

artdaq  
3.13.00

Generated by Doxygen 1.8.5

Thu Sep 5 2024 11:33:40



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Namespace List . . . . .	1
<b>2</b>	<b>Hierarchical Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>9</b>
3.1	Class List . . . . .	9
<b>4</b>	<b>File Index</b>	<b>19</b>
4.1	File List . . . . .	19
<b>5</b>	<b>Namespace Documentation</b>	<b>23</b>
5.1	anonymous_namespace{genToArt.cc} Namespace Reference . . . . .	23
5.1.1	Function Documentation . . . . .	23
5.1.1.1	process_cmd_line . . . . .	23
5.1.1.2	process_data . . . . .	23
5.2	anonymous_namespace{ListenTransferWrapper.cc} Namespace Reference . . . . .	24
5.3	anonymous_namespace{RootDAQOut_module.cc} Namespace Reference . . . . .	24
5.4	anonymous_namespace{RootDAQOutFile.cc} Namespace Reference . . . . .	24
5.5	anonymous_namespace{TransferWrapper.cc} Namespace Reference . . . . .	25
5.6	anonymous_namespace{xmlrpc_commander.cc} Namespace Reference . . . . .	25
5.7	art Namespace Reference . . . . .	25
5.7.1	Detailed Description . . . . .	27
5.7.2	Typedef Documentation . . . . .	27
5.7.2.1	TransferInput . . . . .	27
5.7.3	Function Documentation . . . . .	27
5.7.3.1	printProcessHistoryID . . . . .	27
5.7.3.2	printProcessMap . . . . .	27
5.7.3.3	ReadObjectAny . . . . .	28

5.8	artdaq Namespace Reference	28
5.8.1	Detailed Description	34
5.8.2	Typedef Documentation	34
5.8.2.1	makeFunc_t	34
5.8.3	Function Documentation	34
5.8.3.1	cmd_::getParam< art::RunID >	34
5.8.3.2	cmd_::getParam< fhicl::ParameterSet >	35
5.8.3.3	cmd_::getParam< std::string >	36
5.8.3.4	exception_msg	36
5.8.3.5	exception_msg	36
5.8.3.6	exception_msg	37
5.8.3.7	exception_msg	37
5.8.3.8	exception_msg	37
5.8.3.9	exception_msg	38
5.8.3.10	makeCommandableFragmentGenerator	39
5.8.3.11	MakeCommanderPlugin	39
5.8.3.12	MakeHostMap	39
5.8.3.13	MakeHostMapPset	40
5.8.3.14	makeRoutingManagerPolicy	40
5.8.3.15	MakeTransferPlugin	40
5.9	artdaq::detail Namespace Reference	41
5.9.1	Detailed Description	42
5.9.2	Enumeration Type Documentation	42
5.9.2.1	RequestMessageMode	42
5.9.2.2	RoutingManagerMode	42
5.9.3	Function Documentation	43
5.9.3.1	IntToTaskType	43
5.9.3.2	operator<<	44
5.9.3.3	StringToTaskType	44
5.9.3.4	TaskTypeToString	44
<b>6</b>	<b>Class Documentation</b>	<b>45</b>
6.1	artdaq::add_config_archive_entry_ Class Reference	45
6.1.1	Detailed Description	45
6.1.2	Constructor & Destructor Documentation	45
6.1.2.1	add_config_archive_entry_	45
6.2	art::AnalyzersConfig Struct Reference	46

6.2.1	Detailed Description	46
6.3	FragCounter_test::Apply Struct Reference	46
6.3.1	Detailed Description	46
6.4	FragCounter_test::Apply_id Struct Reference	47
6.4.1	Detailed Description	47
6.5	FragCounter_test::ApplyWithOffset Struct Reference	47
6.5.1	Detailed Description	47
6.6	FragCounter_test::ApplyWithOffset_id Struct Reference	47
6.6.1	Detailed Description	47
6.7	artdaq::art_config_file Class Reference	47
6.7.1	Detailed Description	48
6.7.2	Constructor & Destructor Documentation	48
6.7.2.1	art_config_file	48
6.7.3	Member Function Documentation	48
6.7.3.1	getFileName	48
6.8	artdaq::artdaqapp Class Reference	48
6.8.1	Detailed Description	49
6.8.2	Member Function Documentation	49
6.8.2.1	runArtdaqApp	49
6.9	ArtdaqFragmentNamingService Class Reference	49
6.9.1	Detailed Description	50
6.9.2	Constructor & Destructor Documentation	50
6.9.2.1	ArtdaqFragmentNamingService	50
6.10	ArtdaqFragmentNamingServiceInterface Class Reference	50
6.10.1	Detailed Description	51
6.10.2	Constructor & Destructor Documentation	51
6.10.2.1	ArtdaqFragmentNamingServiceInterface	51
6.10.3	Member Function Documentation	51
6.10.3.1	GetUnidentifiedInstanceName	51
6.11	art::ArtdaqFragmentNamingServiceInterfaceConfig Struct Reference	52
6.11.1	Detailed Description	52
6.12	ArtdaqGlobalsService Class Reference	52
6.12.1	Detailed Description	53
6.12.2	Constructor & Destructor Documentation	54
6.12.2.1	ArtdaqGlobalsService	54
6.12.3	Member Function Documentation	54
6.12.3.1	GetEventHeader	54

6.12.3.2	GetMyId	54
6.12.3.3	GetQueueCapacity	54
6.12.3.4	GetQueueSize	54
6.12.3.5	ReceiveEvent	55
6.13	art::ArtdaqInputHelper< U > Class Template Reference	55
6.13.1	Detailed Description	56
6.13.2	Constructor & Destructor Documentation	56
6.13.2.1	ArtdaqInputHelper	56
6.13.3	Member Function Documentation	56
6.13.3.1	hasMoreData	56
6.13.3.2	operator=	56
6.13.3.3	readFile	57
6.13.3.4	readNext	58
6.14	art::ArtdaqOutput Class Reference	58
6.14.1	Detailed Description	59
6.14.2	Constructor & Destructor Documentation	60
6.14.2.1	ArtdaqOutput	60
6.14.2.2	~ArtdaqOutput	61
6.14.3	Member Function Documentation	61
6.14.3.1	beginRun	61
6.14.3.2	beginRun_	61
6.14.3.3	beginSubRun	61
6.14.3.4	beginSubRun_	61
6.14.3.5	closeFile	61
6.14.3.6	endJob	62
6.14.3.7	event	62
6.14.3.8	event_	62
6.14.3.9	extractProducts_	62
6.14.3.10	openFile	62
6.14.3.11	respondToCloseInputFile	62
6.14.3.12	respondToCloseOutputFiles	62
6.14.3.13	send_init_message	63
6.14.3.14	SendMessage	63
6.14.3.15	write	63
6.14.3.16	writeDataProducts	63
6.14.3.17	writeRun	63
6.14.3.18	writeSubRun	63

6.15	ArtdaqSharedMemoryService Class Reference	64
6.15.1	Detailed Description	65
6.15.2	Constructor & Destructor Documentation	65
6.15.2.1	ArtdaqSharedMemoryService	65
6.15.3	Member Function Documentation	65
6.15.3.1	GetEventHeader	65
6.15.3.2	GetMyId	65
6.15.3.3	GetQueueCapacity	66
6.15.3.4	GetQueueSize	66
6.15.3.5	ReceiveEvent	66
6.16	ArtdaqSharedMemoryServiceInterface Class Reference	66
6.16.1	Detailed Description	67
6.16.2	Member Function Documentation	67
6.16.2.1	GetEventHeader	67
6.16.2.2	GetMyId	67
6.16.2.3	GetQueueCapacity	68
6.16.2.4	GetQueueSize	68
6.16.2.5	ReceiveEvent	68
6.17	art::ArtdaqSharedMemoryServiceInterfaceConfig Struct Reference	68
6.17.1	Detailed Description	69
6.18	artdaq::AutodetectTransfer Class Reference	69
6.18.1	Detailed Description	70
6.18.2	Constructor & Destructor Documentation	70
6.18.2.1	AutodetectTransfer	70
6.18.3	Member Function Documentation	70
6.18.3.1	isRunning	70
6.18.3.2	receiveFragment	71
6.18.3.3	receiveFragmentData	71
6.18.3.4	receiveFragmentHeader	71
6.18.3.5	transfer_fragment_min_blocking_mode	72
6.18.3.6	transfer_fragment_reliable_mode	72
6.19	art::BinaryFileOutput Class Reference	72
6.19.1	Detailed Description	73
6.19.2	Constructor & Destructor Documentation	73
6.19.2.1	BinaryFileOutput	73
6.20	art::BinaryNetOutput Class Reference	73
6.20.1	Detailed Description	74

6.20.2	Constructor & Destructor Documentation	74
6.20.2.1	BinaryNetOutput	74
6.21	artdaq::BoardReaderApp Class Reference	74
6.21.1	Detailed Description	76
6.21.2	Member Function Documentation	76
6.21.2.1	BootedEnter	76
6.21.2.2	do_initialize	76
6.21.2.3	do_meta_command	76
6.21.2.4	do_pause	77
6.21.2.5	do_reinitialize	78
6.21.2.6	do_resume	78
6.21.2.7	do_shutdown	78
6.21.2.8	do_soft_initialize	79
6.21.2.9	do_start	79
6.21.2.10	do_stop	79
6.21.2.11	operator=	80
6.21.2.12	report	80
6.22	artdaq::BoardReaderCore Class Reference	80
6.22.1	Detailed Description	82
6.22.2	Constructor & Destructor Documentation	82
6.22.2.1	BoardReaderCore	82
6.22.3	Member Function Documentation	82
6.22.3.1	GetDataSenderManagerPtr	82
6.22.3.2	GetFragmentsProcessed	83
6.22.3.3	GetReceiverThreadActive	83
6.22.3.4	GetSenderThreadActive	83
6.22.3.5	initialize	83
6.22.3.6	metaCommand	84
6.22.3.7	operator=	84
6.22.3.8	pause	84
6.22.3.9	receive_fragments	85
6.22.3.10	reinitialize	85
6.22.3.11	report	85
6.22.3.12	resume	85
6.22.3.13	send_fragments	86
6.22.3.14	SetStartTransitionTimeout	86
6.22.3.15	shutdown	86



6.22.3.16	soft_initialize	86
6.22.3.17	start	86
6.22.3.18	stop	87
6.23	artdaqtest::BrokenTransferTest Class Reference	87
6.23.1	Detailed Description	88
6.23.2	Constructor & Destructor Documentation	88
6.23.2.1	BrokenTransferTest	88
6.23.3	Member Function Documentation	88
6.23.3.1	TestReceiverPause	88
6.23.3.2	TestReceiverReconnect	88
6.23.3.3	TestSenderPause	88
6.23.3.4	TestSenderReconnect	89
6.24	FragmentBuffer_t::BufferMode Struct Reference	89
6.24.1	Detailed Description	89
6.25	FragmentBuffer_t::BufferMode_id Struct Reference	89
6.25.1	Detailed Description	89
6.26	FragmentBuffer_t::BufferMode_KeepLatest Struct Reference	89
6.26.1	Detailed Description	90
6.27	FragmentBuffer_t::BufferMode_KeepLatest_id Struct Reference	90
6.27.1	Detailed Description	90
6.28	FragmentBuffer_t::BufferMode_MultipleIDs Struct Reference	90
6.28.1	Detailed Description	90
6.29	FragmentBuffer_t::BufferMode_MultipleIDs_id Struct Reference	91
6.29.1	Detailed Description	91
6.30	artdaq::BuildInfo< instanceName, Pkgs > Class Template Reference	91
6.30.1	Detailed Description	91
6.30.2	Constructor & Destructor Documentation	92
6.30.2.1	BuildInfo	92
6.30.3	Member Function Documentation	92
6.30.3.1	beginRun	92
6.30.3.2	produce	92
6.31	artdaq::BundleTransfer Class Reference	93
6.31.1	Detailed Description	93
6.31.2	Constructor & Destructor Documentation	93
6.31.2.1	BundleTransfer	93
6.31.3	Member Function Documentation	94
6.31.3.1	isRunning	94

6.31.3.2	<a href="#">receiveFragment</a>	94
6.31.3.3	<a href="#">receiveFragmentData</a>	94
6.31.3.4	<a href="#">receiveFragmentHeader</a>	95
6.31.3.5	<a href="#">transfer_fragment_min_blocking_mode</a>	96
6.31.3.6	<a href="#">transfer_fragment_reliable_mode</a>	96
6.32	<a href="#">artdaq::CapacityTestPolicy Class Reference</a>	96
6.32.1	<a href="#">Detailed Description</a>	97
6.32.2	<a href="#">Constructor &amp; Destructor Documentation</a>	97
6.32.2.1	<a href="#">CapacityTestPolicy</a>	97
6.32.3	<a href="#">Member Function Documentation</a>	97
6.32.3.1	<a href="#">CreateRouteForSequenceID</a>	98
6.32.3.2	<a href="#">CreateRoutingTable</a>	99
6.33	<a href="#">FragmentBuffer_t::CircularBufferMode Struct Reference</a>	99
6.33.1	<a href="#">Detailed Description</a>	99
6.34	<a href="#">FragmentBuffer_t::CircularBufferMode_id Struct Reference</a>	100
6.34.1	<a href="#">Detailed Description</a>	100
6.35	<a href="#">FragmentBuffer_t::CircularBufferMode_MultipleIDs Struct Reference</a>	100
6.35.1	<a href="#">Detailed Description</a>	100
6.36	<a href="#">FragmentBuffer_t::CircularBufferMode_MultipleIDs_id Struct Reference</a>	100
6.36.1	<a href="#">Detailed Description</a>	100
6.37	<a href="#">FragmentBuffer_t::CircularBufferMode_RateTests Struct Reference</a>	101
6.37.1	<a href="#">Detailed Description</a>	101
6.38	<a href="#">FragmentBuffer_t::CircularBufferMode_RateTests_id Struct Reference</a>	101
6.38.1	<a href="#">Detailed Description</a>	101
6.39	<a href="#">artdaq::clear_config_archive_ Class Reference</a>	101
6.39.1	<a href="#">Detailed Description</a>	102
6.39.2	<a href="#">Constructor &amp; Destructor Documentation</a>	102
6.39.2.1	<a href="#">clear_config_archive_</a>	102
6.40	<a href="#">artdaq::cmd_ Class Reference</a>	102
6.40.1	<a href="#">Detailed Description</a>	104
6.40.2	<a href="#">Constructor &amp; Destructor Documentation</a>	104
6.40.2.1	<a href="#">cmd_</a>	104
6.40.3	<a href="#">Member Function Documentation</a>	104
6.40.3.1	<a href="#">execute</a>	104
6.40.3.2	<a href="#">execute_</a>	105
6.40.3.3	<a href="#">getParam</a>	106
6.40.3.4	<a href="#">getParam</a>	106

6.40.3.5	getParam	107
6.40.3.6	getParam	107
6.41	artdaq::Commandable Class Reference	107
6.41.1	Detailed Description	110
6.41.2	Constructor & Destructor Documentation	110
6.41.2.1	Commandable	110
6.41.3	Member Function Documentation	110
6.41.3.1	badTransition	110
6.41.3.2	BootedEnter	110
6.41.3.3	current_state	110
6.41.3.4	do_add_config_archive_entry	110
6.41.3.5	do_clear_config_archive	111
6.41.3.6	do_initialize	111
6.41.3.7	do_meta_command	111
6.41.3.8	do_pause	112
6.41.3.9	do_reinitialize	112
6.41.3.10	do_resume	112
6.41.3.11	do_rollover_subrun	112
6.41.3.12	do_shutdown	113
6.41.3.13	do_soft_initialize	113
6.41.3.14	do_start	113
6.41.3.15	do_stop	114
6.41.3.16	do_trace_get	114
6.41.3.17	do_trace_set	114
6.41.3.18	in_run_failure	114
6.41.3.19	initialize	115
6.41.3.20	InRunExit	115
6.41.3.21	legal_commands	115
6.41.3.22	operator=	115
6.41.3.23	pause	115
6.41.3.24	register_monitor	116
6.41.3.25	reinitialize	116
6.41.3.26	report	116
6.41.3.27	resume	116
6.41.3.28	shutdown	117
6.41.3.29	soft_initialize	117
6.41.3.30	start	117

6.41.3.31 status . . . . .	118
6.41.3.32 stop . . . . .	118
6.41.3.33 unregister_monitor . . . . .	118
6.42 artdaq::CommandableFragmentGenerator Class Reference . . . . .	119
6.42.1 Detailed Description . . . . .	121
6.42.2 Constructor & Destructor Documentation . . . . .	122
6.42.2.1 CommandableFragmentGenerator . . . . .	122
6.42.2.2 ~CommandableFragmentGenerator . . . . .	122
6.42.3 Member Function Documentation . . . . .	122
6.42.3.1 check_stop . . . . .	122
6.42.3.2 checkHWStatus_ . . . . .	122
6.42.3.3 ev_counter . . . . .	123
6.42.3.4 ev_counter_inc . . . . .	123
6.42.3.5 exception . . . . .	123
6.42.3.6 fragment_id . . . . .	123
6.42.3.7 fragmentIDs . . . . .	123
6.42.3.8 getNext . . . . .	124
6.42.3.9 getNext_ . . . . .	124
6.42.3.10 GetRequestBuffer . . . . .	124
6.42.3.11 joinThreads . . . . .	124
6.42.3.12 metaCommand . . . . .	125
6.42.3.13 metricsReportingInstanceName . . . . .	126
6.42.3.14 metricsReportingInstanceName . . . . .	126
6.42.3.15 pause . . . . .	126
6.42.3.16 PauseCmd . . . . .	126
6.42.3.17 pauseNoMutex . . . . .	127
6.42.3.18 report . . . . .	127
6.42.3.19 ReportCmd . . . . .	127
6.42.3.20 reportSpecific . . . . .	127
6.42.3.21 resume . . . . .	128
6.42.3.22 ResumeCmd . . . . .	128
6.42.3.23 run_number . . . . .	128
6.42.3.24 set_exception . . . . .	128
6.42.3.25 SetRequestBuffer . . . . .	128
6.42.3.26 should_stop . . . . .	129
6.42.3.27 start . . . . .	129
6.42.3.28 StartCmd . . . . .	129

6.42.3.29 stop	129
6.42.3.30 StopCmd	130
6.42.3.31 stopNoMutex	131
6.42.3.32 subrun_number	131
6.42.3.33 timeout	131
6.42.3.34 timestamp	131
6.43 artdaqtest::CommandableFragmentGeneratorTest Class Reference	132
6.43.1 Detailed Description	133
6.43.2 Member Function Documentation	133
6.43.2.1 checkHWStatus_	133
6.43.2.2 getNext_	133
6.43.2.3 getTimestamp	133
6.43.2.4 setEnabledIds	133
6.43.2.5 setFireCount	134
6.43.2.6 setTimestamp	134
6.44 artdaq::CommanderInterface Class Reference	134
6.44.1 Detailed Description	136
6.44.2 Constructor & Destructor Documentation	136
6.44.2.1 CommanderInterface	136
6.44.3 Member Function Documentation	136
6.44.3.1 add_config_archive_entry	136
6.44.3.2 clear_config_archive	137
6.44.3.3 GetStatus	137
6.44.3.4 operator=	137
6.44.3.5 run_server	138
6.44.3.6 send_init	138
6.44.3.7 send_legal_commands	138
6.44.3.8 send_meta_command	138
6.44.3.9 send_pause	139
6.44.3.10 send_register_monitor	139
6.44.3.11 send_reinit	139
6.44.3.12 send_report	141
6.44.3.13 send_resume	141
6.44.3.14 send_rollover_subrun	141
6.44.3.15 send_shutdown	143
6.44.3.16 send_soft_init	143
6.44.3.17 send_start	143

6.44.3.18	send_status	145
6.44.3.19	send_stop	145
6.44.3.20	send_trace_get	145
6.44.3.21	send_trace_set	146
6.44.3.22	send_unregister_monitor	146
6.44.4	Member Data Documentation	147
6.44.4.1	_commandable	147
6.45	artdaq::CompositeDriver Class Reference	147
6.45.1	Detailed Description	148
6.45.2	Constructor & Destructor Documentation	148
6.45.2.1	CompositeDriver	148
6.45.2.2	CompositeDriver	148
6.46	artdaq::CommandableFragmentGenerator::Config Struct Reference	149
6.46.1	Detailed Description	149
6.46.2	Member Data Documentation	149
6.46.2.1	fragment_id	150
6.46.2.2	fragment_ids	150
6.47	artdaq::TransferInterface::Config Struct Reference	150
6.47.1	Detailed Description	151
6.48	artdaq::artdaqapp::Config Struct Reference	151
6.48.1	Detailed Description	151
6.48.2	Member Data Documentation	152
6.48.2.1	commanderPluginConfig	152
6.49	Config Struct Reference	152
6.49.1	Detailed Description	152
6.50	ArtdaqGlobalsService::Config Struct Reference	152
6.50.1	Detailed Description	152
6.51	mfplugins::ELArtdaqMetric::Config Struct Reference	152
6.51.1	Detailed Description	153
6.51.2	Member Data Documentation	153
6.51.2.1	showDebug	153
6.51.2.2	showError	153
6.51.2.3	showInfo	154
6.51.2.4	showWarning	154
6.52	ArtdaqSharedMemoryService::Config Struct Reference	154
6.52.1	Detailed Description	154
6.53	art::Config Struct Reference	155

6.53.1 Detailed Description . . . . .	155
6.53.2 Member Data Documentation . . . . .	155
6.53.2.1 source . . . . .	155
6.54 art::RootDAQOut::Config Struct Reference . . . . .	155
6.54.1 Detailed Description . . . . .	156
6.54.2 Member Data Documentation . . . . .	156
6.54.2.1 dropMetaDataForDroppedData . . . . .	156
6.54.2.2 fileProperties . . . . .	157
6.54.2.3 saveMemoryObjectThreshold . . . . .	157
6.55 artdaq::GenericFragmentSimulator::Config Struct Reference . . . . .	157
6.55.1 Detailed Description . . . . .	158
6.55.2 Member Data Documentation . . . . .	158
6.55.2.1 content_selection . . . . .	158
6.55.2.2 starting_fragment_id . . . . .	158
6.56 artdaq::HostMap::Config Struct Reference . . . . .	158
6.56.1 Detailed Description . . . . .	159
6.56.2 Member Data Documentation . . . . .	159
6.56.2.1 host_map . . . . .	159
6.57 artdaq::PortManager::Config Struct Reference . . . . .	159
6.57.1 Detailed Description . . . . .	160
6.58 artdaqtest::BrokenTransferTest::Config Struct Reference . . . . .	160
6.58.1 Detailed Description . . . . .	161
6.59 artdaq::DataSenderManager::Config Struct Reference . . . . .	161
6.59.1 Detailed Description . . . . .	162
6.59.2 Member Data Documentation . . . . .	162
6.59.2.1 destinations . . . . .	162
6.59.2.2 routing_table_config . . . . .	162
6.60 artdaq::RequestReceiver::Config Struct Reference . . . . .	163
6.60.1 Detailed Description . . . . .	163
6.61 artdaq::RequestSender::Config Struct Reference . . . . .	163
6.61.1 Detailed Description . . . . .	164
6.62 artdaq::TableReceiver::Config Struct Reference . . . . .	164
6.62.1 Detailed Description . . . . .	165
6.63 artdaq::TokenReceiver::Config Struct Reference . . . . .	165
6.63.1 Detailed Description . . . . .	165
6.64 artdaq::FragmentBuffer::Config Struct Reference . . . . .	165
6.64.1 Detailed Description . . . . .	167

