 133
- set_full_scb_max_queue_size
 - cl_syoscb_cfg, 134
- set_max_queue_size
 - cl_syoscb_cfg, 134
- set_max_search_window
 - cl_syoscb_cfg, 135
- set_mutexed_add_item_enable
 - cl_syoscb_subscriber, 280
- set_no_scbs
 - cl_syoscbcs_cfg, 302
- set_primary_queue
 - cl_syoscb_cfg, 135
- set_printer
 - cl_syoscb_cfg, 136
- set_printer_begin_elements
 - cl_syoscb_printer_config, 209
- set_printer_end_elements
 - cl_syoscb_printer_config, 209
- set_printer_verbosity
 - cl_syoscb_cfg, 136
- set_producer
 - cl_syoscb_cfg, 137
 - cl_syoscb_item, 203
- set_producers
 - cl_syoscbcs_cfg, 302
- set_queue
 - cl_syoscb_cfg, 137
 - cl_syoscb_proxy_item_base, 211
 - cl_syoscb_queue_iterator_base, 246
 - cl_syoscb_queue_iterator_hash, 252
 - cl_syoscb_queue_iterator_std, 258
- set_queue_stat_interval
 - cl_syoscb_cfg, 138
- set_queue_type
 - cl_syoscbcs_cfg, 304
- set_queues
 - cl_syoscb_cfg, 138
 - cl_syoscbcs_cfg, 304
- set_scb_end_greediness
 - cl_syoscbcs_cfg, 305
- set_scb_names
 - cl_syoscbcs_cfg, 305
- set_scb_stat_interval
 - cl_syoscb_cfg, 138
- set_scb_trigger_greediness
 - cl_syoscbcs_cfg, 306
- set_show_max
 - cl_syoscb_comparer_config, 183
- set_verbosity
 - cl_syoscb_comparer_config, 183
- size
 - cl_syoscb_hash_aa_wrapper, 190
- size_queues
 - cl_syoscb_cfg, 139
- split_queues
 - cl_syoscb_compare_base, 163
- split_string
 - cl_syoscb_string_library, 278
- sprint_item
 - cl_syoscb_string_library, 279
- static_queue_split_do

- [cl_syoscb_compare_base](#), [164](#)
- transform
 - [pk_utils_uvm::filter_trfm](#), [318](#)
- trigger_greediness
 - [cl_syoscb_cfg](#), [150](#)
- uvm_xml_printer, [321](#)
 - [format_array](#), [322](#)
 - [format_object](#), [322](#)
 - [format_primitive](#), [322](#)
 - [format_syoscb_item](#), [323](#)
 - [is_array](#), [323](#)
 - [is_object](#), [324](#)
 - [is_primitive](#), [324](#)
- validate_match
 - [cl_syoscb_queue_locator_hash](#), [263](#)
- validate_no_match
 - [cl_syoscb_queue_locator_hash](#), [263](#)
- write
 - [pk_utils_uvm::filter_trfm](#), [318](#)