6.64.2	Member Data Documentation	167
6.64.2.1	fragment_id	168
6.64.2.2	fragment_ids	168
6.64.2.3	request_mode	168
6.65	artdaq::TokenSender::Config Struct Reference	168
6.65.1	Detailed Description	169
6.66	artdaq::SharedMemoryEventManager::Config Struct Reference	169
6.66.1	Detailed Description	171
6.66.2	Member Data Documentation	171
6.66.2.1	expected_art_event_processing_time_us	171
6.66.2.2	fragment_broadcast_timeout_ms	171
6.66.2.3	max_event_size_bytes	172
6.66.2.4	max_fragment_size_bytes	172
6.67	artdaq::CommanderInterface::Config Struct Reference	172
6.67.1	Detailed Description	172
6.68	FragCounter_test::Construct Struct Reference	172
6.68.1	Detailed Description	173
6.69	RequestSender_test::Construct Struct Reference	173
6.69.1	Detailed Description	173
6.70	FragCounter_test::Construct_id Struct Reference	173
6.70.1	Detailed Description	173
6.71	RequestSender_test::Construct_id Struct Reference	174
6.71.1	Detailed Description	174
6.72	CapacityTest_policy_t::DataFlowMode Struct Reference	174
6.72.1	Detailed Description	174
6.73	NoOp_policy_t::DataFlowMode Struct Reference	174
6.73.1	Detailed Description	175
6.74	PreferSameHost_policy_t::DataFlowMode Struct Reference	175
6.74.1	Detailed Description	175
6.75	RoundRobin_policy_t::DataFlowMode Struct Reference	175
6.75.1	Detailed Description	176
6.76	CapacityTest_policy_t::DataFlowMode_id Struct Reference	176
6.76.1	Detailed Description	176
6.77	NoOp_policy_t::DataFlowMode_id Struct Reference	176
6.77.1	Detailed Description	176
6.78	PreferSameHost_policy_t::DataFlowMode_id Struct Reference	176
6.78.1	Detailed Description	176



6.79 RoundRobin_policy_t::DataFlowMode_id Struct Reference . . . . .	176
6.79.1 Detailed Description . . . . .	176
6.80 artdaq::DataLoggerApp Class Reference . . . . .	177
6.80.1 Detailed Description . . . . .	178
6.80.2 Member Function Documentation . . . . .	178
6.80.2.1 do_add_config_archive_entry . . . . .	178
6.80.2.2 do_clear_config_archive . . . . .	178
6.80.2.3 do_initialize . . . . .	178
6.80.2.4 do_pause . . . . .	179
6.80.2.5 do_reinitialize . . . . .	179
6.80.2.6 do_resume . . . . .	179
6.80.2.7 do_shutdown . . . . .	179
6.80.2.8 do_soft_initialize . . . . .	179
6.80.2.9 do_start . . . . .	180
6.80.2.10 do_stop . . . . .	180
6.80.2.11 operator= . . . . .	180
6.80.2.12 report . . . . .	180
6.81 artdaq::DataLoggerCore Class Reference . . . . .	181
6.81.1 Detailed Description . . . . .	182
6.81.2 Constructor & Destructor Documentation . . . . .	182
6.81.2.1 ~DataLoggerCore . . . . .	182
6.81.3 Member Function Documentation . . . . .	182
6.81.3.1 initialize . . . . .	182
6.81.3.2 operator= . . . . .	183
6.82 artdaq::DataReceiverCore Class Reference . . . . .	183
6.82.1 Detailed Description . . . . .	185
6.82.2 Constructor & Destructor Documentation . . . . .	185
6.82.2.1 ~DataReceiverCore . . . . .	185
6.82.3 Member Function Documentation . . . . .	185
6.82.3.1 add_config_archive_entry . . . . .	185
6.82.3.2 clear_config_archive . . . . .	185
6.82.3.3 initialize . . . . .	185
6.82.3.4 initializeDataReceiver . . . . .	186
6.82.3.5 operator= . . . . .	186
6.82.3.6 pause . . . . .	186
6.82.3.7 reinitialize . . . . .	187
6.82.3.8 report . . . . .	188

6.82.3.9	resume	188
6.82.3.10	rollover_subrun	188
6.82.3.11	shutdown	189
6.82.3.12	soft_initialize	189
6.82.3.13	start	189
6.82.3.14	stop	189
6.83	artdaq::DataReceiverManager Class Reference	190
6.83.1	Detailed Description	190
6.83.2	Constructor & Destructor Documentation	190
6.83.2.1	DataReceiverManager	190
6.83.3	Member Function Documentation	191
6.83.3.1	byteCount	191
6.83.3.2	count	191
6.83.3.3	enabled_sources	191
6.83.3.4	getSharedMemoryEventManager	191
6.83.3.5	running_sources	192
6.83.3.6	slotCount	192
6.84	artdaq::DataSenderManager Class Reference	192
6.84.1	Detailed Description	193
6.84.2	Constructor & Destructor Documentation	193
6.84.2.1	DataSenderManager	193
6.84.3	Member Function Documentation	194
6.84.3.1	count	194
6.84.3.2	destinationCount	194
6.84.3.3	enabled_destinations	194
6.84.3.4	GetRemainingRoutingTableEntries	194
6.84.3.5	GetRoutingTableEntryCount	194
6.84.3.6	GetSentSequenceIDCount	194
6.84.3.7	RemoveRoutingTableEntry	195
6.84.3.8	sendFragment	195
6.84.3.9	slotCount	195
6.85	artdaq::DataSenderManager::DestinationsConfig Struct Reference	196
6.85.1	Detailed Description	196
6.86	artdaq::DispatcherApp Class Reference	196
6.86.1	Detailed Description	197
6.86.2	Member Function Documentation	197
6.86.2.1	do_initialize	197

6.86.2.2	<a href="#">do_pause</a>	198
6.86.2.3	<a href="#">do_reinitialize</a>	198
6.86.2.4	<a href="#">do_resume</a>	198
6.86.2.5	<a href="#">do_shutdown</a>	198
6.86.2.6	<a href="#">do_soft_initialize</a>	198
6.86.2.7	<a href="#">do_start</a>	199
6.86.2.8	<a href="#">do_stop</a>	199
6.86.2.9	<a href="#">operator=</a>	199
6.86.2.10	<a href="#">register_monitor</a>	199
6.86.2.11	<a href="#">report</a>	200
6.86.2.12	<a href="#">unregister_monitor</a>	200
6.87	<a href="#">artdaq::DispatcherCore Class Reference</a>	200
6.87.1	<a href="#">Detailed Description</a>	201
6.87.2	<a href="#">Constructor &amp; Destructor Documentation</a>	201
6.87.2.1	<a href="#">~DispatcherCore</a>	201
6.87.3	<a href="#">Member Function Documentation</a>	202
6.87.3.1	<a href="#">initialize</a>	202
6.87.3.2	<a href="#">operator=</a>	203
6.87.3.3	<a href="#">register_monitor</a>	203
6.87.3.4	<a href="#">unregister_monitor</a>	204
6.88	<a href="#">mfplugins::ELArtdaqMetric Class Reference</a>	205
6.88.1	<a href="#">Detailed Description</a>	206
6.88.2	<a href="#">Constructor &amp; Destructor Documentation</a>	206
6.88.2.1	<a href="#">ELArtdaqMetric</a>	206
6.88.3	<a href="#">Member Function Documentation</a>	206
6.88.3.1	<a href="#">fillPrefix</a>	206
6.88.3.2	<a href="#">fillUsrMsg</a>	206
6.88.3.3	<a href="#">routePayload</a>	206
6.89	<a href="#">anonymous_namespace{xmlrpc_commander.cc}::env_wrap Class Reference</a>	207
6.89.1	<a href="#">Detailed Description</a>	207
6.90	<a href="#">artdaq::EventBuilderApp Class Reference</a>	207
6.90.1	<a href="#">Detailed Description</a>	208
6.90.2	<a href="#">Member Function Documentation</a>	208
6.90.2.1	<a href="#">BootedEnter</a>	208
6.90.2.2	<a href="#">do_add_config_archive_entry</a>	209
6.90.2.3	<a href="#">do_clear_config_archive</a>	209
6.90.2.4	<a href="#">do_initialize</a>	209

6.90.2.5	<a href="#">do_pause</a>	209
6.90.2.6	<a href="#">do_reinitialize</a>	209
6.90.2.7	<a href="#">do_resume</a>	210
6.90.2.8	<a href="#">do_rollover_subrun</a>	210
6.90.2.9	<a href="#">do_shutdown</a>	210
6.90.2.10	<a href="#">do_soft_initialize</a>	210
6.90.2.11	<a href="#">do_start</a>	211
6.90.2.12	<a href="#">do_stop</a>	211
6.90.2.13	<a href="#">operator=</a>	211
6.90.2.14	<a href="#">report</a>	211
6.91	<a href="#">artdaq::EventBuilderCore Class Reference</a>	212
6.91.1	<a href="#">Detailed Description</a>	213
6.91.2	<a href="#">Constructor &amp; Destructor Documentation</a>	213
6.91.2.1	<a href="#">~EventBuilderCore</a>	213
6.91.3	<a href="#">Member Function Documentation</a>	213
6.91.3.1	<a href="#">initialize</a>	213
6.91.3.2	<a href="#">operator=</a>	213
6.92	<a href="#">artdaq::EventDump Class Reference</a>	214
6.92.1	<a href="#">Detailed Description</a>	214
6.92.2	<a href="#">Constructor &amp; Destructor Documentation</a>	214
6.92.2.1	<a href="#">EventDump</a>	214
6.92.3	<a href="#">Member Function Documentation</a>	215
6.92.3.1	<a href="#">analyze</a>	215
6.93	<a href="#">art::RootDAQOut::Config::FileNameSubstitution Struct Reference</a>	215
6.93.1	<a href="#">Detailed Description</a>	215
6.94	<a href="#">art::FiltersConfig Struct Reference</a>	215
6.94.1	<a href="#">Detailed Description</a>	215
6.95	<a href="#">artdaq::detail::FragCounter Class Reference</a>	215
6.95.1	<a href="#">Detailed Description</a>	216
6.95.2	<a href="#">Member Function Documentation</a>	216
6.95.2.1	<a href="#">count</a>	216
6.95.2.2	<a href="#">incSlot</a>	216
6.95.2.3	<a href="#">incSlot</a>	216
6.95.2.4	<a href="#">minCount</a>	217
6.95.2.5	<a href="#">nSlots</a>	217
6.95.2.6	<a href="#">operator[]</a>	217
6.95.2.7	<a href="#">setSlot</a>	217

6.95.2.8 slotCount . . . . .	217
6.96 artdaq::FragmentBuffer Class Reference . . . . .	218
6.96.1 Detailed Description . . . . .	220
6.96.2 Constructor & Destructor Documentation . . . . .	220
6.96.2.1 FragmentBuffer . . . . .	220
6.96.2.2 ~FragmentBuffer . . . . .	221
6.96.3 Member Function Documentation . . . . .	221
6.96.3.1 AddFragmentsToBuffer . . . . .	221
6.96.3.2 applyRequests . . . . .	221
6.96.3.3 applyRequestsBufferMode . . . . .	221
6.96.3.4 applyRequestsIgnoredMode . . . . .	221
6.96.3.5 applyRequestsSequenceIDMode . . . . .	222
6.96.3.6 applyRequestsSingleMode . . . . .	223
6.96.3.7 applyRequestsWindowMode . . . . .	223
6.96.3.8 applyRequestsWindowMode_CheckAndFillDataBuffer . . . . .	223
6.96.3.9 check_stop . . . . .	223
6.96.3.10 checkDataBuffer . . . . .	223
6.96.3.11 checkSentWindows . . . . .	224
6.96.3.12 dataBufferFragmentCount_ . . . . .	224
6.96.3.13 dataBufferIsTooLarge . . . . .	224
6.96.3.14 fragment_id . . . . .	224
6.96.3.15 fragmentIDs . . . . .	225
6.96.3.16 getDataBufferStats . . . . .	225
6.96.3.17 GetNextSequenceID . . . . .	225
6.96.3.18 GetSentWindowList . . . . .	225
6.96.3.19 printMode_ . . . . .	225
6.96.3.20 request_mode . . . . .	226
6.96.3.21 Reset . . . . .	226
6.96.3.22 sendEmptyFragment . . . . .	226
6.96.3.23 sendEmptyFragments . . . . .	226
6.96.3.24 SetRequestBuffer . . . . .	226
6.96.3.25 waitForDataBufferReady . . . . .	227
6.97 artdaqtest::FragmentBufferTestGenerator Class Reference . . . . .	227
6.97.1 Detailed Description . . . . .	227
6.97.2 Member Function Documentation . . . . .	228
6.97.2.1 Generate . . . . .	228
6.97.2.2 getTimestamp . . . . .	229

6.97.2.3	setTimestamp	229
6.98	artdaq::FragmentReceiverManager Class Reference	229
6.98.1	Detailed Description	230
6.98.2	Constructor & Destructor Documentation	230
6.98.2.1	FragmentReceiverManager	230
6.98.3	Member Function Documentation	230
6.98.3.1	byteCount	230
6.98.3.2	count	231
6.98.3.3	enabled_sources	231
6.98.3.4	recvFragment	231
6.98.3.5	running_sources	231
6.98.3.6	slotCount	231
6.99	artdaq::FragmentSniffer Class Reference	232
6.99.1	Detailed Description	232
6.99.2	Constructor & Destructor Documentation	233
6.99.2.1	FragmentSniffer	233
6.99.3	Member Function Documentation	234
6.99.3.1	analyze	234
6.100	artdaq::FragmentStoreElement Class Reference	234
6.100.1	Detailed Description	235
6.100.2	Member Function Documentation	235
6.100.2.1	emplace_back	235
6.100.2.2	emplace_front	235
6.100.2.3	empty	235
6.100.2.4	front	235
6.100.2.5	GetEndOfData	236
6.100.2.6	SetEndOfData	236
6.100.2.7	size	236
6.101	artdaq::FragmentWatcher Class Reference	236
6.101.1	Detailed Description	237
6.101.2	Constructor & Destructor Documentation	237
6.101.2.1	FragmentWatcher	237
6.101.3	Member Function Documentation	237
6.101.3.1	analyze	237
6.102	artdaq::GenericFragmentSimulator Class Reference	237
6.102.1	Detailed Description	238
6.102.2	Member Enumeration Documentation	238

6.102.2.1 content_selector_t . . . . .	238
6.102.3 Constructor & Destructor Documentation . . . . .	239
6.102.3.1 GenericFragmentSimulator . . . . .	239
6.102.4 Member Function Documentation . . . . .	239
6.102.4.1 fragmentIDs . . . . .	239
6.102.4.2 getNext . . . . .	239
6.102.4.3 getNext . . . . .	240
6.103artdaq::GenToBufferTest Class Reference . . . . .	241
6.103.1 Detailed Description . . . . .	241
6.103.2 Constructor & Destructor Documentation . . . . .	241
6.103.2.1 GenToBufferTest . . . . .	241
6.103.3 Member Function Documentation . . . . .	242
6.103.3.1 GetRequestBuffer . . . . .	242
6.103.3.2 start . . . . .	242
6.104artdaq::GetPackageBuildInfo Struct Reference . . . . .	242
6.104.1 Detailed Description . . . . .	242
6.104.2 Member Function Documentation . . . . .	242
6.104.2.1 getPackageBuildInfo . . . . .	242
6.105artdaq::Globals Class Reference . . . . .	243
6.105.1 Detailed Description . . . . .	244
6.105.2 Member Function Documentation . . . . .	244
6.105.2.1 GetMFIteration_ . . . . .	244
6.105.2.2 GetMFModuleName_ . . . . .	244
6.105.2.3 getPartitionNumber_ . . . . .	244
6.105.2.4 seedAndRandom_ . . . . .	244
6.105.2.5 SetMFIteration_ . . . . .	245
6.105.2.6 SetMFModuleName_ . . . . .	246
6.106CommandableFragmentGenerator_t::HardwareFailure_NonThreaded Struct Reference . . . . .	246
6.106.1 Detailed Description . . . . .	246
6.107CommandableFragmentGenerator_t::HardwareFailure_NonThreaded_id Struct Reference . . . . .	246
6.107.1 Detailed Description . . . . .	246
6.108CommandableFragmentGenerator_t::HardwareFailure_Threaded Struct Reference . . . . .	247
6.108.1 Detailed Description . . . . .	247
6.109CommandableFragmentGenerator_t::HardwareFailure_Threaded_id Struct Reference . . . . .	247
6.109.1 Detailed Description . . . . .	247
6.110artdaq::HostMap::HostConfig Struct Reference . . . . .	247
6.110.1 Detailed Description . . . . .	248

6.111	<a href="#">artdaq::HostMap Struct Reference</a>	248
6.111.1	<a href="#">Detailed Description</a>	248
6.112	<a href="#">FragmentBuffer_t::IgnoreRequests Struct Reference</a>	248
6.112.1	<a href="#">Detailed Description</a>	248
6.113	<a href="#">FragmentBuffer_t::IgnoreRequests_id Struct Reference</a>	249
6.113.1	<a href="#">Detailed Description</a>	249
6.114	<a href="#">FragmentBuffer_t::IgnoreRequests_MultipleIDs Struct Reference</a>	249
6.114.1	<a href="#">Detailed Description</a>	249
6.115	<a href="#">FragmentBuffer_t::IgnoreRequests_MultipleIDs_id Struct Reference</a>	249
6.115.1	<a href="#">Detailed Description</a>	249
6.116	<a href="#">FragmentBuffer_t::IgnoreRequests_StateMachine Struct Reference</a>	249
6.116.1	<a href="#">Detailed Description</a>	250
6.117	<a href="#">FragmentBuffer_t::IgnoreRequests_StateMachine_id Struct Reference</a>	250
6.117.1	<a href="#">Detailed Description</a>	250
6.118	<a href="#">FragmentBuffer_t::ImproperConfiguration Struct Reference</a>	250
6.118.1	<a href="#">Detailed Description</a>	250
6.119	<a href="#">FragmentBuffer_t::ImproperConfiguration_id Struct Reference</a>	251
6.119.1	<a href="#">Detailed Description</a>	251
6.120	<a href="#">artdaq::init_ Class Reference</a>	251
6.120.1	<a href="#">Detailed Description</a>	251
6.120.2	<a href="#">Constructor &amp; Destructor Documentation</a>	251
6.120.2.1	<a href="#">init_</a>	251
6.120.3	<a href="#">Member Data Documentation</a>	252
6.120.3.1	<a href="#">defaultTimeout</a>	252
6.120.3.2	<a href="#">defaultTimestamp</a>	252
6.121	<a href="#">art::RootOutputConfig::KeysToIgnore Struct Reference</a>	252
6.121.1	<a href="#">Detailed Description</a>	252
6.121.2	<a href="#">Member Function Documentation</a>	252
6.121.2.1	<a href="#">operator()</a>	252
6.122	<a href="#">art::RootDAQOut::Config::KeysToIgnore Struct Reference</a>	253
6.122.1	<a href="#">Detailed Description</a>	253
6.123	<a href="#">RoundRobin_policy_t::LargeMinimumParticipants Struct Reference</a>	253
6.123.1	<a href="#">Detailed Description</a>	253
6.124	<a href="#">PreferSameHost_policy_t::LargeMinimumParticipants Struct Reference</a>	254
6.124.1	<a href="#">Detailed Description</a>	254
6.125	<a href="#">RoundRobin_policy_t::LargeMinimumParticipants_id Struct Reference</a>	254
6.125.1	<a href="#">Detailed Description</a>	254



6.126	<a href="#">PreferSameHost_policy_t::LargeMinimumParticipants_id Struct Reference</a>	254
6.126.1	Detailed Description	254
6.127	<a href="#">artdaq::legal_commands_ Class Reference</a>	254
6.127.1	Detailed Description	255
6.127.2	Constructor & Destructor Documentation	255
6.127.2.1	legal_commands_	255
6.128	<a href="#">artdaq::ListenTransferWrapper Class Reference</a>	255
6.128.1	Detailed Description	256
6.128.2	Constructor & Destructor Documentation	256
6.128.2.1	ListenTransferWrapper	256
6.128.3	Member Function Documentation	256
6.128.3.1	getEventHeader	256
6.128.3.2	receiveInitMessage	257
6.128.3.3	receiveMessage	257
6.128.3.4	receiveMessages	257
6.129	<a href="#">PreferSameHost_policy_t::ManyMissingParticipants Struct Reference</a>	257
6.129.1	Detailed Description	258
6.130	<a href="#">RoundRobin_policy_t::ManyMissingParticipants Struct Reference</a>	258
6.130.1	Detailed Description	258
6.131	<a href="#">PreferSameHost_policy_t::ManyMissingParticipants_id Struct Reference</a>	258
6.131.1	Detailed Description	258
6.132	<a href="#">RoundRobin_policy_t::ManyMissingParticipants_id Struct Reference</a>	258
6.132.1	Detailed Description	259
6.133	<a href="#">MessHead Struct Reference</a>	259
6.133.1	Detailed Description	259
6.133.2	Member Enumeration Documentation	259
6.133.2.1	MessType	259
6.134	<a href="#">artdaq::meta_command_ Class Reference</a>	260
6.134.1	Detailed Description	260
6.134.2	Constructor & Destructor Documentation	260
6.134.2.1	meta_command_	260
6.135	<a href="#">PreferSameHost_policy_t::MinimumParticipants Struct Reference</a>	261
6.135.1	Detailed Description	261
6.136	<a href="#">RoundRobin_policy_t::MinimumParticipants Struct Reference</a>	261
6.136.1	Detailed Description	261
6.137	<a href="#">PreferSameHost_policy_t::MinimumParticipants_id Struct Reference</a>	262
6.137.1	Detailed Description	262

6.138RoundRobin_policy_t::MinimumParticipants_id Struct Reference . . . . .	262
6.138.1 Detailed Description . . . . .	262
6.139artdaq::MissingDataCheck Class Reference . . . . .	262
6.139.1 Detailed Description . . . . .	262
6.139.2 Constructor & Destructor Documentation . . . . .	263
6.139.2.1 MissingDataCheck . . . . .	263
6.139.3 Member Function Documentation . . . . .	263
6.139.3.1 analyze . . . . .	263
6.140artdaq::MulticastTransfer Class Reference . . . . .	263
6.140.1 Detailed Description . . . . .	264
6.140.2 Constructor & Destructor Documentation . . . . .	264
6.140.2.1 MulticastTransfer . . . . .	264
6.140.3 Member Function Documentation . . . . .	265
6.140.3.1 isRunning . . . . .	265
6.140.3.2 receiveFragment . . . . .	265
6.140.3.3 receiveFragmentData . . . . .	265
6.140.3.4 receiveFragmentHeader . . . . .	265
6.140.3.5 transfer_fragment_min_blocking_mode . . . . .	266
6.140.3.6 transfer_fragment_reliable_mode . . . . .	266
6.141CommandableFragmentGenerator_t::MultipleIDs Struct Reference . . . . .	266
6.141.1 Detailed Description . . . . .	267
6.142CommandableFragmentGenerator_t::MultipleIDs_id Struct Reference . . . . .	267
6.142.1 Detailed Description . . . . .	267
6.143artdaq::NetMonHeader Struct Reference . . . . .	267
6.143.1 Detailed Description . . . . .	267
6.144art::RootDAQOut::Config::NewSubStringForApp Struct Reference . . . . .	268
6.144.1 Detailed Description . . . . .	268
6.145artdaq::NoOpPolicy Class Reference . . . . .	268
6.145.1 Detailed Description . . . . .	268
6.145.2 Constructor & Destructor Documentation . . . . .	269
6.145.2.1 NoOpPolicy . . . . .	269
6.145.3 Member Function Documentation . . . . .	269
6.145.3.1 CreateRouteForSequenceID . . . . .	269
6.145.3.2 CreateRoutingTable . . . . .	269
6.146FragCounter_test::nSlots Struct Reference . . . . .	269
6.146.1 Detailed Description . . . . .	270
6.147FragCounter_test::nSlots_id Struct Reference . . . . .	270

6.147.1 Detailed Description . . . . .	270
6.148artdaq::NullTransfer Class Reference . . . . .	270
6.148.1 Detailed Description . . . . .	271
6.148.2 Constructor & Destructor Documentation . . . . .	271
6.148.2.1 NullTransfer . . . . .	271
6.148.3 Member Function Documentation . . . . .	271
6.148.3.1 isRunning . . . . .	271
6.148.3.2 receiveFragment . . . . .	272
6.148.3.3 receiveFragmentData . . . . .	272
6.148.3.4 receiveFragmentHeader . . . . .	272
6.148.3.5 transfer_fragment_min_blocking_mode . . . . .	272
6.148.3.6 transfer_fragment_reliable_mode . . . . .	273
6.149artdaq::RTIDDS::OctetsListener Class Reference . . . . .	273
6.149.1 Detailed Description . . . . .	273
6.149.2 Member Function Documentation . . . . .	273
6.149.2.1 on_data_available . . . . .	273
6.149.2.2 receiveFragmentFromDDS . . . . .	274
6.150artdaq::OffsetPrescale Class Reference . . . . .	274
6.150.1 Detailed Description . . . . .	274
6.151art::OutputItem Struct Reference . . . . .	275
6.151.1 Detailed Description . . . . .	275
6.152art::OutputsConfig Struct Reference . . . . .	275
6.152.1 Detailed Description . . . . .	275
6.153artdaq::pause_ Class Reference . . . . .	276
6.153.1 Detailed Description . . . . .	276
6.153.2 Constructor & Destructor Documentation . . . . .	276
6.153.2.1 pause_ . . . . .	276
6.153.3 Member Data Documentation . . . . .	276
6.153.3.1 defaultTimeout . . . . .	276
6.153.3.2 defaultTimestamp . . . . .	277
6.154art::PhysicsConfig Struct Reference . . . . .	277
6.154.1 Detailed Description . . . . .	277
6.154.2 Member Data Documentation . . . . .	277
6.154.2.1 filters . . . . .	277
6.155artdaq::PortManager Class Reference . . . . .	277
6.155.1 Detailed Description . . . . .	278
6.155.2 Member Function Documentation . . . . .	279

6.155.2.1 GetMulticastOutputAddress . . . . .	279
6.155.2.2 GetMulticastTransferGroupAddress . . . . .	280
6.155.2.3 GetMulticastTransferPort . . . . .	280
6.155.2.4 GetRequestMessageGroupAddress . . . . .	280
6.155.2.5 GetRequestMessagePort . . . . .	281
6.155.2.6 GetRoutingAckPort . . . . .	281
6.155.2.7 GetRoutingTableGroupAddress . . . . .	281
6.155.2.8 GetRoutingTablePort . . . . .	281
6.155.2.9 GetRoutingTokenPort . . . . .	281
6.155.2.10 GetTCPSocketTransferPort . . . . .	282
6.155.2.11 GetXMLRPCPort . . . . .	282
6.155.2.12 UpdateConfiguration . . . . .	282
6.156 artdaq::PreferSameHostPolicy Class Reference . . . . .	283
6.156.1 Detailed Description . . . . .	283
6.156.2 Constructor & Destructor Documentation . . . . .	283
6.156.2.1 PreferSameHostPolicy . . . . .	283
6.156.3 Member Function Documentation . . . . .	284
6.156.3.1 CreateRouteForSequenceID . . . . .	284
6.156.3.2 CreateRoutingTable . . . . .	285
6.157 artdaq::PrintBuildInfo Class Reference . . . . .	285
6.157.1 Detailed Description . . . . .	286
6.157.2 Constructor & Destructor Documentation . . . . .	286
6.157.2.1 PrintBuildInfo . . . . .	286
6.157.3 Member Function Documentation . . . . .	286
6.157.3.1 beginRun . . . . .	286
6.158 art::ProducersConfig Struct Reference . . . . .	286
6.158.1 Detailed Description . . . . .	286
6.159 artdaq::RandomDelayFilter Class Reference . . . . .	287
6.159.1 Detailed Description . . . . .	287
6.159.2 Constructor & Destructor Documentation . . . . .	287
6.159.2.1 RandomDelayFilter . . . . .	287
6.159.3 Member Function Documentation . . . . .	288
6.159.3.1 filter . . . . .	288
6.159.3.2 operator= . . . . .	288
6.159.3.3 operator= . . . . .	288
6.160 artdaq::register_monitor_ Class Reference . . . . .	289
6.160.1 Detailed Description . . . . .	289

6.160.2 Constructor & Destructor Documentation . . . . .	289
6.160.2.1 register_monitor_ . . . . .	289
6.161 artdaq::reinit_ Class Reference . . . . .	289
6.161.1 Detailed Description . . . . .	290
6.161.2 Constructor & Destructor Documentation . . . . .	290
6.161.2.1 reinit_ . . . . .	290
6.161.3 Member Data Documentation . . . . .	290
6.161.3.1 defaultTimeout . . . . .	290
6.161.3.2 defaultTimestamp . . . . .	290
6.162 artdaq::report_ Class Reference . . . . .	291
6.162.1 Detailed Description . . . . .	291
6.162.2 Constructor & Destructor Documentation . . . . .	291
6.162.2.1 report_ . . . . .	291
6.163 CapacityTest_policy_t::RequestBasedEventBuilding Struct Reference . . . . .	291
6.163.1 Detailed Description . . . . .	292
6.164 NoOp_policy_t::RequestBasedEventBuilding Struct Reference . . . . .	292
6.164.1 Detailed Description . . . . .	292
6.165 PreferSameHost_policy_t::RequestBasedEventBuilding Struct Reference . . . . .	292
6.165.1 Detailed Description . . . . .	293
6.166 RoundRobin_policy_t::RequestBasedEventBuilding Struct Reference . . . . .	293
6.166.1 Detailed Description . . . . .	293
6.167 CapacityTest_policy_t::RequestBasedEventBuilding_id Struct Reference . . . . .	293
6.167.1 Detailed Description . . . . .	293
6.168 PreferSameHost_policy_t::RequestBasedEventBuilding_id Struct Reference . . . . .	294
6.168.1 Detailed Description . . . . .	294
6.169 RoundRobin_policy_t::RequestBasedEventBuilding_id Struct Reference . . . . .	294
6.169.1 Detailed Description . . . . .	294
6.170 NoOp_policy_t::RequestBasedEventBuilding_id Struct Reference . . . . .	294
6.170.1 Detailed Description . . . . .	294
6.171 artdaq::RequestBuffer Class Reference . . . . .	294
6.171.1 Detailed Description . . . . .	295
6.171.2 Constructor & Destructor Documentation . . . . .	295
6.171.2.1 RequestBuffer . . . . .	295
6.171.3 Member Function Documentation . . . . .	296
6.171.3.1 ClearRequests . . . . .	296
6.171.3.2 GetAndClearRequests . . . . .	296
6.171.3.3 GetNextRequest . . . . .	296

6.171.3.4 GetRequests . . . . .	296
6.171.3.5 GetRequestTime . . . . .	296
6.171.3.6 isRunning . . . . .	297
6.171.3.7 push . . . . .	297
6.171.3.8 RemoveRequest . . . . .	297
6.171.3.9 setRunning . . . . .	297
6.171.3.10size . . . . .	297
6.171.3.11WaitForRequests . . . . .	298
6.172artdaq::detail::RequestHeader Struct Reference . . . . .	298
6.172.1 Detailed Description . . . . .	299
6.172.2 Member Function Documentation . . . . .	299
6.172.2.1 isValid . . . . .	299
6.172.3 Member Data Documentation . . . . .	299
6.172.3.1 header . . . . .	299
6.173artdaq::detail::RequestMessage Class Reference . . . . .	299
6.173.1 Detailed Description . . . . .	300
6.173.2 Member Function Documentation . . . . .	300
6.173.2.1 addRequest . . . . .	300
6.173.2.2 GetMessage . . . . .	300
6.173.2.3 max_request_count . . . . .	300
6.173.2.4 setMode . . . . .	301
6.173.2.5 setRank . . . . .	302
6.173.2.6 setRunNumber . . . . .	302
6.173.2.7 size . . . . .	302
6.174artdaq::detail::RequestPacket Struct Reference . . . . .	302
6.174.1 Detailed Description . . . . .	303
6.174.2 Constructor & Destructor Documentation . . . . .	303
6.174.2.1 RequestPacket . . . . .	303
6.174.3 Member Function Documentation . . . . .	303
6.174.3.1 isValid . . . . .	303
6.174.4 Member Data Documentation . . . . .	303
6.174.4.1 header . . . . .	303
6.175artdaq::RequestReceiver Class Reference . . . . .	304
6.175.1 Detailed Description . . . . .	304
6.175.2 Constructor & Destructor Documentation . . . . .	305
6.175.2.1 RequestReceiver . . . . .	305
6.175.3 Member Function Documentation . . . . .	306

6.175.3.1 isRunning . . . . .	306
6.175.3.2 SetRunNumber . . . . .	306
6.175.3.3 stopRequestReception . . . . .	306
6.176RequestSender_test::Requests Struct Reference . . . . .	306
6.176.1 Detailed Description . . . . .	307
6.177RequestSender_test::Requests_id Struct Reference . . . . .	307
6.177.1 Detailed Description . . . . .	307
6.178artdaq::RequestSender Class Reference . . . . .	307
6.178.1 Detailed Description . . . . .	308
6.178.2 Constructor & Destructor Documentation . . . . .	308
6.178.2.1 RequestSender . . . . .	308
6.178.3 Member Function Documentation . . . . .	308
6.178.3.1 AddRequest . . . . .	308
6.178.3.2 GetRequestMode . . . . .	309
6.178.3.3 GetSentMessageCount . . . . .	309
6.178.3.4 operator= . . . . .	309
6.178.3.5 RemoveRequest . . . . .	309
6.178.3.6 RequestsInFlight . . . . .	309
6.178.3.7 SendRequest . . . . .	310
6.178.3.8 SetRequestMode . . . . .	310
6.178.3.9 SetRunNumber . . . . .	310
6.179artdaq::RequestSenderModule Class Reference . . . . .	310
6.179.1 Detailed Description . . . . .	311
6.179.2 Constructor & Destructor Documentation . . . . .	311
6.179.2.1 RequestSenderModule . . . . .	311
6.179.3 Member Function Documentation . . . . .	311
6.179.3.1 analyze . . . . .	311
6.179.3.2 beginRun . . . . .	311
6.179.3.3 endRun . . . . .	311
6.180artdaq::resume_ Class Reference . . . . .	312
6.180.1 Detailed Description . . . . .	312
6.180.2 Constructor & Destructor Documentation . . . . .	312
6.180.2.1 resume_ . . . . .	312
6.180.3 Member Data Documentation . . . . .	313
6.180.3.1 defaultTimeout . . . . .	313
6.180.3.2 defaultTimestamp . . . . .	313
6.181artdaq::rollover_subrun_ Class Reference . . . . .	313

6.181.1 Detailed Description . . . . .	314
6.181.2 Constructor & Destructor Documentation . . . . .	314
6.181.2.1 rollover_subrun_ . . . . .	314
6.182art::RootDAQOut Class Reference . . . . .	314
6.182.1 Detailed Description . . . . .	315
6.183art::RootDAQOutFile Class Reference . . . . .	315
6.183.1 Detailed Description . . . . .	316
6.184art::RootNetOutput Class Reference . . . . .	316
6.184.1 Detailed Description . . . . .	317
6.184.2 Constructor & Destructor Documentation . . . . .	317
6.184.2.1 RootNetOutput . . . . .	317
6.184.3 Member Function Documentation . . . . .	317
6.184.3.1 dataReceiverCount . . . . .	317
6.184.3.2 SendMessage . . . . .	317
6.185art::RootOutputConfig Struct Reference . . . . .	318
6.185.1 Detailed Description . . . . .	319
6.185.2 Member Typedef Documentation . . . . .	319
6.185.2.1 Comment . . . . .	319
6.186artdaq::RoundRobinPolicy Class Reference . . . . .	319
6.186.1 Detailed Description . . . . .	320
6.186.2 Constructor & Destructor Documentation . . . . .	320
6.186.2.1 RoundRobinPolicy . . . . .	320
6.186.3 Member Function Documentation . . . . .	320
6.186.3.1 CreateRouteForSequenceID . . . . .	320
6.186.3.2 CreateRoutingTable . . . . .	321
6.187artdaq::RoutingManagerApp Class Reference . . . . .	322
6.187.1 Detailed Description . . . . .	323
6.187.2 Constructor & Destructor Documentation . . . . .	323
6.187.2.1 RoutingManagerApp . . . . .	323
6.187.3 Member Function Documentation . . . . .	323
6.187.3.1 BootedEnter . . . . .	323
6.187.3.2 do_initialize . . . . .	323
6.187.3.3 do_pause . . . . .	324
6.187.3.4 do_reinitialize . . . . .	324
6.187.3.5 do_resume . . . . .	324
6.187.3.6 do_shutdown . . . . .	325
6.187.3.7 do_soft_initialize . . . . .	325



6.187.3.8 do_start . . . . .	325
6.187.3.9 do_stop . . . . .	326
6.187.3.10operator= . . . . .	326
6.187.3.11report . . . . .	326
6.188artdaq::RoutingManagerCore Class Reference . . . . .	327
6.188.1 Detailed Description . . . . .	328
6.188.2 Constructor & Destructor Documentation . . . . .	328
6.188.2.1 ~RoutingManagerCore . . . . .	328
6.188.3 Member Function Documentation . . . . .	328
6.188.3.1 get_update_count . . . . .	328
6.188.3.2 initialize . . . . .	328
6.188.3.3 operator= . . . . .	329
6.188.3.4 pause . . . . .	329
6.188.3.5 reinitialize . . . . .	329
6.188.3.6 report . . . . .	330
6.188.3.7 resume . . . . .	330
6.188.3.8 send_event_table . . . . .	330
6.188.3.9 shutdown . . . . .	330
6.188.3.10soft_initialize . . . . .	330
6.188.3.11start . . . . .	331
6.188.3.12stop . . . . .	331
6.189artdaq::detail::RoutingManagerModeConverter Class Reference . . . . .	331
6.189.1 Detailed Description . . . . .	332
6.189.2 Member Function Documentation . . . . .	332
6.189.2.1 routingManagerModeToString . . . . .	332
6.189.2.2 stringToRoutingManagerMode . . . . .	332
6.190artdaq::RoutingManagerPolicy Class Reference . . . . .	332
6.190.1 Detailed Description . . . . .	334
6.190.2 Constructor & Destructor Documentation . . . . .	334
6.190.2.1 RoutingManagerPolicy . . . . .	334
6.190.3 Member Function Documentation . . . . .	334
6.190.3.1 AddReceiverToken . . . . .	334
6.190.3.2 CacheHasRoute . . . . .	334
6.190.3.3 CreateRouteForSequenceID . . . . .	335
6.190.3.4 CreateRoutingTable . . . . .	335
6.190.3.5 GetCacheSize . . . . .	335
6.190.3.6 GetCurrentTable . . . . .	335

6.190.3.7 GetHeldTokenCount . . . . .	336
6.190.3.8 GetMaxNumberOfTokens . . . . .	336
6.190.3.9 GetNextSequenceID . . . . .	336
6.190.3.10GetReceiverCount . . . . .	336
6.190.3.11GetRouteForSequenceID . . . . .	336
6.190.3.12GetRoutingMode . . . . .	337
6.190.3.13GetTokensUsedSinceLastUpdate . . . . .	337
6.191artdaq::detail::RoutingPacketEntry Struct Reference . . . . .	337
6.191.1 Detailed Description . . . . .	338
6.191.2 Constructor & Destructor Documentation . . . . .	338
6.191.2.1 RoutingPacketEntry . . . . .	338
6.192artdaq::detail::RoutingPacketHeader Struct Reference . . . . .	338
6.192.1 Detailed Description . . . . .	339
6.192.2 Constructor & Destructor Documentation . . . . .	339
6.192.2.1 RoutingPacketHeader . . . . .	339
6.193artdaq::RoutingReceiverConfig Struct Reference . . . . .	339
6.193.1 Detailed Description . . . . .	339
6.194artdaq::detail::RoutingRequest Struct Reference . . . . .	340
6.194.1 Detailed Description . . . . .	340
6.194.2 Constructor & Destructor Documentation . . . . .	341
6.194.2.1 RoutingRequest . . . . .	341
6.194.2.2 RoutingRequest . . . . .	342
6.194.3 Member Function Documentation . . . . .	342
6.194.3.1 RequestModeToString . . . . .	342
6.195artdaq::detail::RoutingToken Struct Reference . . . . .	342
6.195.1 Detailed Description . . . . .	343
6.196artdaq::RTIDDS Class Reference . . . . .	343
6.196.1 Detailed Description . . . . .	344
6.196.2 Constructor & Destructor Documentation . . . . .	344
6.196.2.1 RTIDDS . . . . .	344
6.196.3 Member Function Documentation . . . . .	344
6.196.3.1 transfer_fragment_min_blocking_mode_via_DDS_ . . . . .	344
6.196.3.2 transfer_fragment_reliable_mode_via_DDS_ . . . . .	344
6.197artdaq::RTIDDSTransfer Class Reference . . . . .	344
6.197.1 Detailed Description . . . . .	345
6.197.2 Constructor & Destructor Documentation . . . . .	345
6.197.2.1 RTIDDSTransfer . . . . .	345

6.197.3 Member Function Documentation	346
6.197.3.1 isRunning	346
6.197.3.2 receiveFragment	346
6.197.3.3 transfer_fragment_min_blocking_mode	346
6.197.3.4 transfer_fragment_reliable_mode	346
6.198 FragmentBuffer_t::SequenceIDMode Struct Reference	347
6.198.1 Detailed Description	347
6.199 FragmentBuffer_t::SequenceIDMode_id Struct Reference	347
6.199.1 Detailed Description	347
6.200 FragmentBuffer_t::SequenceIDMode_MultipleIDs Struct Reference	348
6.200.1 Detailed Description	348
6.201 FragmentBuffer_t::SequenceIDMode_MultipleIDs_id Struct Reference	348
6.201.1 Detailed Description	348
6.202 art::ServicesConfig Struct Reference	348
6.202.1 Detailed Description	349
6.203 artdaq::SharedMemoryEventManager Class Reference	349
6.203.1 Detailed Description	351
6.203.2 Constructor & Destructor Documentation	351
6.203.2.1 SharedMemoryEventManager	351
6.203.3 Member Function Documentation	352
6.203.3.1 AddFragment	352
6.203.3.2 DoneWritingFragment	353
6.203.3.3 endOfData	353
6.203.3.4 endRun	353
6.203.3.5 GetArtEventCount	353
6.203.3.6 GetBroadcastKey	354
6.203.3.7 GetCurrentSubrun	354
6.203.3.8 GetDroppedDataAddress	354
6.203.3.9 GetFragmentCount	354
6.203.3.10 GetFragmentCountInBuffer	354
6.203.3.11 GetLockedBufferCount	355
6.203.3.12 GetOpenEventCount	355
6.203.3.13 GetPendingEventCount	355
6.203.3.14 GetSubrunForSequenceID	355
6.203.3.15 ReconfigureArt	356
6.203.3.16 rolloverSubrun	356
6.203.3.17 runID	356

6.203.3.18	setOverwrite	356
6.203.3.19	setRequestMode	357
6.203.3.20	ShutdownArtProcesses	358
6.203.3.21	StartArtProcess	358
6.203.3.22	startRun	358
6.203.3.23	UpdateArtConfiguration	358
6.203.3.24	WriteFragmentHeader	359
6.204	artdaq::ShmemTransfer Class Reference	360
6.204.1	Detailed Description	361
6.204.2	Constructor & Destructor Documentation	361
6.204.2.1	ShmemTransfer	361
6.204.3	Member Function Documentation	361
6.204.3.1	isRunning	361
6.204.3.2	receiveFragment	361
6.204.3.3	receiveFragmentData	362
6.204.3.4	receiveFragmentHeader	362
6.204.3.5	transfer_fragment_min_blocking_mode	362
6.204.3.6	transfer_fragment_reliable_mode	363
6.205	art::ShmemWrapper Class Reference	363
6.205.1	Detailed Description	364
6.205.2	Constructor & Destructor Documentation	364
6.205.2.1	ShmemWrapper	364
6.205.3	Member Function Documentation	364
6.205.3.1	getEventHeader	364
6.205.3.2	receiveInitMessage	364
6.205.3.3	receiveMessage	365
6.205.3.4	receiveMessages	365
6.206	artdaq::shutdown_ Class Reference	365
6.206.1	Detailed Description	366
6.206.2	Constructor & Destructor Documentation	366
6.206.2.1	shutdown_	366
6.206.3	Member Data Documentation	366
6.206.3.1	defaultTimeout	366
6.207	GenericFragmentSimulator_t::Simple Struct Reference	366
6.207.1	Detailed Description	366
6.208	CommandableFragmentGenerator_t::Simple Struct Reference	367
6.208.1	Detailed Description	367

6.209CapacityTest_policy_t::Simple Struct Reference . . . . .	367
6.209.1 Detailed Description . . . . .	367
6.210NoOp_policy_t::Simple Struct Reference . . . . .	368
6.210.1 Detailed Description . . . . .	368
6.211PreferSameHost_policy_t::Simple Struct Reference . . . . .	368
6.211.1 Detailed Description . . . . .	368
6.212RoundRobin_policy_t::Simple Struct Reference . . . . .	369
6.212.1 Detailed Description . . . . .	369
6.213RoundRobin_policy_t::Simple_id Struct Reference . . . . .	369
6.213.1 Detailed Description . . . . .	369
6.214CommandableFragmentGenerator_t::Simple_id Struct Reference . . . . .	369
6.214.1 Detailed Description . . . . .	369
6.215GenericFragmentSimulator_t::Simple_id Struct Reference . . . . .	369
6.215.1 Detailed Description . . . . .	369
6.216NoOp_policy_t::Simple_id Struct Reference . . . . .	370
6.216.1 Detailed Description . . . . .	370
6.217CapacityTest_policy_t::Simple_id Struct Reference . . . . .	370
6.217.1 Detailed Description . . . . .	370
6.218PreferSameHost_policy_t::Simple_id Struct Reference . . . . .	370
6.218.1 Detailed Description . . . . .	370
6.219FragmentBuffer_t::SingleMode Struct Reference . . . . .	370
6.219.1 Detailed Description . . . . .	371
6.220FragmentBuffer_t::SingleMode_id Struct Reference . . . . .	371
6.220.1 Detailed Description . . . . .	371
6.221FragmentBuffer_t::SingleMode_MultipleIDs Struct Reference . . . . .	371
6.221.1 Detailed Description . . . . .	371
6.222FragmentBuffer_t::SingleMode_MultipleIDs_id Struct Reference . . . . .	371
6.222.1 Detailed Description . . . . .	371
6.223FragmentBuffer_t::SingleMode_StateMachine Struct Reference . . . . .	372
6.223.1 Detailed Description . . . . .	372
6.224FragmentBuffer_t::SingleMode_StateMachine_id Struct Reference . . . . .	372
6.224.1 Detailed Description . . . . .	372
6.225artdaq::soft_init_ Class Reference . . . . .	372
6.225.1 Detailed Description . . . . .	373
6.225.2 Constructor & Destructor Documentation . . . . .	373
6.225.2.1 soft_init_ . . . . .	373
6.225.3 Member Data Documentation . . . . .	373

6.225.3.1 defaultTimeout . . . . .	373
6.225.3.2 defaultTimestamp . . . . .	373
6.226art::Source_generator< ArtdaqInputHelper< artdaq::ListenTransferWrapper > > Struct Template Reference . . . . .	373
6.226.1 Detailed Description . . . . .	374
6.227art::Source_generator< ArtdaqInputHelper< artdaq::TransferWrapper > > Struct Template Reference . . . . .	374
6.227.1 Detailed Description . . . . .	374
6.228art::Source_generator< ArtdaqInputHelper< ShmemWrapper > > Struct Template Reference . . . . .	374
6.228.1 Detailed Description . . . . .	375
6.229art::SourceConfig Struct Reference . . . . .	375
6.229.1 Detailed Description . . . . .	375
6.230artdaq::start_ Class Reference . . . . .	375
6.230.1 Detailed Description . . . . .	376
6.230.2 Constructor & Destructor Documentation . . . . .	376
6.230.2.1 start_ . . . . .	376
6.230.3 Member Data Documentation . . . . .	376
6.230.3.1 defaultTimeout . . . . .	376
6.230.3.2 defaultTimestamp . . . . .	377
6.231CommandableFragmentGenerator_t::StateMachine Struct Reference . . . . .	377
6.231.1 Detailed Description . . . . .	377
6.232CommandableFragmentGenerator_t::StateMachine_id Struct Reference . . . . .	377
6.232.1 Detailed Description . . . . .	377
6.233artdaq::StatisticsHelper Class Reference . . . . .	377
6.233.1 Detailed Description . . . . .	378
6.233.2 Member Function Documentation . . . . .	378
6.233.2.1 addMonitoredQuantityName . . . . .	378
6.233.2.2 addSample . . . . .	378
6.233.2.3 createCollectors . . . . .	379
6.233.2.4 operator= . . . . .	379
6.233.2.5 readyToReport . . . . .	379
6.233.2.6 statsRollingWindowHasMoved . . . . .	380
6.234artdaq::status_ Class Reference . . . . .	380
6.234.1 Detailed Description . . . . .	380
6.234.2 Constructor & Destructor Documentation . . . . .	380
6.234.2.1 status_ . . . . .	380
6.235artdaq::stop_ Class Reference . . . . .	381
6.235.1 Detailed Description . . . . .	381

6.235.2 Constructor & Destructor Documentation . . . . .	381
6.235.2.1 stop_ . . . . .	381
6.235.3 Member Data Documentation . . . . .	382
6.235.3.1 defaultTimeout . . . . .	382
6.235.3.2 defaultTimestamp . . . . .	382
6.236swig_artdaq Class Reference . . . . .	382
6.236.1 Detailed Description . . . . .	383
6.236.2 Constructor & Destructor Documentation . . . . .	383
6.236.2.1 swig_artdaq . . . . .	383
6.236.3 Member Function Documentation . . . . .	383
6.236.3.1 send_metric . . . . .	383
6.236.3.2 send_metric . . . . .	384
6.236.3.3 send_metric . . . . .	384
6.236.3.4 send_rate_metric . . . . .	384
6.236.3.5 send_rate_metric . . . . .	385
6.236.3.6 send_rate_metric . . . . .	386
6.236.3.7 send_sum_metric . . . . .	386
6.236.3.8 send_sum_metric . . . . .	386
6.236.3.9 send_sum_metric . . . . .	387
6.236.3.10write_debug . . . . .	388
6.236.3.11write_error . . . . .	388
6.236.3.12write_info . . . . .	388
6.236.3.13write_trace . . . . .	388
6.236.3.14write_warning . . . . .	389
6.237artdaq::TableReceiver Class Reference . . . . .	390
6.237.1 Detailed Description . . . . .	391
6.237.2 Constructor & Destructor Documentation . . . . .	391
6.237.2.1 TableReceiver . . . . .	391
6.237.3 Member Function Documentation . . . . .	391
6.237.3.1 GetRemainingRoutingTableEntries . . . . .	391
6.237.3.2 GetRoutingTableEntry . . . . .	391
6.237.3.3 GetRoutingTableEntryCount . . . . .	392
6.237.3.4 RemoveRoutingTableEntry . . . . .	392
6.238artdaq::TCPSocketTransfer Class Reference . . . . .	392
6.238.1 Detailed Description . . . . .	393
6.238.2 Constructor & Destructor Documentation . . . . .	393
6.238.2.1 TCPSocketTransfer . . . . .	393

6.238.3 Member Function Documentation	393
6.238.3.1 isRunning	393
6.238.3.2 receiveFragmentData	394
6.238.3.3 receiveFragmentHeader	394
6.238.3.4 transfer_fragment_min_blocking_mode	394
6.238.3.5 transfer_fragment_reliable_mode	395
6.239anonymous_namespace{genToArt.cc}::ThrottledGenerator Class Reference	395
6.239.1 Detailed Description	395
6.239.2 Constructor & Destructor Documentation	396
6.239.2.1 ThrottledGenerator	396
6.239.3 Member Function Documentation	396
6.239.3.1 getNext	396
6.239.3.2 numFragIDs	396
6.239.3.3 start	396
6.239.3.4 stop	397
6.240Timeout Class Reference	398
6.240.1 Detailed Description	399
6.240.2 Constructor & Destructor Documentation	399
6.240.2.1 Timeout	399
6.240.3 Member Function Documentation	399
6.240.3.1 add_periodic	399
6.240.3.2 add_periodic	399
6.240.3.3 add_periodic	400
6.240.3.4 add_relative	401
6.240.3.5 add_relative	401
6.240.3.6 cancel_timeout	401
6.240.3.7 copy_in_timeout	401
6.240.3.8 get_next_expired_timeout	402
6.240.3.9 get_next_timeout_delay	402
6.240.3.10get_next_timeout_msdlly	402
6.240.3.11is_consistent	402
6.241Timeout::timeoutspec Struct Reference	403
6.241.1 Detailed Description	403
6.242artdaq::TokenReceiver Class Reference	403
6.242.1 Detailed Description	404
6.242.2 Constructor & Destructor Documentation	404
6.242.2.1 TokenReceiver	404



6.242.3 Member Function Documentation . . . . .	405
6.242.3.1 getReceivedTokenCount . . . . .	405
6.242.3.2 setRunNumber . . . . .	405
6.242.3.3 setStatsHelper . . . . .	405
6.242.3.4 stopTokenReception . . . . .	405
6.243artdaq::TokenSender Class Reference . . . . .	406
6.243.1 Detailed Description . . . . .	406
6.243.2 Constructor & Destructor Documentation . . . . .	407
6.243.2.1 TokenSender . . . . .	407
6.243.3 Member Function Documentation . . . . .	408
6.243.3.1 GetSentTokenCount . . . . .	408
6.243.3.2 operator= . . . . .	408
6.243.3.3 RoutingTokenSendsEnabled . . . . .	408
6.243.3.4 SendRoutingToken . . . . .	408
6.243.3.5 SetRunNumber . . . . .	408
6.244artdaq::trace_get_ Class Reference . . . . .	409
6.244.1 Detailed Description . . . . .	409
6.244.2 Constructor & Destructor Documentation . . . . .	409
6.244.2.1 trace_get_ . . . . .	409
6.245artdaq::trace_set_ Class Reference . . . . .	410
6.245.1 Detailed Description . . . . .	410
6.245.2 Constructor & Destructor Documentation . . . . .	410
6.245.2.1 trace_set_ . . . . .	410
6.246artdaq::TransferInterface Class Reference . . . . .	410
6.246.1 Detailed Description . . . . .	412
6.246.2 Member Enumeration Documentation . . . . .	413
6.246.2.1 CopyStatus . . . . .	413
6.246.2.2 ReceiveReturnCode . . . . .	413
6.246.2.3 Role . . . . .	413
6.246.3 Constructor & Destructor Documentation . . . . .	413
6.246.3.1 TransferInterface . . . . .	413
6.246.4 Member Function Documentation . . . . .	414
6.246.4.1 CopyStatusToString . . . . .	414
6.246.4.2 destination_rank . . . . .	414
6.246.4.3 isRunning . . . . .	414
6.246.4.4 operator= . . . . .	414
6.246.4.5 receiveFragment . . . . .	415

6.246.4.6 receiveFragmentData . . . . .	416
6.246.4.7 receiveFragmentHeader . . . . .	416
6.246.4.8 role . . . . .	416
6.246.4.9 source_rank . . . . .	417
6.246.4.10transfer_fragment_min_blocking_mode . . . . .	417
6.246.4.11transfer_fragment_reliable_mode . . . . .	417
6.246.4.12uniqueLabel . . . . .	418
6.247art::TransferOutput Class Reference . . . . .	418
6.247.1 Detailed Description . . . . .	418
6.247.2 Constructor & Destructor Documentation . . . . .	419
6.247.2.1 TransferOutput . . . . .	419
6.247.3 Member Function Documentation . . . . .	419
6.247.3.1 SendMessage . . . . .	419
6.248artdaq::TransferTest Class Reference . . . . .	419
6.248.1 Detailed Description . . . . .	419
6.248.2 Constructor & Destructor Documentation . . . . .	420
6.248.2.1 TransferTest . . . . .	420
6.248.3 Member Function Documentation . . . . .	420
6.248.3.1 returnCode . . . . .	420
6.248.3.2 runTest . . . . .	420
6.249artdaq::TransferWrapper Class Reference . . . . .	420
6.249.1 Detailed Description . . . . .	421
6.249.2 Constructor & Destructor Documentation . . . . .	421
6.249.2.1 TransferWrapper . . . . .	421
6.249.3 Member Function Documentation . . . . .	422
6.249.3.1 getEventHeader . . . . .	422
6.249.3.2 receiveInitMessage . . . . .	422
6.249.3.3 receiveMessage . . . . .	422
6.249.3.4 receiveMessages . . . . .	422
6.250artdaq::unregister_monitor_ Class Reference . . . . .	423
6.250.1 Detailed Description . . . . .	423
6.250.2 Constructor & Destructor Documentation . . . . .	423
6.250.2.1 unregister_monitor_ . . . . .	423
6.251RoundRobin_policy_t::VerifyRMPSHaredPtr Struct Reference . . . . .	423
6.251.1 Detailed Description . . . . .	424
6.252RoundRobin_policy_t::VerifyRMPSHaredPtr_id Struct Reference . . . . .	424
6.252.1 Detailed Description . . . . .	424

6.253FragmentBuffer_t::WaitForDataBufferReady_RaceCondition Struct Reference . . . . .	424
6.253.1 Detailed Description . . . . .	424
6.254FragmentBuffer_t::WaitForDataBufferReady_RaceCondition_id Struct Reference . . . . .	425
6.254.1 Detailed Description . . . . .	425
6.255CommandableFragmentGenerator_t::WaitForStart Struct Reference . . . . .	425
6.255.1 Detailed Description . . . . .	425
6.256CommandableFragmentGenerator_t::WaitForStart_id Struct Reference . . . . .	425
6.256.1 Detailed Description . . . . .	425
6.257FragmentBuffer_t::WindowMode_Function Struct Reference . . . . .	426
6.257.1 Detailed Description . . . . .	426
6.258FragmentBuffer_t::WindowMode_Function_id Struct Reference . . . . .	426
6.258.1 Detailed Description . . . . .	426
6.259FragmentBuffer_t::WindowMode_Function_MultipleIDs Struct Reference . . . . .	426
6.259.1 Detailed Description . . . . .	427
6.260FragmentBuffer_t::WindowMode_Function_MultipleIDs_id Struct Reference . . . . .	427
6.260.1 Detailed Description . . . . .	427
6.261FragmentBuffer_t::WindowMode_RateTests Struct Reference . . . . .	427
6.261.1 Detailed Description . . . . .	427
6.262FragmentBuffer_t::WindowMode_RateTests_id Struct Reference . . . . .	427
6.262.1 Detailed Description . . . . .	427
6.263FragmentBuffer_t::WindowMode_RateTests_threaded Struct Reference . . . . .	428
6.263.1 Detailed Description . . . . .	428
6.264FragmentBuffer_t::WindowMode_RateTests_threaded_id Struct Reference . . . . .	428
6.264.1 Detailed Description . . . . .	428
6.265FragmentBuffer_t::WindowMode_RequestAfterBuffer Struct Reference . . . . .	428
6.265.1 Detailed Description . . . . .	429
6.266FragmentBuffer_t::WindowMode_RequestAfterBuffer_id Struct Reference . . . . .	429
6.266.1 Detailed Description . . . . .	429
6.267FragmentBuffer_t::WindowMode_RequestBeforeBuffer Struct Reference . . . . .	429
6.267.1 Detailed Description . . . . .	429
6.268FragmentBuffer_t::WindowMode_RequestBeforeBuffer_id Struct Reference . . . . .	429
6.268.1 Detailed Description . . . . .	429
6.269FragmentBuffer_t::WindowMode_RequestEndsAfterBuffer Struct Reference . . . . .	430
6.269.1 Detailed Description . . . . .	430
6.270FragmentBuffer_t::WindowMode_RequestEndsAfterBuffer_id Struct Reference . . . . .	430
6.270.1 Detailed Description . . . . .	430
6.271FragmentBuffer_t::WindowMode_RequestInBuffer Struct Reference . . . . .	430

6.271.1 Detailed Description	431
6.272FragmentBuffer_t::WindowMode_RequestInBuffer_id Struct Reference	431
6.272.1 Detailed Description	431
6.273FragmentBuffer_t::WindowMode_RequestOutsideBuffer Struct Reference	431
6.273.1 Detailed Description	431
6.274FragmentBuffer_t::WindowMode_RequestOutsideBuffer_id Struct Reference	431
6.274.1 Detailed Description	431
6.275FragmentBuffer_t::WindowMode_RequestStartsBeforeBuffer Struct Reference	432
6.275.1 Detailed Description	432
6.276FragmentBuffer_t::WindowMode_RequestStartsBeforeBuffer_id Struct Reference	432
6.276.1 Detailed Description	432
6.277artdaq::xmlrpc_commander Class Reference	432
6.277.1 Detailed Description	434
6.277.2 Constructor & Destructor Documentation	434
6.277.2.1 xmlrpc_commander	434
6.277.3 Member Function Documentation	434
6.277.3.1 send_init	434
6.277.3.2 send_legal_commands	434
6.277.3.3 send_meta_command	435
6.277.3.4 send_pause	435
6.277.3.5 send_register_monitor	435
6.277.3.6 send_reinit	436
6.277.3.7 send_report	436
6.277.3.8 send_resume	436
6.277.3.9 send_rollover_subrun	438
6.277.3.10send_shutdown	438
6.277.3.11send_soft_init	438
6.277.3.12send_start	440
6.277.3.13send_status	440
6.277.3.14send_stop	440
6.277.3.15send_trace_get	441
6.277.3.16send_trace_set	441
6.277.3.17send_unregister_monitor	442
<b>7 File Documentation</b>	<b>445</b>
7.1 artdaq/artdaq/DAQdata/TCP_listen_fd.hh File Reference	445
7.1.1 Detailed Description	445

7.1.2	Function Documentation	445
7.1.2.1	TCP_listen_fd	445
7.2	artdaq/artdaq/DAQdata/TCPConnect.hh File Reference	445
7.2.1	Detailed Description	446
7.2.2	Function Documentation	446
7.2.2.1	AutodetectPrivateInterface	446
7.2.2.2	GetInterfaceForNetwork	446
7.2.2.3	GetIPOfInterface	447
7.2.2.4	ResolveHost	447
7.2.2.5	ResolveHost	447
7.2.2.6	TCPConnect	448
7.3	artdaq/tools/swig_artdaq.h File Reference	448
7.3.1	Detailed Description	448

**Index****449**



# Chapter 1

## Namespace Index

### 1.1 Namespace List

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

<a href="#">anonymous_namespace{genToArt.cc}</a>	23
<a href="#">anonymous_namespace{ListenTransferWrapper.cc}</a>	24
<a href="#">anonymous_namespace{RootDAQOut_module.cc}</a>	24
<a href="#">anonymous_namespace{RootDAQOutFile.cc}</a>	24
<a href="#">anonymous_namespace{TransferWrapper.cc}</a>	25
<a href="#">anonymous_namespace{xmlrpc_commander.cc}</a>	25
<a href="#">art</a>	
Namespace used for classes that interact directly with art	25
<a href="#">artdaq</a>	
The artdaq namespace	28
<a href="#">artdaq::detail</a>	
The <a href="#">artdaq::detail</a> namespace contains internal implementation details for some classes	41





## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

art::AnalyzersConfig . . . . .	46
FragCounter_test::Apply_id . . . . .	47
FragCounter_test::ApplyWithOffset_id . . . . .	47
artdaq::art_config_file . . . . .	47
artdaq::artdaqapp . . . . .	48
ArtdaqFragmentNamingServiceInterface . . . . .	50
ArtdaqFragmentNamingService . . . . .	49
art::ArtdaqFragmentNamingServiceInterfaceConfig . . . . .	52
art::ArtdaqInputHelper< U > . . . . .	55
ArtdaqSharedMemoryServiceInterface . . . . .	66
ArtdaqGlobalsService . . . . .	52
ArtdaqSharedMemoryService . . . . .	64
art::ArtdaqSharedMemoryServiceInterfaceConfig . . . . .	68
artdaq::BoardReaderCore . . . . .	80
BOOST_AUTO_TEST_CASE_FIXTURE	
CapacityTest_policy_t::DataFlowMode . . . . .	174
CapacityTest_policy_t::RequestBasedEventBuilding . . . . .	291
CapacityTest_policy_t::Simple . . . . .	367
CommandableFragmentGenerator_t::HardwareFailure_NonThreaded . . . . .	246
CommandableFragmentGenerator_t::HardwareFailure_Threaded . . . . .	247
CommandableFragmentGenerator_t::MultipleIDs . . . . .	266
CommandableFragmentGenerator_t::Simple . . . . .	367
CommandableFragmentGenerator_t::StateMachine . . . . .	377
CommandableFragmentGenerator_t::WaitForStart . . . . .	425
FragCounter_test::Apply . . . . .	46
FragCounter_test::ApplyWithOffset . . . . .	47
FragCounter_test::Construct . . . . .	172
FragCounter_test::nSlots . . . . .	269
FragmentBuffer_t::BufferMode . . . . .	89
FragmentBuffer_t::BufferMode_KeepLatest . . . . .	89
FragmentBuffer_t::BufferMode_MultipleIDs . . . . .	90
FragmentBuffer_t::CircularBufferMode . . . . .	99
FragmentBuffer_t::CircularBufferMode_MultipleIDs . . . . .	100

FragmentBuffer_t::CircularBufferMode_RateTests	101
FragmentBuffer_t::IgnoreRequests	248
FragmentBuffer_t::IgnoreRequests_MultipleIDs	249
FragmentBuffer_t::IgnoreRequests_StateMachine	249
FragmentBuffer_t::ImproperConfiguration	250
FragmentBuffer_t::SequenceIDMode	347
FragmentBuffer_t::SequenceIDMode_MultipleIDs	348
FragmentBuffer_t::SingleMode	370
FragmentBuffer_t::SingleMode_MultipleIDs	371
FragmentBuffer_t::SingleMode_StateMachine	372
FragmentBuffer_t::WaitForDataBufferReady_RaceCondition	424
FragmentBuffer_t::WindowMode_Function	426
FragmentBuffer_t::WindowMode_Function_MultipleIDs	426
FragmentBuffer_t::WindowMode_RateTests	427
FragmentBuffer_t::WindowMode_RateTests_threaded	428
FragmentBuffer_t::WindowMode_RequestAfterBuffer	428
FragmentBuffer_t::WindowMode_RequestBeforeBuffer	429
FragmentBuffer_t::WindowMode_RequestEndsAfterBuffer	430
FragmentBuffer_t::WindowMode_RequestInBuffer	430
FragmentBuffer_t::WindowMode_RequestOutsideBuffer	431
FragmentBuffer_t::WindowMode_RequestStartsBeforeBuffer	432
GenericFragmentSimulator_t::Simple	366
NoOp_policy_t::DataFlowMode	174
NoOp_policy_t::RequestBasedEventBuilding	292
NoOp_policy_t::Simple	368
PreferSameHost_policy_t::DataFlowMode	175
PreferSameHost_policy_t::LargeMinimumParticipants	254
PreferSameHost_policy_t::ManyMissingParticipants	257
PreferSameHost_policy_t::MinimumParticipants	261
PreferSameHost_policy_t::RequestBasedEventBuilding	292
PreferSameHost_policy_t::Simple	368
RequestSender_test::Construct	173
RequestSender_test::Requests	306
RoundRobin_policy_t::DataFlowMode	175
RoundRobin_policy_t::LargeMinimumParticipants	253
RoundRobin_policy_t::ManyMissingParticipants	258
RoundRobin_policy_t::MinimumParticipants	261
RoundRobin_policy_t::RequestBasedEventBuilding	293
RoundRobin_policy_t::Simple	369
RoundRobin_policy_t::VerifyRMPSHaredPtr	423
artdaqtest::BrokenTransferTest	87
FragmentBuffer_t::BufferMode_id	89
FragmentBuffer_t::BufferMode_KeepLatest_id	90
FragmentBuffer_t::BufferMode_MultipleIDs_id	91
FragmentBuffer_t::CircularBufferMode_id	100
FragmentBuffer_t::CircularBufferMode_MultipleIDs_id	100
FragmentBuffer_t::CircularBufferMode_RateTests_id	101
artdaq::Commandable	107
artdaq::BoardReaderApp	74
artdaq::DataLoggerApp	177
artdaq::DispatcherApp	196
artdaq::EventBuilderApp	207
artdaq::RoutingManagerApp	322
artdaq::CommanderInterface	134

artdaq::xmlrpc_commander . . . . .	432
artdaq::CommandableFragmentGenerator::Config . . . . .	149
artdaq::TransferInterface::Config . . . . .	150
artdaq::artdaqapp::Config . . . . .	151
Config . . . . .	152
ArtdaqGlobalsService::Config . . . . .	152
mfplugins::ELArtdaqMetric::Config . . . . .	152
ArtdaqSharedMemoryService::Config . . . . .	154
art::Config . . . . .	155
art::RootDAQOut::Config . . . . .	155
artdaq::GenericFragmentSimulator::Config . . . . .	157
artdaq::HostMap::Config . . . . .	158
artdaq::PortManager::Config . . . . .	159
artdaqtest::BrokenTransferTest::Config . . . . .	160
artdaq::DataSenderManager::Config . . . . .	161
artdaq::RequestReceiver::Config . . . . .	163
artdaq::RequestSender::Config . . . . .	163
artdaq::TableReceiver::Config . . . . .	164
artdaq::TokenReceiver::Config . . . . .	165
artdaq::FragmentBuffer::Config . . . . .	165
artdaq::TokenSender::Config . . . . .	168
artdaq::SharedMemoryEventManager::Config . . . . .	169
artdaq::CommanderInterface::Config . . . . .	172
FragCounter_test::Construct_id . . . . .	173
RequestSender_test::Construct_id . . . . .	174
CapacityTest_policy_t::DataFlowMode_id . . . . .	176
NoOp_policy_t::DataFlowMode_id . . . . .	176
PreferSameHost_policy_t::DataFlowMode_id . . . . .	176
RoundRobin_policy_t::DataFlowMode_id . . . . .	176
artdaq::DataReceiverCore . . . . .	183
artdaq::DataLoggerCore . . . . .	181
artdaq::DispatcherCore . . . . .	200
artdaq::EventBuilderCore . . . . .	212
artdaq::DataReceiverManager . . . . .	190
artdaq::DataSenderManager . . . . .	192
DDSDataReaderListener . . . . .	
artdaq::RTIDDS::OctetsListener . . . . .	273
artdaq::DataSenderManager::DestinationsConfig . . . . .	196
EDAnalyzer . . . . .	
artdaq::EventDump . . . . .	214
artdaq::FragmentSniffer . . . . .	232
artdaq::FragmentWatcher . . . . .	236
artdaq::MissingDataCheck . . . . .	262
artdaq::PrintBuildInfo . . . . .	285
artdaq::RequestSenderModule . . . . .	310
EDFilter . . . . .	
artdaq::OffsetPrescale . . . . .	274
artdaq::RandomDelayFilter . . . . .	287
EDProducer . . . . .	
artdaq::BuildInfo< instanceName, Pkgs > . . . . .	91
anonymous_namespace{xmlrpc_commander.cc}::env_wrap . . . . .	207
art::RootDAQOut::Config::FileNameSubstitution . . . . .	215
art::FiltersConfig . . . . .	215
artdaq::detail::FragCounter . . . . .	215

artdaq::FragmentBuffer	218
artdaqtest::FragmentBufferTestGenerator	227
FragmentGenerator	
artdaq::CommandableFragmentGenerator	119
artdaq::CompositeDriver	147
artdaq::CompositeDriver	147
artdaqtest::CommandableFragmentGeneratorTest	132
artdaq::GenericFragmentSimulator	237
artdaq::FragmentReceiverManager	229
artdaq::FragmentStoreElement	234
artdaq::GenToBufferTest	241
artdaq::GetPackageBuildInfo	242
artdaq::Globals	243
CommandableFragmentGenerator_t::HardwareFailure_NonThreaded_id	246
CommandableFragmentGenerator_t::HardwareFailure_Threaded_id	247
artdaq::HostMap::HostConfig	247
artdaq::HostMap	248
FragmentBuffer_t::IgnoreRequests_id	249
FragmentBuffer_t::IgnoreRequests_MultipleIDs_id	249
FragmentBuffer_t::IgnoreRequests_StateMachine_id	250
FragmentBuffer_t::ImproperConfiguration_id	251
art::RootOutputConfig::KeysToIgnore	252
art::RootDAQOut::Config::KeysToIgnore	253
RoundRobin_policy_t::LargeMinimumParticipants_id	254
PreferSameHost_policy_t::LargeMinimumParticipants_id	254
artdaq::ListenTransferWrapper	255
PreferSameHost_policy_t::ManyMissingParticipants_id	258
RoundRobin_policy_t::ManyMissingParticipants_id	258
MessHead	259
method	
artdaq::cmd_	102
artdaq::add_config_archive_entry_	45
artdaq::clear_config_archive_	101
artdaq::init_	251
artdaq::legal_commands_	254
artdaq::meta_command_	260
artdaq::pause_	276
artdaq::register_monitor_	289
artdaq::reinit_	289
artdaq::report_	291
artdaq::resume_	312
artdaq::rollover_subrun_	313
artdaq::shutdown_	365
artdaq::soft_init_	372
artdaq::start_	375
artdaq::status_	380
artdaq::stop_	381
artdaq::trace_get_	409
artdaq::trace_set_	410
artdaq::unregister_monitor_	423
PreferSameHost_policy_t::MinimumParticipants_id	262
RoundRobin_policy_t::MinimumParticipants_id	262
CommandableFragmentGenerator_t::MultipleIDs_id	267
artdaq::NetMonHeader	267

art::RootDAQOut::Config::NewSubStringForApp . . . . .	268
FragCounter_test::nSlots_id . . . . .	270
art::OutputItem . . . . .	275
OutputModule	
art::BinaryFileOutput . . . . .	72
art::BinaryNetOutput . . . . .	73
OutputModule	
art::ArtdaqOutput . . . . .	58
art::RootNetOutput . . . . .	316
art::TransferOutput . . . . .	418
art::RootDAQOut . . . . .	314
art::OutputsConfig . . . . .	275
art::PhysicsConfig . . . . .	277
artdaq::PortManager . . . . .	277
art::ProducersConfig . . . . .	286
CapacityTest_policy_t::RequestBasedEventBuilding_id . . . . .	293
PreferSameHost_policy_t::RequestBasedEventBuilding_id . . . . .	294
RoundRobin_policy_t::RequestBasedEventBuilding_id . . . . .	294
NoOp_policy_t::RequestBasedEventBuilding_id . . . . .	294
artdaq::RequestBuffer . . . . .	294
artdaq::detail::RequestHeader . . . . .	298
artdaq::detail::RequestMessage . . . . .	299
artdaq::detail::RequestPacket . . . . .	302
artdaq::RequestReceiver . . . . .	304
RequestSender_test::Requests_id . . . . .	307
artdaq::RequestSender . . . . .	307
art::RootDAQOutFile . . . . .	315
art::RootOutputConfig . . . . .	318
artdaq::RoutingManagerCore . . . . .	327
artdaq::detail::RoutingManagerModeConverter . . . . .	331
artdaq::RoutingManagerPolicy . . . . .	332
artdaq::CapacityTestPolicy . . . . .	96
artdaq::NoOpPolicy . . . . .	268
artdaq::PreferSameHostPolicy . . . . .	283
artdaq::RoundRobinPolicy . . . . .	319
artdaq::detail::RoutingPacketEntry . . . . .	337
artdaq::detail::RoutingPacketHeader . . . . .	338
artdaq::RoutingReceiverConfig . . . . .	339
artdaq::detail::RoutingRequest . . . . .	340
artdaq::detail::RoutingToken . . . . .	342
artdaq::RTIDDS . . . . .	343
FragmentBuffer_t::SequenceIDMode_id . . . . .	347
FragmentBuffer_t::SequenceIDMode_MultipleIDs_id . . . . .	348
art::ServicesConfig . . . . .	348
SharedMemoryManager	
artdaq::SharedMemoryEventManager . . . . .	349
art::ShmemWrapper . . . . .	363
RoundRobin_policy_t::Simple_id . . . . .	369
CommandableFragmentGenerator_t::Simple_id . . . . .	369
GenericFragmentSimulator_t::Simple_id . . . . .	369
NoOp_policy_t::Simple_id . . . . .	370
CapacityTest_policy_t::Simple_id . . . . .	370
PreferSameHost_policy_t::Simple_id . . . . .	370
FragmentBuffer_t::SingleMode_id . . . . .	371

FragmentBuffer_t::SingleMode_MultipleIDs_id . . . . .	371
FragmentBuffer_t::SingleMode_StateMachine_id . . . . .	372
art::Source_generator< ArtdaqInputHelper< artdaq::ListenTransferWrapper > > . . . . .	373
art::Source_generator< ArtdaqInputHelper< artdaq::TransferWrapper > > . . . . .	374
art::Source_generator< ArtdaqInputHelper< ShmemWrapper > > . . . . .	374
art::SourceConfig . . . . .	375
CommandableFragmentGenerator_t::StateMachine_id . . . . .	377
artdaq::StatisticsHelper . . . . .	377
swig_artdaq . . . . .	382
artdaq::TableReceiver . . . . .	390
anonymous_namespace{genToArt.cc}::ThrottledGenerator . . . . .	395
Timeout . . . . .	398
Timeout::timeoutspec . . . . .	403
artdaq::TokenReceiver . . . . .	403
artdaq::TokenSender . . . . .	406
artdaq::TransferInterface . . . . .	410
artdaq::AutodetectTransfer . . . . .	69
artdaq::BundleTransfer . . . . .	93
artdaq::MulticastTransfer . . . . .	263
artdaq::NullTransfer . . . . .	270
artdaq::RTIDDSTransfer . . . . .	344
artdaq::ShmemTransfer . . . . .	360
artdaq::TCPSocketTransfer . . . . .	392
artdaq::TransferTest . . . . .	419
artdaq::TransferWrapper . . . . .	420
RoundRobin_policy_t::VerifyRMPSHaredPtr_id . . . . .	424
FragmentBuffer_t::WaitForDataBufferReady_RaceCondition_id . . . . .	425
CommandableFragmentGenerator_t::WaitForStart_id . . . . .	425
FragmentBuffer_t::WindowMode_Function_id . . . . .	426
FragmentBuffer_t::WindowMode_Function_MultipleIDs_id . . . . .	427
FragmentBuffer_t::WindowMode_RateTests_id . . . . .	427
FragmentBuffer_t::WindowMode_RateTests_threaded_id . . . . .	428
FragmentBuffer_t::WindowMode_RequestAfterBuffer_id . . . . .	429
FragmentBuffer_t::WindowMode_RequestBeforeBuffer_id . . . . .	429
FragmentBuffer_t::WindowMode_RequestEndsAfterBuffer_id . . . . .	430
FragmentBuffer_t::WindowMode_RequestInBuffer_id . . . . .	431
FragmentBuffer_t::WindowMode_RequestOutsideBuffer_id . . . . .	431
FragmentBuffer_t::WindowMode_RequestStartsBeforeBuffer_id . . . . .	432
ELdestination	
mfplugins::ELArtdaqMetric . . . . .	205

## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">artdaq::add_config_archive_entry_</a>	
Add_config_archive_entry_ Command class . . . . .	45
<a href="#">art::AnalyzersConfig</a> . . . . .	46
<a href="#">FragCounter_test::Apply</a> . . . . .	46
<a href="#">FragCounter_test::Apply_id</a> . . . . .	47
<a href="#">FragCounter_test::ApplyWithOffset</a> . . . . .	47
<a href="#">FragCounter_test::ApplyWithOffset_id</a> . . . . .	47
<a href="#">artdaq::art_config_file</a>	
Art_config_file wraps a temporary file used to configure art . . . . .	47
<a href="#">artdaq::artdaqapp</a>	
Class representing an artdaq application. Used by all "main" functions to start artdaq. . . . .	48
<a href="#">ArtdaqFragmentNamingService</a>	
<a href="#">ArtdaqFragmentNamingService</a> extends <a href="#">ArtdaqFragmentNamingServiceInterface</a> . This implementa- tion uses the default SystemTypeMap and directly assigns names based on it . . . . .	49
<a href="#">ArtdaqFragmentNamingServiceInterface</a>	
Interface for <a href="#">ArtdaqFragmentNamingService</a> . This interface is declared to art as part of the required registration of an art Service . . . . .	50
<a href="#">art::ArtdaqFragmentNamingServiceInterfaceConfig</a>	
Configuration for the <a href="#">ArtdaqFragmentNamingServiceInterface</a> . . . . .	52
<a href="#">ArtdaqGlobalsService</a>	
<a href="#">ArtdaqGlobalsService</a> extends <a href="#">ArtdaqSharedMemoryServiceInterface</a> . It manages the artdaq Global variables my_rank and app_name, and initializes MetricManager. Users should retrieve a Service- Handle to this class before using artdaq Globals to ensure the correct values are used . . . . .	52
<a href="#">art::ArtdaqInputHelper&lt; U &gt;</a>	
This template class provides a unified interface for reading data into art . . . . .	55
<a href="#">art::ArtdaqOutput</a>	
This is the base class for artdaq OutputModules, providing the serialization interface for art Events. . .	58
<a href="#">ArtdaqSharedMemoryService</a>	
<a href="#">ArtdaqSharedMemoryService</a> extends <a href="#">ArtdaqSharedMemoryServiceInterface</a> . It receives events from shared memory using SharedMemoryEventReceiver. It also manages the artdaq Global variables my_rank and app_name. Users should retrieve a ServiceHandle to this class before using artdaq Globals to ensure the correct values are used . . . . .	64

<a href="#">ArtdaqSharedMemoryServiceInterface</a>	
Interface for <a href="#">ArtdaqSharedMemoryService</a> . This interface is declared to art as part of the required registration of an art Service . . . . .	66
<a href="#">art::ArtdaqSharedMemoryServiceInterfaceConfig</a>	
Configuration for the <a href="#">ArtdaqSharedMemoryServiceInterface</a> . . . . .	68
<a href="#">artdaq::AutodetectTransfer</a>	
The <a href="#">AutodetectTransfer TransferInterface</a> plugin sets up a Shmem_transfer plugin or TCPSocket_transfer plugin depending if the source and destination are on the same host, to maximize throughput . . . . .	69
<a href="#">art::BinaryFileOutput</a>	
The <a href="#">BinaryFileOutput</a> module streams art Events to a binary file, bypassing ROOT . . . . .	72
<a href="#">art::BinaryNetOutput</a>	
An art::OutputModule which sends Fragments using DataSenderManager. This module produces output identical to that of a BoardReader, for use in systems which have multiple layers of Event-Builders . . . . .	73
<a href="#">artdaq::BoardReaderApp</a>	
<a href="#">BoardReaderApp</a> is an <a href="#">artdaq::Commandable</a> derived class which controls the <a href="#">BoardReaderCore</a> state machine . . . . .	74
<a href="#">artdaq::BoardReaderCore</a>	
<a href="#">BoardReaderCore</a> implements the state machine for the BoardReader artdaq application. It contains a <a href="#">CommandableFragmentGenerator</a> , which generates Fragments which are then sent to a <a href="#">Data-SenderManager</a> by <a href="#">BoardReaderCore</a> . . . . .	80
<a href="#">artdaqtest::BrokenTransferTest</a>	
A class which simulates several failure modes for TransferPlugins such as sender pause/restart and receiver pause/restart . . . . .	87
<a href="#">FragmentBuffer_t::BufferMode</a> . . . . .	89
<a href="#">FragmentBuffer_t::BufferMode_id</a> . . . . .	89
<a href="#">FragmentBuffer_t::BufferMode_KeepLatest</a> . . . . .	89
<a href="#">FragmentBuffer_t::BufferMode_KeepLatest_id</a> . . . . .	90
<a href="#">FragmentBuffer_t::BufferMode_MultipleIDs</a> . . . . .	90
<a href="#">FragmentBuffer_t::BufferMode_MultipleIDs_id</a> . . . . .	91
<a href="#">artdaq::BuildInfo&lt; instanceName, Pkgs &gt;</a>	
<a href="#">BuildInfo</a> is an art::EDProducer which saves information about package builds to the data file . . . . .	91
<a href="#">artdaq::BundleTransfer</a>	
The <a href="#">BundleTransfer TransferInterface</a> plugin sets up a Shmem_transfer plugin or TCPSocket_transfer plugin depending if the source and destination are on the same host, to maximize throughput . . . . .	93
<a href="#">artdaq::CapacityTestPolicy</a>	
A <a href="#">RoutingManagerPolicy</a> which tries to fully load the first receiver, then the second, and so on . . . . .	96
<a href="#">FragmentBuffer_t::CircularBufferMode</a> . . . . .	99
<a href="#">FragmentBuffer_t::CircularBufferMode_id</a> . . . . .	100
<a href="#">FragmentBuffer_t::CircularBufferMode_MultipleIDs</a> . . . . .	100
<a href="#">FragmentBuffer_t::CircularBufferMode_MultipleIDs_id</a> . . . . .	100
<a href="#">FragmentBuffer_t::CircularBufferMode_RateTests</a> . . . . .	101
<a href="#">FragmentBuffer_t::CircularBufferMode_RateTests_id</a> . . . . .	101
<a href="#">artdaq::clear_config_archive_</a>	
<a href="#">Clear_config_archive_</a> Command class . . . . .	101
<a href="#">artdaq::cmd_</a>	
The "cmd_" class serves as the base class for all artdaq's XML-RPC commands . . . . .	102
<a href="#">artdaq::Commandable</a>	
<a href="#">Commandable</a> is the base class for all artdaq components which implement the artdaq state machine . . . . .	107
<a href="#">artdaq::CommandableFragmentGenerator</a>	
<a href="#">CommandableFragmentGenerator</a> is a <a href="#">FragmentGenerator</a> -derived abstract class that defines the interface for a <a href="#">FragmentGenerator</a> designed as a state machine with start, stop, etc., transition commands . . . . .	119



<a href="#">artdaqtest::CommandableFragmentGeneratorTest</a>	
CommandableFragmentGenerator derived class for testing	132
<a href="#">artdaq::CommanderInterface</a>	
This interface defines the functions used to transfer data between artdaq applications	134
<a href="#">artdaq::CompositeDriver</a>	
CompositeDriver handles a set of lower-level generators	147
<a href="#">artdaq::CommandableFragmentGenerator::Config</a>	
Configuration of the <a href="#">CommandableFragmentGenerator</a> . May be used for parameter validation	149
<a href="#">artdaq::TransferInterface::Config</a>	
Configuration of the <a href="#">TransferInterface</a> . May be used for parameter validation	150
<a href="#">artdaq::artdaqapp::Config</a>	
Configuration of artdaqapp. May be used for parameter validation	151
<a href="#">Config</a>	
Configuration for simple_metric_sender	152
<a href="#">ArtdaqGlobalsService::Config</a>	
Allowed Configuration parameters of <a href="#">ArtdaqGlobalsService</a> . May be used for configuration validation	152
<a href="#">mfplugins::ELArtdaqMetric::Config</a>	
Configuration Parameters for <a href="#">ELArtdaqMetric</a>	152
<a href="#">ArtdaqSharedMemoryService::Config</a>	
Allowed Configuration parameters of NetMonTransportService. May be used for configuration validation	154
<a href="#">art::Config</a>	
Required configuration for art processes started by artdaq, with artdaq-specific defaults where applicable	155
<a href="#">art::RootDAQOut::Config</a>	155
<a href="#">artdaq::GenericFragmentSimulator::Config</a>	
Configuration of the <a href="#">GenericFragmentSimulator</a> . May be used for parameter validation	157
<a href="#">artdaq::HostMap::Config</a>	
Template for the host_map configuration parameter.	158
<a href="#">artdaq::PortManager::Config</a>	
Configuration of <a href="#">PortManager</a> . May be used for parameter validation	159
<a href="#">artdaqtest::BrokenTransferTest::Config</a>	
Configuration parameters for <a href="#">BrokenTransferTest</a>	160
<a href="#">artdaq::DataSenderManager::Config</a>	
Configuration of <a href="#">DataSenderManager</a> . May be used for parameter validation	161
<a href="#">artdaq::RequestReceiver::Config</a>	
Configuration of the <a href="#">RequestReceiver</a> . May be used for parameter validation	163
<a href="#">artdaq::RequestSender::Config</a>	
Configuration of the <a href="#">RequestSender</a> . May be used for parameter validation	163
<a href="#">artdaq::TableReceiver::Config</a>	
Configuration for Routing table reception	164
<a href="#">artdaq::TokenReceiver::Config</a>	
Configuration of the <a href="#">TokenReceiver</a> . May be used for parameter validation	165
<a href="#">artdaq::FragmentBuffer::Config</a>	
Configuration of the <a href="#">FragmentBuffer</a> . May be used for parameter validation	165
<a href="#">artdaq::TokenSender::Config</a>	
Configuration for Routing token sending	168
<a href="#">artdaq::SharedMemoryEventManager::Config</a>	
Configuration of the <a href="#">SharedMemoryEventManager</a> . May be used for parameter validation	169
<a href="#">artdaq::CommanderInterface::Config</a>	
Configuration of the <a href="#">CommanderInterface</a> . May be used for parameter validation	172
<a href="#">FragCounter_test::Construct</a>	172
<a href="#">RequestSender_test::Construct</a>	173
<a href="#">FragCounter_test::Construct_id</a>	173

<a href="#">RequestSender_test::Construct_id</a>	174
<a href="#">CapacityTest_policy_t::DataFlowMode</a>	174
<a href="#">NoOp_policy_t::DataFlowMode</a>	174
<a href="#">PreferSameHost_policy_t::DataFlowMode</a>	175
<a href="#">RoundRobin_policy_t::DataFlowMode</a>	175
<a href="#">CapacityTest_policy_t::DataFlowMode_id</a>	176
<a href="#">NoOp_policy_t::DataFlowMode_id</a>	176
<a href="#">PreferSameHost_policy_t::DataFlowMode_id</a>	176
<a href="#">RoundRobin_policy_t::DataFlowMode_id</a>	176
<a href="#">artdaq::DataLoggerApp</a>	
<a href="#">DataLoggerApp</a> is an <a href="#">artdaq::Commandable</a> derived class which controls the <a href="#">DataLoggerCore</a>	177
<a href="#">artdaq::DataLoggerCore</a>	
<a href="#">DataLoggerCore</a> implements the state machine for the DataLogger artdaq application. <a href="#">DataLoggerCore</a> processes incoming events in one of three roles: Data Logger, Online Monitor, or Dispatcher	181
<a href="#">artdaq::DataReceiverCore</a>	
<a href="#">DataReceiverCore</a> implements the state machine for the DataReceiver artdaq application. <a href="#">DataReceiverCore</a> receives Fragment objects from the <a href="#">DataReceiverManager</a> , and sends them to the <a href="#">EventStore</a>	183
<a href="#">artdaq::DataReceiverManager</a>	
Receives Fragment objects from one or more <a href="#">DataSenderManager</a> instances using <a href="#">TransferInterface</a> plugins <a href="#">DataReceiverManager</a> runs a reception thread for each source, and can automatically suppress reception from sources which are going faster than the others	190
<a href="#">artdaq::DataSenderManager</a>	
Sends Fragment objects using <a href="#">TransferInterface</a> plugins. Uses Routing Tables if configured, otherwise will Round-Robin Fragments to the destinations	192
<a href="#">artdaq::DataSenderManager::DestinationsConfig</a>	
Configuration for transfers to destinations	196
<a href="#">artdaq::DispatcherApp</a>	
<a href="#">DispatcherApp</a> is an <a href="#">artdaq::Commandable</a> derived class which controls the <a href="#">DispatcherCore</a>	196
<a href="#">artdaq::DispatcherCore</a>	
<a href="#">DispatcherCore</a> implements the state machine for the Dispatcher artdaq application. <a href="#">DispatcherCore</a> processes incoming events in one of three roles: Data Logger, Online Monitor, or Dispatcher	200
<a href="#">mfplugins::ELArtdaqMetric</a>	
Message Facility destination which logs messages to a TRACE buffer	205
<a href="#">anonymous_namespace{xmlrpc_commander.cc}::env_wrap</a>	
Wrapper for XMLRPC environment construction/destruction	207
<a href="#">artdaq::EventBuilderApp</a>	
<a href="#">EventBuilderApp</a> is an <a href="#">artdaq::Commandable</a> derived class which controls the <a href="#">EventBuilderCore</a>	207
<a href="#">artdaq::EventBuilderCore</a>	
<a href="#">EventBuilderCore</a> implements the state machine for the EventBuilder artdaq application. <a href="#">EventBuilderCore</a> receives Fragment objects from the <a href="#">DataReceiverManager</a> , and sends them to the <a href="#">EventStore</a>	212
<a href="#">artdaq::EventDump</a>	
Write Event information to the console	214
<a href="#">art::RootDAQOut::Config::FileNameSubstitution</a>	215
<a href="#">art::FiltersConfig</a>	215
<a href="#">artdaq::detail::FragCounter</a>	
Keep track of the count of Fragments received from a set of sources	215
<a href="#">artdaq::FragmentBuffer</a>	
<a href="#">FragmentBuffer</a> is a <a href="#">FragmentGenerator</a> -derived abstract class that defines the interface for a <a href="#">FragmentGenerator</a> designed as a state machine with start, stop, etc., transition commands	218
<a href="#">artdaqtest::FragmentBufferTestGenerator</a>	
<a href="#">CommandableFragmentGenerator</a> derived class for testing	227

<a href="#">artdaq::FragmentReceiverManager</a>	
Receives Fragment objects from one or more <a href="#">DataSenderManager</a> instances using <a href="#">TransferInterface</a> plugins DataReceiverMaanger runs a reception thread for each source, and can automatically suppress reception from sources which are going faster than the others	229
<a href="#">artdaq::FragmentSniffer</a>	
This art::EDAnalyzer plugin tries to get Fragments from each event, asserting that the correct number of Fragments were present	232
<a href="#">artdaq::FragmentStoreElement</a>	
This class contains tracking information for all Fragment objects which have been received from a specific source	234
<a href="#">artdaq::FragmentWatcher</a>	
An art::EDAnalyzer module which checks events for certain error conditions (missing fragments, empty fragments, etc)	236
<a href="#">artdaq::GenericFragmentSimulator</a>	
<a href="#">GenericFragmentSimulator</a> creates simulated Generic events, with data distributed according to a "histogram" provided in the configuration data	237
<a href="#">artdaq::GenToBufferTest</a>	
Test fixture for GenToBuffer_t	241
<a href="#">artdaq::GetPackageBuildInfo</a>	
Wrapper around the <a href="#">artdaq::GetPackageBuildInfo::getPackageBuildInfo</a> function	242
<a href="#">artdaq::Globals</a>	
The <a href="#">artdaq::Globals</a> class contains several variables which are useful across the entire artdaq system	243
<a href="#">CommandableFragmentGenerator_t::HardwareFailure_NonThreaded</a>	246
<a href="#">CommandableFragmentGenerator_t::HardwareFailure_NonThreaded_id</a>	246
<a href="#">CommandableFragmentGenerator_t::HardwareFailure_Threated</a>	247
<a href="#">CommandableFragmentGenerator_t::HardwareFailure_Threated_id</a>	247
<a href="#">artdaq::HostMap::HostConfig</a>	
Entries in the host_map should have these parameters. May be used for parameter validation	247
<a href="#">artdaq::HostMap</a>	248
<a href="#">FragmentBuffer_t::IgnoreRequests</a>	248
<a href="#">FragmentBuffer_t::IgnoreRequests_id</a>	249
<a href="#">FragmentBuffer_t::IgnoreRequests_MultipleIDs</a>	249
<a href="#">FragmentBuffer_t::IgnoreRequests_MultipleIDs_id</a>	249
<a href="#">FragmentBuffer_t::IgnoreRequests_StateMachine</a>	249
<a href="#">FragmentBuffer_t::IgnoreRequests_StateMachine_id</a>	250
<a href="#">FragmentBuffer_t::ImproperConfiguration</a>	250
<a href="#">FragmentBuffer_t::ImproperConfiguration_id</a>	251
<a href="#">artdaq::init_</a>	251
<a href="#">art::RootOutputConfig::KeysToIgnore</a>	
These keys should be ignored by the configuration validation processor	252
<a href="#">art::RootDAQOut::Config::KeysToIgnore</a>	253
<a href="#">RoundRobin_policy_t::LargeMinimumParticipants</a>	253
<a href="#">PreferSameHost_policy_t::LargeMinimumParticipants</a>	254
<a href="#">RoundRobin_policy_t::LargeMinimumParticipants_id</a>	254
<a href="#">PreferSameHost_policy_t::LargeMinimumParticipants_id</a>	254
<a href="#">artdaq::legal_commands_</a>	
<a href="#">Legal_commands_</a> Command class	254
<a href="#">artdaq::ListenTransferWrapper</a>	
<a href="#">ListenTransferWrapper</a> wraps a <a href="#">TransferInterface</a> so that it can be used in the ArtdaqInput class to make an art::Source	255
<a href="#">PreferSameHost_policy_t::ManyMissingParticipants</a>	257
<a href="#">RoundRobin_policy_t::ManyMissingParticipants</a>	258
<a href="#">PreferSameHost_policy_t::ManyMissingParticipants_id</a>	258
<a href="#">RoundRobin_policy_t::ManyMissingParticipants_id</a>	258

<a href="#">MessHead</a>	
This header is sent by the TCPSocket_transfer to allow for more efficient writev calls . . . . .	259
<a href="#">artdaq::meta_command_</a>	
Meta_command_ Command class . . . . .	260
<a href="#">PreferSameHost_policy_t::MinimumParticipants</a> . . . . .	261
<a href="#">RoundRobin_policy_t::MinimumParticipants</a> . . . . .	261
<a href="#">PreferSameHost_policy_t::MinimumParticipants_id</a> . . . . .	262
<a href="#">RoundRobin_policy_t::MinimumParticipants_id</a> . . . . .	262
<a href="#">artdaq::MissingDataCheck</a>	
Check art::Event for potentially missing data . . . . .	262
<a href="#">artdaq::MulticastTransfer</a>	
MulticastTransfer is a TransferInterface implementation plugin that transfers data using Multicast . . . . .	263
<a href="#">CommandableFragmentGenerator_t::MultipleIDs</a> . . . . .	266
<a href="#">CommandableFragmentGenerator_t::MultipleIDs_id</a> . . . . .	267
<a href="#">artdaq::NetMonHeader</a>	
Header with length information for NetMonTransport messages . . . . .	267
<a href="#">art::RootDAQOut::Config::NewSubStringForApp</a> . . . . .	268
<a href="#">artdaq::NoOpPolicy</a>	
A RoutingManagerPolicy which simply assigns Sequence IDs to tokens in the order they were received . . . . .	268
<a href="#">FragCounter_test::nSlots</a> . . . . .	269
<a href="#">FragCounter_test::nSlots_id</a> . . . . .	270
<a href="#">artdaq::NullTransfer</a>	
NullTransfer does not send or receive data, but acts as if it did . . . . .	270
<a href="#">artdaq::RTIDDS::OctetsListener</a>	
A class that reads data from DDS . . . . .	273
<a href="#">artdaq::OffsetPrescale</a> . . . . .	274
<a href="#">art::OutputItem</a> . . . . .	275
<a href="#">art::OutputsConfig</a>	
Configuration for the outputs block of artdaq art processes . . . . .	275
<a href="#">artdaq::pause_</a> . . . . .	276
<a href="#">art::PhysicsConfig</a>	
Configuration of the physics block for artdaq art processes . . . . .	277
<a href="#">artdaq::PortManager</a>	
PortManager attempts to automatically detect interfaces and ports used for the various TCP and UDP sockets used by artdaq . . . . .	277
<a href="#">artdaq::PreferSameHostPolicy</a>	
A RoutingManagerPolicy which tries to keep data on the same host. For EventBuilding mode, performs RoundRobin . . . . .	283
<a href="#">artdaq::PrintBuildInfo</a>	
An art::EDAnalyzer which prints any <a href="#">artdaq::BuildInfo</a> objects stored in the run . . . . .	285
<a href="#">art::ProducersConfig</a>	
Artdaq does not provide any producers . . . . .	286
<a href="#">artdaq::RandomDelayFilter</a>	
A filter which delays for a random amount of time, then drops a random fraction of events. Used to simulate the delays and efficiency of real filters . . . . .	287
<a href="#">artdaq::register_monitor_</a>	
Register_monitor_ Command class . . . . .	289
<a href="#">artdaq::reinit_</a> . . . . .	289
<a href="#">artdaq::report_</a>	
Report_ Command class . . . . .	291
<a href="#">CapacityTest_policy_t::RequestBasedEventBuilding</a> . . . . .	291
<a href="#">NoOp_policy_t::RequestBasedEventBuilding</a> . . . . .	292
<a href="#">PreferSameHost_policy_t::RequestBasedEventBuilding</a> . . . . .	292

RoundRobin_policy_t::RequestBasedEventBuilding	293
CapacityTest_policy_t::RequestBasedEventBuilding_id	293
PreferSameHost_policy_t::RequestBasedEventBuilding_id	294
RoundRobin_policy_t::RequestBasedEventBuilding_id	294
NoOp_policy_t::RequestBasedEventBuilding_id	294
artdaq::RequestBuffer	
Holds requests from <a href="#">RequestReceiver</a> while they are being processed	294
artdaq::detail::RequestHeader	
Header of a <a href="#">RequestMessage</a> . Contains magic bytes for validation and a count of expected Request-Packets	298
artdaq::detail::RequestMessage	
A <a href="#">RequestMessage</a> consists of a <a href="#">RequestHeader</a> and zero or more <a href="#">RequestPackets</a> . They will usually be sent in two calls to <code>send()</code>	299
artdaq::detail::RequestPacket	
The <a href="#">RequestPacket</a> contains information about a single data request	302
artdaq::RequestReceiver	
Receive data requests and make them available to <a href="#">CommandableFragmentGenerator</a> or other interested parties. Track received requests and report errors when inconsistency is detected.	304
RequestSender_test::Requests	306
RequestSender_test::Requests_id	307
artdaq::RequestSender	
The <a href="#">RequestSender</a> contains methods used to send data requests and Routing tokens	307
artdaq::RequestSenderModule	
An art::EDAnalyzer module which sends requests for events (e.g. for the Mu2e CRV system)	310
artdaq::resume_	312
artdaq::rollover_subrun_	
Rollover_subrun_ Command class	313
art::RootDAQOut	314
art::RootDAQOutFile	315
art::RootNetOutput	
An art::OutputModule which sends events using <a href="#">DataSenderManager</a> . This module is designed for transporting Fragment-wrapped art::Events after they have been read into art, for example between the EventBuilder and the Aggregator	316
art::RootOutputConfig	
Configuration for ROOT output modules	318
artdaq::RoundRobinPolicy	
A <a href="#">RoutingManagerPolicy</a> which evenly distributes Sequence IDs to all receivers. If an uneven number of tokens have been received, extra tokens are stored for the next table update	319
artdaq::RoutingManagerApp	
<a href="#">RoutingManagerApp</a> is an <a href="#">artdaq::Commandable</a> derived class which controls the <a href="#">RoutingManager-Core</a> state machine	322
artdaq::RoutingManagerCore	
<a href="#">RoutingManagerCore</a> implements the state machine for the RoutingManager artdaq application. <a href="#">RoutingManagerCore</a> collects tokens from receivers, and at regular intervals uses these tokens to build Routing Tables that are sent to the senders	327
artdaq::detail::RoutingManagerModeConverter	
Convert RoutingManagerMode to/from strings	331
artdaq::RoutingManagerPolicy	
The interface through which <a href="#">RoutingManagerCore</a> obtains Routing Tables using received Routing Tokens	332
artdaq::detail::RoutingPacketEntry	
A row of the Routing Table	337
artdaq::detail::RoutingPacketHeader	
The header of the Routing Table, containing the magic bytes and the number of entries	338

<a href="#">artdaq::RoutingReceiverConfig</a>	
Class which receives routing tables and prints updates	339
<a href="#">artdaq::detail::RoutingRequest</a>	
Represents a request sent to the RoutingManager for routing information	340
<a href="#">artdaq::detail::RoutingToken</a>	
The <a href="#">RoutingToken</a> contains the magic bytes, the rank of the token sender, and the number of slots free. This is a TCP message, so additional verification is not necessary	342
<a href="#">artdaq::RTIDDS</a>	
DDS Transport Implementation	343
<a href="#">artdaq::RTIDDSTransfer</a>	
<a href="#">RTIDDSTransfer</a> is a <a href="#">TransferInterface</a> implementation plugin that transfers data using RTI DDS	344
<a href="#">FragmentBuffer_t::SequenceIDMode</a>	347
<a href="#">FragmentBuffer_t::SequenceIDMode_id</a>	347
<a href="#">FragmentBuffer_t::SequenceIDMode_MultipleIDs</a>	348
<a href="#">FragmentBuffer_t::SequenceIDMode_MultipleIDs_id</a>	348
<a href="#">art::ServicesConfig</a>	
Configuration of the services block for artdaq art processes	348
<a href="#">artdaq::SharedMemoryEventManager</a>	
The <a href="#">SharedMemoryEventManager</a> is a <a href="#">SharedMemoryManger</a> which tracks events as they are built	349
<a href="#">artdaq::ShmemTransfer</a>	
A <a href="#">TransferInterface</a> implementation plugin that transfers data using Shared Memory	360
<a href="#">art::ShmemWrapper</a>	
This class wraps <a href="#">ArtdaqSharedMemoryService</a> so that it can act as an <a href="#">ArtdaqInput</a> template class	363
<a href="#">artdaq::shutdown_</a>	
<a href="#">Shutdown_</a> Command class	365
<a href="#">GenericFragmentSimulator_t::Simple</a>	366
<a href="#">CommandableFragmentGenerator_t::Simple</a>	367
<a href="#">CapacityTest_policy_t::Simple</a>	367
<a href="#">NoOp_policy_t::Simple</a>	368
<a href="#">PreferSameHost_policy_t::Simple</a>	368
<a href="#">RoundRobin_policy_t::Simple</a>	369
<a href="#">RoundRobin_policy_t::Simple_id</a>	369
<a href="#">CommandableFragmentGenerator_t::Simple_id</a>	369
<a href="#">GenericFragmentSimulator_t::Simple_id</a>	369
<a href="#">NoOp_policy_t::Simple_id</a>	370
<a href="#">CapacityTest_policy_t::Simple_id</a>	370
<a href="#">PreferSameHost_policy_t::Simple_id</a>	370
<a href="#">FragmentBuffer_t::SingleMode</a>	370
<a href="#">FragmentBuffer_t::SingleMode_id</a>	371
<a href="#">FragmentBuffer_t::SingleMode_MultipleIDs</a>	371
<a href="#">FragmentBuffer_t::SingleMode_MultipleIDs_id</a>	371
<a href="#">FragmentBuffer_t::SingleMode_StateMachine</a>	372
<a href="#">FragmentBuffer_t::SingleMode_StateMachine_id</a>	372
<a href="#">artdaq::soft_init_</a>	372
<a href="#">art::Source_generator&lt; ArtdaqInputHelper&lt; artdaq::ListenTransferWrapper &gt; &gt;</a>	
Trait definition (must precede source typedef)	373
<a href="#">art::Source_generator&lt; ArtdaqInputHelper&lt; artdaq::TransferWrapper &gt; &gt;</a>	
Trait definition (must precede source typedef)	374
<a href="#">art::Source_generator&lt; ArtdaqInputHelper&lt; ShmemWrapper &gt; &gt;</a>	
Trait definition (must precede source typedef)	374
<a href="#">art::SourceConfig</a>	
Configuration for the source block of artdaq art processes	375
<a href="#">artdaq::start_</a>	
Command class representing a start transition	375

<a href="#">CommandableFragmentGenerator_t::StateMachine</a>	377
<a href="#">CommandableFragmentGenerator_t::StateMachine_id</a>	377
<a href="#">artdaq::StatisticsHelper</a>	
This class manages MonitoredQuantity instances for the *Core classes	377
<a href="#">artdaq::status_</a>	
Status_ Command class	380
<a href="#">artdaq::stop_</a>	381
<a href="#">swig_artdaq</a>	
Simple class exposing methods for TRACEing and sending metrics that can be wrapped with SWIG	382
<a href="#">artdaq::TableReceiver</a>	
Sends Fragment objects using <a href="#">TransferInterface</a> plugins. Uses Routing Tables if configured, otherwise will Round-Robin Fragments to the destinations	390
<a href="#">artdaq::TCPSocketTransfer</a>	
<a href="#">TransferInterface</a> implementation plugin that sends data using TCP sockets	392
<a href="#">anonymous_namespace{genToArt.cc}::ThrottledGenerator</a>	
ThrottledGenerator: ensure that we only get one fragment per type at a time from the generator	395
<a href="#">Timeout</a>	
Performs registered actions at specified intervals	398
<a href="#">Timeout::timeoutspec</a>	
Specification for a <a href="#">Timeout</a> function	403
<a href="#">artdaq::TokenReceiver</a>	
Receives event builder "free buffer" tokens and adds them to a specified RoutingPolicy	403
<a href="#">artdaq::TokenSender</a>	
The <a href="#">TokenSender</a> contains methods used to send data requests and Routing tokens	406
<a href="#">artdaq::trace_get_</a>	
Trace_get_ Command class	409
<a href="#">artdaq::trace_set_</a>	
Trace_set_ Command class	410
<a href="#">artdaq::TransferInterface</a>	
This interface defines the functions used to transfer data between artdaq applications	410
<a href="#">art::TransferOutput</a>	
An art::OutputModule which sends events using DataSenderManager. This module is designed for transporting Fragment-wrapped art::Events after they have been read into art, for example between the EventBuilder and the Aggregator	418
<a href="#">artdaq::TransferTest</a>	
Test a set of <a href="#">TransferInterface</a> plugins	419
<a href="#">artdaq::TransferWrapper</a>	
TransferWrapper wraps a <a href="#">TransferInterface</a> so that it can be used in the ArtdaqInput class to make an art::Source	420
<a href="#">artdaq::unregister_monitor_</a>	
Unregister_monitor_ Command class	423
<a href="#">RoundRobin_policy_t::VerifyRMPSHaredPtr</a>	423
<a href="#">RoundRobin_policy_t::VerifyRMPSHaredPtr_id</a>	424
<a href="#">FragmentBuffer_t::WaitForDataBufferReady_RaceCondition</a>	424
<a href="#">FragmentBuffer_t::WaitForDataBufferReady_RaceCondition_id</a>	425
<a href="#">CommandableFragmentGenerator_t::WaitForStart</a>	425
<a href="#">CommandableFragmentGenerator_t::WaitForStart_id</a>	425
<a href="#">FragmentBuffer_t::WindowMode_Function</a>	426
<a href="#">FragmentBuffer_t::WindowMode_Function_id</a>	426
<a href="#">FragmentBuffer_t::WindowMode_Function_MultipleIDs</a>	426
<a href="#">FragmentBuffer_t::WindowMode_Function_MultipleIDs_id</a>	427
<a href="#">FragmentBuffer_t::WindowMode_RateTests</a>	427
<a href="#">FragmentBuffer_t::WindowMode_RateTests_id</a>	427
<a href="#">FragmentBuffer_t::WindowMode_RateTests_threaded</a>	428

FragmentBuffer_t::WindowMode_RateTests_threaded_id . . . . .	428
FragmentBuffer_t::WindowMode_RequestAfterBuffer . . . . .	428
FragmentBuffer_t::WindowMode_RequestAfterBuffer_id . . . . .	429
FragmentBuffer_t::WindowMode_RequestBeforeBuffer . . . . .	429
FragmentBuffer_t::WindowMode_RequestBeforeBuffer_id . . . . .	429
FragmentBuffer_t::WindowMode_RequestEndsAfterBuffer . . . . .	430
FragmentBuffer_t::WindowMode_RequestEndsAfterBuffer_id . . . . .	430
FragmentBuffer_t::WindowMode_RequestInBuffer . . . . .	430
FragmentBuffer_t::WindowMode_RequestInBuffer_id . . . . .	431
FragmentBuffer_t::WindowMode_RequestOutsideBuffer . . . . .	431
FragmentBuffer_t::WindowMode_RequestOutsideBuffer_id . . . . .	431
FragmentBuffer_t::WindowMode_RequestStartsBeforeBuffer . . . . .	432
FragmentBuffer_t::WindowMode_RequestStartsBeforeBuffer_id . . . . .	432
artdaq::xmlrpc_commander	
Serves as the XMLRPC server run in each artdaq application . . . . .	432



## Chapter 4

# File Index

### 4.1 File List

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

artdaq/artdaq/Application/ <b>BoardReaderApp.cc</b>	??
artdaq/artdaq/Application/ <b>BoardReaderApp.hh</b>	??
artdaq/artdaq/Application/ <b>BoardReaderCore.cc</b>	??
artdaq/artdaq/Application/ <b>BoardReaderCore.hh</b>	??
artdaq/artdaq/Application/ <b>Commandable.cc</b>	??
artdaq/artdaq/Application/ <b>Commandable.hh</b>	??
artdaq/artdaq/Application/ <b>DataLoggerApp.cc</b>	??
artdaq/artdaq/Application/ <b>DataLoggerApp.hh</b>	??
artdaq/artdaq/Application/ <b>DataLoggerCore.cc</b>	??
artdaq/artdaq/Application/ <b>DataLoggerCore.hh</b>	??
artdaq/artdaq/Application/ <b>DataReceiverCore.cc</b>	??
artdaq/artdaq/Application/ <b>DataReceiverCore.hh</b>	??
artdaq/artdaq/Application/ <b>DispatcherApp.cc</b>	??
artdaq/artdaq/Application/ <b>DispatcherApp.hh</b>	??
artdaq/artdaq/Application/ <b>DispatcherCore.cc</b>	??
artdaq/artdaq/Application/ <b>DispatcherCore.hh</b>	??
artdaq/artdaq/Application/ <b>EventBuilderApp.cc</b>	??
artdaq/artdaq/Application/ <b>EventBuilderApp.hh</b>	??
artdaq/artdaq/Application/ <b>EventBuilderCore.cc</b>	??
artdaq/artdaq/Application/ <b>EventBuilderCore.hh</b>	??
artdaq/artdaq/Application/ <b>LoadParameterSet.hh</b>	??
artdaq/artdaq/Application/ <b>RoutingManagerApp.cc</b>	??
artdaq/artdaq/Application/ <b>RoutingManagerApp.hh</b>	??
artdaq/artdaq/Application/ <b>RoutingManagerCore.cc</b>	??
artdaq/artdaq/Application/ <b>RoutingManagerCore.hh</b>	??
artdaq/artdaq/Application/ <b>TaskType.hh</b>	??
artdaq/artdaq/ArtModules/ <b>ArtdaqBuildInfo_module.cc</b>	??
artdaq/artdaq/ArtModules/ <b>ArtdaqFragmentNamingService.h</b>	??
artdaq/artdaq/ArtModules/ <b>ArtdaqFragmentNamingService_service.cc</b>	??
artdaq/artdaq/ArtModules/ <b>ArtdaqGlobalsService.h</b>	??
artdaq/artdaq/ArtModules/ <b>ArtdaqGlobalsService_service.cc</b>	??
artdaq/artdaq/ArtModules/ <b>ArtdaqInput_source.cc</b>	??
artdaq/artdaq/ArtModules/ <b>ArtdaqInputHelper.hh</b>	??
artdaq/artdaq/ArtModules/ <b>ArtdaqMetric_mfPlugin.cc</b>	??

artdaq/artdaq/ArtModules/ <b>ArtdaqOutput</b> .hh	??
artdaq/artdaq/ArtModules/ <b>ArtdaqSharedMemoryService</b> _service.cc	??
artdaq/artdaq/ArtModules/ <b>ArtdaqSharedMemoryServiceInterface</b> .h	??
artdaq/artdaq/ArtModules/ <b>BinaryFileOutput</b> _module.cc	??
artdaq/artdaq/ArtModules/ <b>BinaryNetOutput</b> _module.cc	??
artdaq/artdaq/ArtModules/ <b>BuildInfo</b> _module.hh	??
artdaq/artdaq/ArtModules/ <b>EventDump</b> _module.cc	??
artdaq/artdaq/ArtModules/ <b>FragmentWatcher</b> _module.cc	??
artdaq/artdaq/ArtModules/ <b>InputUtilities</b> .hh	??
artdaq/artdaq/ArtModules/ <b>MissingDataCheck</b> _module.cc	??
artdaq/artdaq/ArtModules/ <b>OffsetPrescale</b> _module.cc	??
artdaq/artdaq/ArtModules/ <b>PrintBuildInfo</b> _module.cc	??
artdaq/artdaq/ArtModules/ <b>RandomDelayFilter</b> _module.cc	??
artdaq/artdaq/ArtModules/ <b>RequestSender</b> _module.cc	??
artdaq/artdaq/ArtModules/ <b>RootNetOutput</b> _module.cc	??
artdaq/artdaq/ArtModules/ <b>TransferInput</b> _source.cc	??
artdaq/artdaq/ArtModules/ <b>TransferListenerInput</b> _source.cc	??
artdaq/artdaq/ArtModules/ <b>TransferOutput</b> _module.cc	??
artdaq/artdaq/ArtModules/detail/ <b>ArtConfig</b> .hh	??
artdaq/artdaq/ArtModules/detail/ <b>ListenTransferWrapper</b> .cc	??
artdaq/artdaq/ArtModules/detail/ <b>ListenTransferWrapper</b> .hh	??
artdaq/artdaq/ArtModules/detail/ <b>ShmemWrapper</b> .cc	??
artdaq/artdaq/ArtModules/detail/ <b>ShmemWrapper</b> .hh	??
artdaq/artdaq/ArtModules/detail/ <b>TransferWrapper</b> .cc	??
artdaq/artdaq/ArtModules/detail/ <b>TransferWrapper</b> .hh	??
artdaq/artdaq/ArtModules/RootDAQOutput-s124/ <b>RootDAQOut</b> _module.cc	??
artdaq/artdaq/ArtModules/RootDAQOutput-s124/ <b>RootDAQOutFile</b> .cc	??
artdaq/artdaq/ArtModules/RootDAQOutput-s124/ <b>RootDAQOutFile</b> .h	??
artdaq/artdaq/BuildInfo/ <b>GetPackageBuildInfo</b> .hh	??
artdaq/artdaq/DAQdata/ <b>GenericFragmentSimulator</b> .cc	??
artdaq/artdaq/DAQdata/ <b>GenericFragmentSimulator</b> .hh	??
artdaq/artdaq/DAQdata/ <b>GenericFragmentSimulator_generator</b> .cc	??
artdaq/artdaq/DAQdata/ <b>Globals</b> .cc	??
artdaq/artdaq/DAQdata/ <b>Globals</b> .hh	??
artdaq/artdaq/DAQdata/ <b>HostMap</b> .hh	??
artdaq/artdaq/DAQdata/ <b>NetMonHeader</b> .hh	??
artdaq/artdaq/DAQdata/ <b>PortManager</b> .cc	??
artdaq/artdaq/DAQdata/ <b>PortManager</b> .hh	??
artdaq/artdaq/DAQdata/ <b>TCP_listen_fd</b> .cc	??
artdaq/artdaq/DAQdata/ <b>TCP_listen_fd</b> .hh	445
artdaq/artdaq/DAQdata/ <b>TCPConnect</b> .cc	??
artdaq/artdaq/DAQdata/ <b>TCPConnect</b> .hh	445
artdaq/artdaq/DAQrate/ <b>DataReceiverManager</b> .cc	??
artdaq/artdaq/DAQrate/ <b>DataReceiverManager</b> .hh	??
artdaq/artdaq/DAQrate/ <b>DataSenderManager</b> .cc	??
artdaq/artdaq/DAQrate/ <b>DataSenderManager</b> .hh	??
artdaq/artdaq/DAQrate/ <b>FragmentBuffer</b> .cc	??
artdaq/artdaq/DAQrate/ <b>FragmentBuffer</b> .hh	??
artdaq/artdaq/DAQrate/ <b>FragmentReceiverManager</b> .cc	??
artdaq/artdaq/DAQrate/ <b>FragmentReceiverManager</b> .hh	??
artdaq/artdaq/DAQrate/ <b>RequestBuffer</b> .cc	??
artdaq/artdaq/DAQrate/ <b>RequestBuffer</b> .hh	??
artdaq/artdaq/DAQrate/ <b>SharedMemoryEventManager</b> .cc	??
artdaq/artdaq/DAQrate/ <b>SharedMemoryEventManager</b> .hh	??

artdaq/artdaq/DAQrate/ <b>StatisticsHelper.cc</b>	??
artdaq/artdaq/DAQrate/ <b>StatisticsHelper.hh</b>	??
artdaq/artdaq/DAQrate/ <b>TransferTest.cc</b>	??
artdaq/artdaq/DAQrate/ <b>TransferTest.hh</b>	??
artdaq/artdaq/DAQrate/detail/ <b>FragCounter.hh</b>	??
artdaq/artdaq/DAQrate/detail/ <b>RequestMessage.hh</b>	??
artdaq/artdaq/DAQrate/detail/ <b>RequestReceiver.cc</b>	??
artdaq/artdaq/DAQrate/detail/ <b>RequestReceiver.hh</b>	??
artdaq/artdaq/DAQrate/detail/ <b>RequestSender.cc</b>	??
artdaq/artdaq/DAQrate/detail/ <b>RequestSender.hh</b>	??
artdaq/artdaq/DAQrate/detail/ <b>RoutingPacket.hh</b>	??
artdaq/artdaq/DAQrate/detail/ <b>TableReceiver.cc</b>	??
artdaq/artdaq/DAQrate/detail/ <b>TableReceiver.hh</b>	??
artdaq/artdaq/DAQrate/detail/ <b>TokenReceiver.cc</b>	??
artdaq/artdaq/DAQrate/detail/ <b>TokenReceiver.hh</b>	??
artdaq/artdaq/DAQrate/detail/ <b>TokenSender.cc</b>	??
artdaq/artdaq/DAQrate/detail/ <b>TokenSender.hh</b>	??
artdaq/artdaq/ExternalComms/ <b>CommanderInterface.cc</b>	??
artdaq/artdaq/ExternalComms/ <b>CommanderInterface.hh</b>	??
artdaq/artdaq/ExternalComms/ <b>MakeCommanderPlugin.cc</b>	??
artdaq/artdaq/ExternalComms/ <b>MakeCommanderPlugin.hh</b>	??
artdaq/artdaq/ExternalComms/ <b>xmlrpc_commander.cc</b>	??
artdaq/artdaq/Generators/ <b>CommandableFragmentGenerator.cc</b>	??
artdaq/artdaq/Generators/ <b>CommandableFragmentGenerator.hh</b>	??
artdaq/artdaq/Generators/ <b>CompositeDriver.hh</b>	??
artdaq/artdaq/Generators/ <b>CompositeDriver_generator.cc</b>	??
artdaq/artdaq/Generators/ <b>GeneratorMacros.hh</b>	??
artdaq/artdaq/Generators/ <b>makeCommandableFragmentGenerator.cc</b>	??
artdaq/artdaq/Generators/ <b>makeCommandableFragmentGenerator.hh</b>	??
artdaq/artdaq/RoutingPolicies/ <b>CapacityTest_policy.cc</b>	??
artdaq/artdaq/RoutingPolicies/ <b>makeRoutingManagerPolicy.cc</b>	??
artdaq/artdaq/RoutingPolicies/ <b>makeRoutingManagerPolicy.hh</b>	??
artdaq/artdaq/RoutingPolicies/ <b>NoOp_policy.cc</b>	??
artdaq/artdaq/RoutingPolicies/ <b>PolicyMacros.hh</b>	??
artdaq/artdaq/RoutingPolicies/ <b>PreferSameHost_policy.cc</b>	??
artdaq/artdaq/RoutingPolicies/ <b>RoundRobin_policy.cc</b>	??
artdaq/artdaq/RoutingPolicies/ <b>RoutingManagerPolicy.cc</b>	??
artdaq/artdaq/RoutingPolicies/ <b>RoutingManagerPolicy.hh</b>	??
artdaq/artdaq/RTIDDS/ <b>RTIDDS.cc</b>	??
artdaq/artdaq/RTIDDS/ <b>RTIDDS.hh</b>	??
artdaq/artdaq/TransferPlugins/ <b>Autodetect_transfer.cc</b>	??
artdaq/artdaq/TransferPlugins/ <b>Bundle_transfer.cc</b>	??
artdaq/artdaq/TransferPlugins/ <b>MakeTransferPlugin.cc</b>	??
artdaq/artdaq/TransferPlugins/ <b>MakeTransferPlugin.hh</b>	??
artdaq/artdaq/TransferPlugins/ <b>Multicast_transfer.cc</b>	??
artdaq/artdaq/TransferPlugins/ <b>Null_transfer.cc</b>	??
artdaq/artdaq/TransferPlugins/ <b>RTIDDS_transfer.cc</b>	??
artdaq/artdaq/TransferPlugins/ <b>Shmem_transfer.cc</b>	??
artdaq/artdaq/TransferPlugins/ <b>ShmemTransfer.cc</b>	??
artdaq/artdaq/TransferPlugins/ <b>ShmemTransfer.hh</b>	??
artdaq/artdaq/TransferPlugins/ <b>TCPSocket_transfer.cc</b>	??
artdaq/artdaq/TransferPlugins/ <b>TCPSocketTransfer.cc</b>	??
artdaq/artdaq/TransferPlugins/ <b>TCPSocketTransfer.hh</b>	??
artdaq/artdaq/TransferPlugins/ <b>TransferInterface.cc</b>	??

artdaq/artdaq/TransferPlugins/ <b>TransferInterface.hh</b>	??
artdaq/artdaq/TransferPlugins/detail/ <b>SRSockets.hh</b>	??
artdaq/artdaq/TransferPlugins/detail/ <b>Timeout.cc</b>	??
artdaq/artdaq/TransferPlugins/detail/ <b>Timeout.hh</b>	??
artdaq/doc/sample_MPMR_runs/ <b>cpiA.cc</b>	??
artdaq/doc/sample_MPMR_runs/ <b>cpiB.cc</b>	??
artdaq/proto/ <b>artdaq.cc</b>	??
artdaq/proto/ <b>artdaqapp.hh</b>	??
artdaq/proto/ <b>boardreader.cc</b>	??
artdaq/proto/ <b>datalogger.cc</b>	??
artdaq/proto/ <b>dispatcher.cc</b>	??
artdaq/proto/ <b>driver.cc</b>	??
artdaq/proto/ <b>eventbuilder.cc</b>	??
artdaq/proto/ <b>fhicl_test.cc</b>	??
artdaq/proto/ <b>PrintSharedMemory.cc</b>	??
artdaq/proto/ <b>requestReceiver.cc</b>	??
artdaq/proto/ <b>requestSender.cc</b>	??
artdaq/proto/ <b>routing_manager.cc</b>	??
artdaq/proto/ <b>routingReceiver.cc</b>	??
artdaq/proto/ <b>simple_metric_sender.cc</b>	??
artdaq/proto/ <b>tracemf.cc</b>	??
artdaq/proto/ <b>transfer_plugin_receiver.cc</b>	??
artdaq/proto/ <b>transfer_plugin_sender.cc</b>	??
artdaq/test/ArtModules/ <b>daq_flow_t.cc</b>	??
artdaq/test/ArtModules/ <b>FragmentSniffer_module.cc</b>	??
artdaq/test/ArtModules/ <b>reconfigure_t.cc</b>	??
artdaq/test/DAQdata/ <b>GenericFragmentSimulator_t.cc</b>	??
artdaq/test/DAQrate/ <b>DataReceiverManager_t.cc</b>	??
artdaq/test/DAQrate/ <b>FragCounter_t.cc</b>	??
artdaq/test/DAQrate/ <b>FragmentBuffer_t.cc</b>	??
artdaq/test/DAQrate/ <b>GenToBuffer_t.cc</b>	??
artdaq/test/DAQrate/ <b>RequestSender_t.cc</b>	??
artdaq/test/DAQrate/ <b>SharedMemoryEventManager_t.cc</b>	??
artdaq/test/DAQrate/ <b>TokenSender_t.cc</b>	??
artdaq/test/ExternalComms/ <b>commander_test.cc</b>	??
artdaq/test/Generators/ <b>CommandableFragmentGenerator_t.cc</b>	??
artdaq/test/RoutingPolicies/ <b>CapacityTest_policy_t.cc</b>	??
artdaq/test/RoutingPolicies/ <b>NoOp_policy_t.cc</b>	??
artdaq/test/RoutingPolicies/ <b>PreferSameHost_policy_t.cc</b>	??
artdaq/test/RoutingPolicies/ <b>RoundRobin_policy_t.cc</b>	??
artdaq/test/TransferPlugins/ <b>broken_transfer_driver.cc</b>	??
artdaq/test/TransferPlugins/ <b>BrokenTransferTest.cc</b>	??
artdaq/test/TransferPlugins/ <b>BrokenTransferTest.hh</b>	??
artdaq/test/TransferPlugins/ <b>transfer_driver.cc</b>	??
artdaq/tools/ <b>genToArt.cc</b>	??
artdaq/tools/ <b>periodic_cmd_stats.cc</b>	??
artdaq/tools/ <b>StateResponder.cc</b>	??
artdaq/tools/ <b>swig_artdaq.cc</b>	??
artdaq/tools/ <b>swig_artdaq.h</b>	448

## Chapter 5

# Namespace Documentation

### 5.1 anonymous\_namespace{genToArt.cc} Namespace Reference

#### Classes

- class [ThrottledGenerator](#)

*ThrottledGenerator*: ensure that we only get one fragment per type at a time from the generator.

#### Functions

- int [process\\_cmd\\_line](#) (int argc, char \*\*argv, bpo::variables\_map &vm)  
*Process the command line.*
- int [process\\_data](#) (fhicl::ParameterSet const &pset)  
*Run the test, instantiating configured generators and an EventStore.*

#### 5.1.1 Function Documentation

##### 5.1.1.1 int anonymous\_namespace{genToArt.cc}::process\_cmd\_line ( int argc, char \*\* argv, bpo::variables\_map & vm )

Process the command line.

#### Parameters

	<i>argc</i>	Number of arguments
	<i>argv</i>	Array of arguments as strings
out	<i>vm</i>	Output boost::program_options::variables_map

#### Returns

0 if success, -1 if exception, 1 if help was requested, and 2 if missing required arguments

Definition at line 46 of file genToArt.cc.

##### 5.1.1.2 int anonymous\_namespace{genToArt.cc}::process\_data ( fhicl::ParameterSet const & pset )

Run the test, instantiating configured generators and an EventStore.

## Parameters

<i>pset</i>	ParameterSet used to configure genToArt
-------------	---

## Returns

Art return code, of 15 if EventStore::endOfData fails

```
* genToArt accepts the following Parameters:
* "reset_sequenceID" (Default: true): Set the sequence IDs on generated Fragment objects to the expected value
* "genToArt" (REQUIRED): FHiCL table containing genToArt parameters
* "fragment_receivers" (REQUIRED): List of FHiCL tables configuring the Fragment receivers
*   Each table should contain parameter "generator", the FragmentGenerator plugin to load, and any other parameters
* "event_builder" (Default: {}): ParameterSet for EventStore. See documentation for configuration parameters.
* "run_number" (REQUIRED): Run number to use
* "events_to_generate" (Default: -1): Number of events to generate
*
*
```

Definition at line 224 of file genToArt.cc.

## 5.2 anonymous\_namespace{ListenTransferWrapper.cc} Namespace Reference

## Variables

- volatile std::sig\_atomic\_t [gListenSignalStatus](#) = 0  
*Stores singal from signal handler.*

## 5.3 anonymous\_namespace{RootDAQOut\_module.cc} Namespace Reference

## Functions

- bool **maxCriterionSpecified** (art::ClosingCriteria const &cc)
- auto **shouldFastClone** (bool const fastCloningSet, bool const fastCloning, bool const wantAllEvents, art::ClosingCriteria const &cc)

## Variables

- string const **dev\_null** {"/dev/null"}

## 5.4 anonymous\_namespace{RootDAQOutFile.cc} Namespace Reference

## Functions

- void **create\_table** (sqlite3 \*const db, string const &name, vector< string > const &columns, string const &suffix={})
- void **insert\_eventRanges\_row** (sqlite3\_stmt \*stmt, art::SubRunNumber\_t const sr, art::EventNumber\_t const b, art::EventNumber\_t const e)
- void **insert\_rangeSets\_eventSets\_row** (sqlite3\_stmt \*stmt, unsigned const rsid, unsigned const esid)

- unsigned **getNewRangeSetID** (sqlite3 \*db, art::BranchType const bt, art::RunNumber\_t const r)
- vector< unsigned > **getExistingRangeSetIDs** (sqlite3 \*db, art::RangeSet const &rs)
- void **insertIntoEventRanges** (sqlite3 \*db, art::RangeSet const &rs)
- void **insertIntoJoinTable** (sqlite3 \*db, art::BranchType const bt, unsigned const rsID, vector< unsigned > const &eventRangesIDs)
- void **maybeInvalidateRangeSet** (BranchType const bt, art::RangeSet const &principalRS, art::RangeSet &productRS)
- template<BranchType BT>  
art::RangeSet **getRangeSet** (art::OutputHandle const &oh, art::RangeSet const &principalRS, bool const producedInThisProcess)
- template<BranchType BT>  
void **setProductRangeSetID** (art::RangeSet const &rs, sqlite3 \*db, art::EDProduct \*product, map< unsigned, unsigned > &checksumToIndexLookup)

## 5.5 anonymous\_namespace{TransferWrapper.cc} Namespace Reference

### Variables

- volatile std::sig\_atomic\_t **gSignalStatus** = 0  
*Stores signal from signal handler.*

## 5.6 anonymous\_namespace{xmlrpc\_commander.cc} Namespace Reference

### Classes

- class **env\_wrap**  
*Wrapper for XMLRPC environment construction/destruction*

## 5.7 art Namespace Reference

Namespace used for classes that interact directly with art.

### Classes

- struct **Source\_generator< ArtdaqInputHelper< ShmemWrapper > >**  
*Trait definition (must precede source typedef).*
- class **ArtdaqInputHelper**  
*This template class provides a unified interface for reading data into art.*
- class **ArtdaqOutput**  
*This is the base class for artdaq OutputModules, providing the serialization interface for art Events.*
- class **BinaryFileOutput**  
*The BinaryFileOutput module streams art Events to a binary file, bypassing ROOT.*
- class **BinaryNetOutput**  
*An art::OutputModule which sends Fragments using DataSenderManager. This module produces output identical to that of a BoardReader, for use in systems which have multiple layers of EventBuilders.*
- struct **ArtdaqFragmentNamingServiceInterfaceConfig**

Configuration for the [ArtdaqFragmentNamingServiceInterface](#)

- struct [ArtdaqSharedMemoryServiceInterfaceConfig](#)

Configuration for the [ArtdaqSharedMemoryServiceInterface](#)

- struct [ServicesConfig](#)

Configuration of the services block for artdaq art processes

- struct [AnalyzersConfig](#)
- struct [ProducersConfig](#)

Artdaq does not provide any producers.

- struct [FiltersConfig](#)
- struct [PhysicsConfig](#)

Configuration of the physics block for artdaq art processes

- struct [RootOutputConfig](#)

Configuration for ROOT output modules

- struct [OutputsConfig](#)

Configuration for the outputs block of artdaq art processes

- struct [SourceConfig](#)

Configuration for the source block of artdaq art processes

- struct [Config](#)

Required configuration for art processes started by artdaq, with artdaq-specific defaults where applicable

- class [ShmemWrapper](#)

This class wraps [ArtdaqSharedMemoryService](#) so that it can act as an [ArtdaqInput](#) template class.

- class [RootDAQOut](#)
- struct [OutputItem](#)
- class [RootDAQOutFile](#)
- class [RootNetOutput](#)

An [art::OutputModule](#) which sends events using [DataSenderManager](#). This module is designed for transporting Fragment-wrapped [art::Events](#) after they have been read into art, for example between the [EventBuilder](#) and the [Aggregator](#).

- struct [Source\\_generator< ArtdaqInputHelper< artdaq::TransferWrapper > >](#)

Trait definition (must precede source typedef).

- struct [Source\\_generator< ArtdaqInputHelper< artdaq::ListenTransferWrapper > >](#)

Trait definition (must precede source typedef).

- class [TransferOutput](#)

An [art::OutputModule](#) which sends events using [DataSenderManager](#). This module is designed for transporting Fragment-wrapped [art::Events](#) after they have been read into art, for example between the [EventBuilder](#) and the [Aggregator](#).

## Typedefs

- using [ArtdaqInput](#) = [art::Source< ArtdaqInputHelper< ShmemWrapper > >](#)

[ArtdaqInput](#) is an [art::Source](#) using an [ArtdaqInputHelper](#)-wrapped [ShmemWrapper](#).

- using [TransferInput](#) = [art::Source< ArtdaqInputHelper< artdaq::TransferWrapper > >](#)

[TransferInput](#) is an [art::Source](#) using the [artdaq::TransferWrapper](#) class as the data source.



## Functions

- `template<typename T >`  
`T * ReadObjectAny (const std::unique_ptr< TBufferFile > &infile, const std::string &className, const std::string &callerName)`  
*ReadObjectAny reads data from a TBufferFile and casts it to the given type.*
- `template<typename T >`  
`void printProcessHistoryID (const std::string &label, const T &object)`  
*Print the processHistoryID from the object.*
- `template<typename T >`  
`void printProcessMap (const T &mappable, const std::string &description)`  
*Print data from a map-like class.*

### 5.7.1 Detailed Description

Namespace used for classes that interact directly with art.

### 5.7.2 Typedef Documentation

5.7.2.1 `using art::TransferInput = typedef art::Source<ArtdaqInputHelper<artdaq::ListenTransferWrapper>>`

TransferInput is an art::Source using the [artdaq::TransferWrapper](#) class as the data source.

TransferInput is an art::Source using the [artdaq::ListenTransferWrapper](#) class as the data source.

Definition at line 20 of file TransferInput\_source.cc.

### 5.7.3 Function Documentation

5.7.3.1 `template<typename T > void art::printProcessHistoryID ( const std::string & label, const T & object )`

Print the processHistoryID from the object.

Template Parameters

<i>T</i>	Type of the object
----------	--------------------

Parameters

<i>label</i>	Label for the object
<i>object</i>	Object to print processHistoryID from

Definition at line 64 of file InputUtilities.hh.

5.7.3.2 `template<typename T > void art::printProcessMap ( const T & mappable, const std::string & description )`

Print data from a map-like class.

## Template Parameters

<i>T</i>	Type of the class
----------	-------------------

## Parameters

<i>mappable</i>	Map-like class to print
<i>description</i>	Description of the map-like class

Definition at line 87 of file InputUtilities.hh.

5.7.3.3 `template<typename T> T* art::ReadObjectAny ( const std::unique_ptr< TBufferFile > & infile, const std::string & className, const std::string & callerName )`

ReadObjectAny reads data from a TBufferFile and casts it to the given type.

## Template Parameters

<i>T</i>	The type of the data being read
----------	---------------------------------

## Parameters

<i>infile</i>	A pointer to the TBufferFile being read
<i>className</i>	Name of the class to retrieve (must be in ROOT dictionary)
<i>callerName</i>	Name of the calling class, for logging purposes

## Returns

Pointer to object of type T

Definition at line 31 of file InputUtilities.hh.

## 5.8 artdaq Namespace Reference

The artdaq namespace.

## Namespaces

- [detail](#)

The `artdaq::detail` namespace contains internal implementation details for some classes.

## Classes

- class [BoardReaderApp](#)  
*BoardReaderApp* is an `artdaq::Commandable` derived class which controls the `BoardReaderCore` state machine.
- class [BoardReaderCore](#)  
*BoardReaderCore* implements the state machine for the BoardReader artdaq application. It contains a `Commandable-FragmentGenerator`, which generates Fragments which are then sent to a `DataSenderManager` by `BoardReaderCore`.
- class [Commandable](#)  
*Commandable* is the base class for all artdaq components which implement the artdaq state machine.
- class [DataLoggerApp](#)

- DataLoggerApp* is an [artdaq::Commandable](#) derived class which controls the [DataLoggerCore](#).
- class [DataLoggerCore](#)

*DataLoggerCore* implements the state machine for the DataLogger artdaq application. [DataLoggerCore](#) processes incoming events in one of three roles: Data Logger, Online Monitor, or Dispatcher.
  - class [DataReceiverCore](#)

*DataReceiverCore* implements the state machine for the DataReceiver artdaq application. [DataReceiverCore](#) receives Fragment objects from the [DataReceiverManager](#), and sends them to the EventStore.
  - class [DispatcherApp](#)

*DispatcherApp* is an [artdaq::Commandable](#) derived class which controls the [DispatcherCore](#).
  - class [DispatcherCore](#)

*DispatcherCore* implements the state machine for the Dispatcher artdaq application. [DispatcherCore](#) processes incoming events in one of three roles: Data Logger, Online Monitor, or Dispatcher.
  - class [EventBuilderApp](#)

*EventBuilderApp* is an [artdaq::Commandable](#) derived class which controls the [EventBuilderCore](#).
  - class [EventBuilderCore](#)

*EventBuilderCore* implements the state machine for the EventBuilder artdaq application. [EventBuilderCore](#) receives Fragment objects from the [DataReceiverManager](#), and sends them to the EventStore.
  - class [RoutingManagerApp](#)

*RoutingManagerApp* is an [artdaq::Commandable](#) derived class which controls the [RoutingManagerCore](#) state machine.
  - class [RoutingManagerCore](#)

*RoutingManagerCore* implements the state machine for the RoutingManager artdaq application. [RoutingManagerCore](#) collects tokens from receivers, and at regular intervals uses these tokens to build Routing Tables that are sent to the senders.
  - class [BuildInfo](#)

*BuildInfo* is an [art::EDProducer](#) which saves information about package builds to the data file.
  - class [ListenTransferWrapper](#)

*ListenTransferWrapper* wraps a [TransferInterface](#) so that it can be used in the [ArtdaqInput](#) class to make an [art::Source](#).
  - class [TransferWrapper](#)

*TransferWrapper* wraps a [TransferInterface](#) so that it can be used in the [ArtdaqInput](#) class to make an [art::Source](#).
  - class [EventDump](#)

Write Event information to the console.
  - class [FragmentWatcher](#)

An [art::EDAnalyzer](#) module which checks events for certain error conditions (missing fragments, empty fragments, etc)
  - class [MissingDataCheck](#)

Check [art::Event](#) for potentially missing data.
  - class [OffsetPrescale](#)
  - class [PrintBuildInfo](#)

An [art::EDAnalyzer](#) which prints any [artdaq::BuildInfo](#) objects stored in the run.
  - class [RandomDelayFilter](#)

A filter which delays for a random amount of time, then drops a random fraction of events. Used to simulate the delays and efficiency of real filters.
  - class [RequestSenderModule](#)

An [art::EDAnalyzer](#) module which sends requests for events (e.g. for the Mu2e CRV system)
  - struct [GetPackageBuildInfo](#)

Wrapper around the [artdaq::GetPackageBuildInfo::getPackageBuildInfo](#) function.
  - class [GenericFragmentSimulator](#)

*GenericFragmentSimulator* creates simulated Generic events, with data distributed according to a "histogram" provided in the configuration data.
  - class [Globals](#)

The [artdaq::Globals](#) class contains several variables which are useful across the entire artdaq system.

- struct [HostMap](#)

- struct [NetMonHeader](#)

Header with length information for NetMonTransport messages.

- class [PortManager](#)

[PortManager](#) attempts to automatically detect interfaces and ports used for the various TCP and UDP sockets used by artdaq.

- class [DataReceiverManager](#)

Receives Fragment objects from one or more [DataSenderManager](#) instances using [TransferInterface](#) plugins. [DataReceiverManager](#) runs a reception thread for each source, and can automatically suppress reception from sources which are going faster than the others.

- class [DataSenderManager](#)

Sends Fragment objects using [TransferInterface](#) plugins. Uses Routing Tables if configured, otherwise will Round-Robin Fragments to the destinations.

- class [RequestReceiver](#)

Receive data requests and make them available to [CommandableFragmentGenerator](#) or other interested parties. Track received requests and report errors when inconsistency is detected.

- class [RequestSender](#)

The [RequestSender](#) contains methods used to send data requests and Routing tokens.

- class [TableReceiver](#)

Sends Fragment objects using [TransferInterface](#) plugins. Uses Routing Tables if configured, otherwise will Round-Robin Fragments to the destinations.

- class [TokenReceiver](#)

Receives event builder "free buffer" tokens and adds them to a specified RoutingPolicy.

- class [TokenSender](#)

The [TokenSender](#) contains methods used to send data requests and Routing tokens.

- class [FragmentBuffer](#)

[FragmentBuffer](#) is a [FragmentGenerator](#)-derived abstract class that defines the interface for a [FragmentGenerator](#) designed as a state machine with start, stop, etc., transition commands.

- class [FragmentReceiverManager](#)

Receives Fragment objects from one or more [DataSenderManager](#) instances using [TransferInterface](#) plugins. [DataReceiverManager](#) runs a reception thread for each source, and can automatically suppress reception from sources which are going faster than the others.

- class [FragmentStoreElement](#)

This class contains tracking information for all Fragment objects which have been received from a specific source.

- class [RequestBuffer](#)

Holds requests from [RequestReceiver](#) while they are being processed.

- class [art\\_config\\_file](#)

[art\\_config\\_file](#) wraps a temporary file used to configure art

- class [SharedMemoryEventManager](#)

The [SharedMemoryEventManager](#) is a [SharedMemoryManager](#) which tracks events as they are built.

- class [StatisticsHelper](#)

This class manages [MonitoredQuantity](#) instances for the \*Core classes.

- class [TransferTest](#)

Test a set of [TransferInterface](#) plugins.

- class [CommanderInterface](#)

This interface defines the functions used to transfer data between artdaq applications.

- class [xmlrpc\\_commander](#)

The [xmlrpc\\_commander](#) class serves as the XMLRPC server run in each artdaq application.

- class [cmd\\_](#)

*The "cmd\_" class serves as the base class for all artdaq's XML-RPC commands.*

- class [init\\_](#)
- class [soft\\_init\\_](#)
- class [reinit\\_](#)
- class [start\\_](#)

*Command class representing a start transition.*

- class [pause\\_](#)
- class [resume\\_](#)
- class [stop\\_](#)
- class [shutdown\\_](#)

*[shutdown\\_](#) Command class*

- class [status\\_](#)

*[status\\_](#) Command class*

- class [report\\_](#)

*[report\\_](#) Command class*

- class [legal\\_commands\\_](#)

*[legal\\_commands\\_](#) Command class*

- class [register\\_monitor\\_](#)

*[register\\_monitor\\_](#) Command class*

- class [unregister\\_monitor\\_](#)

*[unregister\\_monitor\\_](#) Command class*

- class [trace\\_set\\_](#)

*[trace\\_set\\_](#) Command class*

- class [trace\\_get\\_](#)

*[trace\\_get\\_](#) Command class*

- class [meta\\_command\\_](#)

*[meta\\_command\\_](#) Command class*

- class [rollover\\_subrun\\_](#)

*[rollover\\_subrun\\_](#) Command class*

- class [add\\_config\\_archive\\_entry\\_](#)

*[add\\_config\\_archive\\_entry\\_](#) Command class*

- class [clear\\_config\\_archive\\_](#)

*[clear\\_config\\_archive\\_](#) Command class*

- class [CommandableFragmentGenerator](#)

*[CommandableFragmentGenerator](#) is a [FragmentGenerator](#)-derived abstract class that defines the interface for a [FragmentGenerator](#) designed as a state machine with start, stop, etc., transition commands.*

- class [CompositeDriver](#)

*[CompositeDriver](#) handles a set of lower-level generators.*

- class [CapacityTestPolicy](#)

*A [RoutingManagerPolicy](#) which tries to fully load the first receiver, then the second, and so on.*

- class [NoOpPolicy](#)

*A [RoutingManagerPolicy](#) which simply assigns Sequence IDs to tokens in the order they were received.*

- class [PreferSameHostPolicy](#)

*A [RoutingManagerPolicy](#) which tries to keep data on the same host. For EventBuilding mode, performs RoundRobin.*

- class [RoundRobinPolicy](#)

*A [RoutingManagerPolicy](#) which evenly distributes Sequence IDs to all receivers. If an uneven number of tokens have been received, extra tokens are stored for the next table update.*

- class [RoutingManagerPolicy](#)  
The interface through which [RoutingManagerCore](#) obtains Routing Tables using received Routing Tokens.
- class [RTIDDS](#)  
DDS Transport Implementation.
- class [AutodetectTransfer](#)  
The [AutodetectTransfer TransferInterface](#) plugin sets up a [Shmem\\_transfer](#) plugin or [TCPSocket\\_transfer](#) plugin depending if the source and destination are on the same host, to maximize throughput.
- class [BundleTransfer](#)  
The [BundleTransfer TransferInterface](#) plugin sets up a [Shmem\\_transfer](#) plugin or [TCPSocket\\_transfer](#) plugin depending if the source and destination are on the same host, to maximize throughput.
- class [MulticastTransfer](#)  
[MulticastTransfer](#) is a [TransferInterface](#) implementation plugin that transfers data using Multicast.
- class [NullTransfer](#)  
[NullTransfer](#) does not send or receive data, but acts as if it did.
- class [RTIDDSTransfer](#)  
[RTIDDSTransfer](#) is a [TransferInterface](#) implementation plugin that transfers data using RTI DDS.
- class [ShmemTransfer](#)  
A [TransferInterface](#) implementation plugin that transfers data using Shared Memory.
- class [TCPSocketTransfer](#)  
[TransferInterface](#) implementation plugin that sends data using TCP sockets.
- class [TransferInterface](#)  
This interface defines the functions used to transfer data between artdaq applications.
- class [artdaqapp](#)  
Class representing an artdaq application. Used by all "main" functions to start artdaq.
- struct [RoutingReceiverConfig](#)  
Class which receives routing tables and prints updates.
- class [FragmentSniffer](#)  
This `art::EDAnalyzer` plugin tries to get Fragments from each event, asserting that the correct number of Fragments were present.
- class [GenToBufferTest](#)  
Test fixture for `GenToBuffer_t`.

## Typedefs

- typedef [artdaq::BuildInfo](#)  
<&[instanceName](#),  
[artdaqcore::GetPackageBuildInfo](#),  
[artdaqutilities::GetPackageBuildInfo](#),  
[artdaq::GetPackageBuildInfo](#) > [ArtdaqBuildInfo](#)  
Specialized [artdaq::BuildInfo](#) object for the Artdaq build info.
- typedef `std::map< int,`  
`std::string >` [hostMap\\_t](#)  
The `host_map` is a map associating ranks with `artdaq::DestinationInfo` objects.
- typedef `std::unique_ptr`  
< [artdaq::CommandableFragmentGenerator](#) > [makeFunc\\_t](#) (`fhicl::ParameterSet const &ps`)  
Constructs a [CommandableFragmentGenerator](#) instance, and returns a pointer to it.

## Enumerations

- enum [RequestMode](#) {  
**Single, Buffer, Window, SequenceID,**  
**Ignored** }

*The RequestMode enumeration contains the possible ways which FragmentBuffer responds to data requests.*

## Functions

- std::vector< fhicl::ParameterSet > [MakeHostMapPset](#) (std::map< int, std::string > input)  
*Create a list of [HostMap::HostConfig](#) ParameterSets from a hostMap\_t map*
- hostMap\_t [MakeHostMap](#) (fhicl::ParameterSet const &pset, hostMap\_t map=hostMap\_t())  
*Make a hostMap\_t from a [HostMap::Config](#) ParameterSet*
- std::unique\_ptr  
< [artdaq::CommanderInterface](#) > [MakeCommanderPlugin](#) (const fhicl::ParameterSet &pset, [artdaq::Commandable](#) &commandable)  
*Load a [CommanderInterface](#) plugin.*
- std::string [exception\\_msg](#) (const std::runtime\_error &er, const std::string &helpText="execute request")  
*Write an exception message.*
- std::string [exception\\_msg](#) (const art::Exception &er, const std::string &helpText)  
*Write an exception message.*
- std::string [exception\\_msg](#) (const cet::exception &er, const std::string &helpText)  
*Write an exception message.*
- std::string [exception\\_msg](#) (const boost::exception &er, const std::string &helpText)  
*Write an exception message.*
- std::string [exception\\_msg](#) (const std::exception &er, const std::string &helpText)  
*Write an exception message.*
- std::string [exception\\_msg](#) (const std::string &erText, const std::string &helpText)  
*Write an exception message.*
- template<>  
std::string [cmd\\_::getParam](#)< std::string > (const xmlrpc\_c::paramList &paramList, int index)  
*Get a parameter from the parameter list.*
- template<>  
art::RunID [cmd\\_::getParam](#)< art::RunID > (const xmlrpc\_c::paramList &paramList, int index)  
*Get a parameter from the parameter list.*
- template<>  
fhicl::ParameterSet [cmd\\_::getParam](#)< fhicl::ParameterSet > (const xmlrpc\_c::paramList &paramList, int index)  
*Get a parameter from the parameter list.*
- std::unique\_ptr  
< [CommandableFragmentGenerator](#) > [makeCommandableFragmentGenerator](#) (std::string const &generator\_plugin\_spec, fhicl::ParameterSet const &ps)  
*Load a [CommandableFragmentGenerator](#) plugin.*
- std::shared\_ptr  
< [RoutingManagerPolicy](#) > [makeRoutingManagerPolicy](#) (std::string const &policy\_plugin\_spec, fhicl::ParameterSet const &ps)  
*Load a [RoutingManagerPolicy](#) plugin.*
- std::unique\_ptr  
< [artdaq::TransferInterface](#) > [MakeTransferPlugin](#) (const fhicl::ParameterSet &pset, const std::string &plugin\_label, [TransferInterface::Role](#) role)  
*Load a [TransferInterface](#) plugin.*

## Variables

- static std::string `instanceName` = "ArtdaqBuildInfo"  
Name of this [BuildInfo](#) instance.

### 5.8.1 Detailed Description

The artdaq namespace. Namespace used to differentiate the artdaq version of [GetPackageBuildInfo](#) from other versions present in the system.

### 5.8.2 Typedef Documentation

5.8.2.1 `typedef std::unique_ptr< artdaq::RoutingManagerPolicy > artdaq::makeFunc_t(fhicl::ParameterSet const &ps)`

Constructs a [CommandableFragmentGenerator](#) instance, and returns a pointer to it.

Constructs a [RoutingManagerPolicy](#) instance, and returns a pointer to it.

#### Parameters

<i>ps</i>	Parameter set for initializing the <a href="#">CommandableFragmentGenerator</a>
-----------	---

#### Returns

A smart pointer to the [CommandableFragmentGenerator](#)

#### Parameters

<i>ps</i>	Parameter set for initializing the <a href="#">RoutingManagerPolicy</a>
-----------	---

#### Returns

A smart pointer to the [RoutingManagerPolicy](#)

Definition at line 20 of file GeneratorMacros.hh.

### 5.8.3 Function Documentation

5.8.3.1 `template<> art::RunID artdaq::cmd_::getParam< art::RunID > ( const xmlrpc_c::paramList & paramList, int index )`

Get a parameter from the parameter list.

#### Parameters

<i>paramList</i>	The parameter list
<i>index</i>	Index of the parameter in the parameter list

#### Returns

The requested parameter

This specialized `cmd_getParam` for the `art::RunID` type

Definition at line 646 of file xmlrpc\_commander.cc.



5.8.3.2 `template<> fhicl::ParameterSet artdaq::cmd_::getParam< fhicl::ParameterSet > ( const xmlrpc_c::paramList & paramList, int index )`

Get a parameter from the parameter list.

## Parameters

<i>paramList</i>	The parameter list
<i>index</i>	Index of the parameter in the parameter list

## Returns

The requested parameter

This specialized cmd\_getParam for the fhicl::ParameterSet type

Definition at line 677 of file xmlrpc\_commander.cc.

**5.8.3.3** `template<> std::string artdaq::cmd_::getParam< std::string > ( const xmlrpc_c::paramList & paramList, int index )`

Get a parameter from the parameter list.

## Parameters

<i>paramList</i>	The parameter list
<i>index</i>	Index of the parameter in the parameter list

## Returns

The requested parameter

This specialized cmd\_getParam for the std::string type

Definition at line 630 of file xmlrpc\_commander.cc.

**5.8.3.4** `std::string artdaq::exception_msg ( const std::runtime_error & er, const std::string & helpText = "execute request" )`

Write an exception message.

## Parameters

<i>er</i>	A std::runtime_error to print
<i>helpText</i>	Additional information about the exception context. Default: "execute request"

## Returns

Exception message

Definition at line 375 of file xmlrpc\_commander.cc.

**5.8.3.5** `std::string artdaq::exception_msg ( const art::Exception & er, const std::string & helpText )`

Write an exception message.

## Parameters

<i>er</i>	An art::Exception to print
<i>helpText</i>	Additional information about the exception context

## Returns

Exception message

Definition at line 395 of file xmlrpc\_commander.cc.

5.8.3.6 `std::string artdaq::exception_msg ( const cet::exception & er, const std::string & helpText )`

Write an exception message.

## Parameters

<i>er</i>	A cet::exception to print
<i>helpText</i>	Additional information about the exception context

## Returns

Exception message

Definition at line 415 of file xmlrpc\_commander.cc.

5.8.3.7 `std::string artdaq::exception_msg ( const boost::exception & er, const std::string & helpText )`

Write an exception message.

## Parameters

<i>er</i>	A boost::exception to print
<i>helpText</i>	Additional information about the exception context

## Returns

Exception message

Definition at line 435 of file xmlrpc\_commander.cc.

5.8.3.8 `std::string artdaq::exception_msg ( const std::exception & er, const std::string & helpText )`

Write an exception message.

## Parameters

<i>er</i>	A std::exception to print
<i>helpText</i>	Additional information about the exception context

## Returns

Exception message

Definition at line 451 of file xmlrpc\_commander.cc.

5.8.3.9 `std::string artdaq::exception_msg ( const std::string & erText, const std::string & helpText )`

Write an exception message.

## Parameters

<i>erText</i>	A std::string to print
<i>helpText</i>	Additional information about the exception context

## Returns

Exception message

Definition at line 468 of file xmlrpc\_commander.cc.

5.8.3.10 `std::unique_ptr< artdaq::CommandableFragmentGenerator > artdaq::makeCommandableFragmentGenerator ( std::string const & generator_plugin_spec, fhicl::ParameterSet const & ps )`

Load a [CommandableFragmentGenerator](#) plugin.

## Parameters

<i>generator_plugin_spec</i>	Name of the plugin
<i>ps</i>	ParameterSet used to configure the plugin

## Returns

Pointer to the new plugin instance

Definition at line 6 of file makeCommandableFragmentGenerator.cc.

5.8.3.11 `std::unique_ptr< CommanderInterface > artdaq::MakeCommanderPlugin ( const fhicl::ParameterSet & pset, artdaq::Commandable & commandable )`

Load a [CommanderInterface](#) plugin.

## Parameters

<i>pset</i>	ParameterSet used to configure the <a href="#">CommanderInterface</a>
<i>commandable</i>	<a href="#">artdaq::Commandable</a> object to send transition commands to

## Returns

Pointer to the new [CommanderInterface](#) instance

Definition at line 11 of file MakeCommanderPlugin.cc.

5.8.3.12 `hostMap_t artdaq::MakeHostMap ( fhicl::ParameterSet const & pset, hostMap_t map = hostMap_t () )`  
[inline]

Make a hostMap\_t from a [HostMap::Config](#) ParameterSet

## Parameters

<i>pset</i>	fhicl::ParameterSet containing a <a href="#">HostMap::Config</a>
<i>map</i>	Input map for consistency checking (Default: <a href="#">hostMap_t()</a> )

## Returns

hostMap\_t object

Definition at line 68 of file HostMap.hh.

5.8.3.13 `std::vector<fhicl::ParameterSet> artdaq::MakeHostMapPset ( std::map< int, std::string > input )` `[inline]`

Create a list of [HostMap::HostConfig](#) ParameterSets from a hostMap\_t map

## Parameters

<i>input</i>	Input hostMap_t
--------------	-----------------

## Returns

std::vector containing [HostMap::HostConfig](#) ParameterSets

Definition at line 49 of file HostMap.hh.

5.8.3.14 `std::shared_ptr< artdaq::RoutingManagerPolicy > artdaq::makeRoutingManagerPolicy ( std::string const & policy_plugin_spec, fhicl::ParameterSet const & ps )`

Load a [RoutingManagerPolicy](#) plugin.

## Parameters

<i>policy_plugin_spec</i>	Name of the <a href="#">RoutingManagerPolicy</a>
<i>ps</i>	ParameterSet used to configure the <a href="#">RoutingManagerPolicy</a>

## Returns

std::shared\_ptr<RoutingManagerPolicy> to the new [RoutingManagerPolicy](#) instance

Definition at line 12 of file makeRoutingManagerPolicy.cc.

5.8.3.15 `std::unique_ptr< TransferInterface > artdaq::MakeTransferPlugin ( const fhicl::ParameterSet & pset, const std::string & plugin_label, TransferInterface::Role role )`

Load a [TransferInterface](#) plugin.

## Parameters

<i>pset</i>	ParameterSet used to configure the <a href="#">TransferInterface</a>
-------------	--

<i>plugin_label</i>	Name of the plugin
<i>role</i>	Whether the <a href="#">TransferInterface</a> should be configured as kSend or kReceive

#### Returns

Pointer to the new [TransferInterface](#) instance

Definition at line 13 of file MakeTransferPlugin.cc.

## 5.9 artdaq::detail Namespace Reference

The [artdaq::detail](#) namespace contains internal implementation details for some classes.

### Classes

- class [FragCounter](#)  
*Keep track of the count of Fragments received from a set of sources.*
- struct [RequestPacket](#)  
*The [RequestPacket](#) contains information about a single data request.*
- struct [RequestHeader](#)  
*Header of a [RequestMessage](#). Contains magic bytes for validation and a count of expected RequestPackets.*
- class [RequestMessage](#)  
*A [RequestMessage](#) consists of a [RequestHeader](#) and zero or more RequestPackets. They will usually be sent in two calls to send()*
- class [RoutingManagerModeConverter](#)  
*Convert RoutingManagerMode to/from strings.*
- struct [RoutingPacketEntry](#)  
*A row of the Routing Table.*
- struct [RoutingPacketHeader](#)  
*The header of the Routing Table, containing the magic bytes and the number of entries.*
- struct [RoutingRequest](#)  
*Represents a request sent to the RoutingManager for routing information.*
- struct [RoutingToken](#)  
*The [RoutingToken](#) contains the magic bytes, the rank of the token sender, and the number of slots free. This is a TCP message, so additional verification is not necessary.*

### Typedefs

- using [RoutingPacket](#) = std::vector< [RoutingPacketEntry](#) >  
*A RoutingPacket is simply a vector of [RoutingPacketEntry](#) objects. It is not suitable for network transmission, rather a [RoutingPacketHeader](#) should be sent, followed by &RoutingPacket.at(0) (the physical storage of the vector)*

## Enumerations

- enum [TaskType](#) : int {  
**BoardReaderTask** = 1, **EventBuilderTask** = 2, **DataLoggerTask** = 3, **DispatcherTask** = 4,  
**RoutingManagerTask** = 5, **UnknownTask** }  
*The types of applications in artdaq.*
- enum [RequestMessageMode](#) : uint8\_t { [RequestMessageMode::Normal](#) = 0, [RequestMessageMode::EndOfRun](#) = 1 }  
*Mode used to indicate current run conditions to the request receiver.*
- enum [RoutingManagerMode](#) : uint8\_t { [RoutingManagerMode::EventBuilding](#), [RoutingManagerMode::Request-BasedEventBuilding](#), [RoutingManagerMode::DataFlow](#), **INVALID** }  
*Mode indicating whether the RoutingManager is routing events by Sequence ID or by Send Count.*

## Functions

- [TaskType](#) [StringToTaskType](#) (std::string const &task)  
*Convert a string to a TaskType (Used by [artdaq.cc](#))*
- [TaskType](#) [IntToTaskType](#) (int const &task)  
*Convert an integer to the corresponding TaskType*
- std::string [TaskTypeToString](#) ([TaskType](#) const &task)  
*Convert a TaskType to string representation*
- TraceStreamer & **operator**<< (TraceStreamer &x, [TaskType](#) r)
- std::ostream & **operator**<< (std::ostream &o, [RequestMessageMode](#) m)  
*Converts the RequestMessageMode to a string and sends it to the output stream.*

### 5.9.1 Detailed Description

The [artdaq::detail](#) namespace contains internal implementation details for some classes.

### 5.9.2 Enumeration Type Documentation

#### 5.9.2.1 enum [artdaq::detail::RequestMessageMode](#) : uint8\_t [[strong](#)]

Mode used to indicate current run conditions to the request receiver.

#### Enumerator

**Normal** Normal running.

**EndOfRun** End of Run mode (Used to end request processing on receiver)

Definition at line 22 of file RequestMessage.hh.

#### 5.9.2.2 enum [artdaq::detail::RoutingManagerMode](#) : uint8\_t [[strong](#)]

Mode indicating whether the RoutingManager is routing events by Sequence ID or by Send Count.

#### Enumerator

**EventBuilding** Multiple sources sending to a single destination. RoutingManager pushes table updates to all senders.



**RequestBasedEventBuilding** Multiple sources sending to a single destination. Table updates are triggered by senders requesting routing information.

**DataFlow** One source sending to one destination (i.e. moving around completed events). Uses request-based routing.

Definition at line 27 of file RoutingPacket.hh.

### 5.9.3 Function Documentation

#### 5.9.3.1 TaskType artdaq::detail::IntToTaskType ( int const & task ) [inline]

Convert an integer to the corresponding TaskType

Parameters

<i>task</i>	Enumeration identifier of Task
-------------	--------------------------------

Returns

Corresponding TaskType or TaskType::UnknownTask if no match

Definition at line 51 of file TaskType.hh.

#### 5.9.3.2 std::ostream& artdaq::detail::operator<< ( std::ostream & o, RequestMessageMode m ) [inline]

Converts the RequestMessageMode to a string and sends it to the output stream.

Parameters

<i>o</i>	Stream to send string to
<i>m</i>	RequestMessageMode to convert to string

Returns

*o* with string sent to it

Definition at line 34 of file RequestMessage.hh.

#### 5.9.3.3 TaskType artdaq::detail::StringToTaskType ( std::string const & task ) [inline]

Convert a string to a TaskType (Used by [artdaq.cc](#))

Parameters

<i>task</i>	Name of the task
-------------	------------------

Returns

Corresponding TaskType or TaskType::UnknownTask if no match

Definition at line 30 of file TaskType.hh.

#### 5.9.3.4 std::string artdaq::detail::TaskTypeToString ( TaskType const & task ) [inline]

Convert a TaskType to string representation

**Parameters**

<i>task</i>	TaskType to convert
-------------	---------------------

**Returns**

String represenation of Task name

Definition at line 64 of file TaskType.hh.

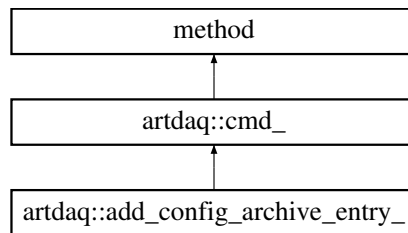
## Chapter 6

# Class Documentation

### 6.1 artdaq::add\_config\_archive\_entry\_ Class Reference

[add\\_config\\_archive\\_entry\\_](#) Command class

Inheritance diagram for artdaq::add\_config\_archive\_entry\_:



#### Public Member Functions

- [add\\_config\\_archive\\_entry\\_](#) (xmlrpc\_commander &c)  
*[add\\_config\\_archive\\_entry\\_](#) Constructor*

#### Additional Inherited Members

##### 6.1.1 Detailed Description

[add\\_config\\_archive\\_entry\\_](#) Command class

Definition at line 1238 of file xmlrpc\_commander.cc.

##### 6.1.2 Constructor & Destructor Documentation

6.1.2.1 artdaq::add\_config\_archive\_entry\_::add\_config\_archive\_entry\_ ( xmlrpc\_commander & c ) [inline]

[add\\_config\\_archive\\_entry\\_](#) Constructor

## Parameters

<code>c</code>	<code>xmlrpc_commander</code> to send transition commands to
----------------	--

Definition at line 1245 of file `xmlrpc_commander.cc`.

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

- `artdaq/artdaq/ExternalComms/xmlrpc_commander.cc`

## 6.2 art::AnalyzersConfig Struct Reference

```
#include <artdaq/ArtModules/detail/ArtConfig.hh>
```

### 6.2.1 Detailed Description

**Todo** Fill in artdaq-provided Analyzer modules

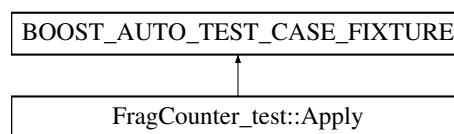
Definition at line 59 of file `ArtConfig.hh`.

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

- `artdaq/artdaq/ArtModules/detail/ArtConfig.hh`

## 6.3 FragCounter\_test::Apply Struct Reference

Inheritance diagram for `FragCounter_test::Apply`:



### Public Member Functions

- `void test_method ()`

### 6.3.1 Detailed Description

Definition at line 22 of file `FragCounter_t.cc`.

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

- `artdaq/test/DAQrate/FragCounter_t.cc`

## 6.4 FragCounter\_test::Apply\_id Struct Reference

### 6.4.1 Detailed Description

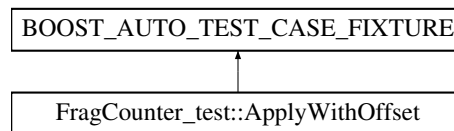
Definition at line 22 of file FragCounter\_t.cc.

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

- artdaq/test/DAQrate/FragCounter\_t.cc

## 6.5 FragCounter\_test::ApplyWithOffset Struct Reference

Inheritance diagram for FragCounter\_test::ApplyWithOffset:



### Public Member Functions

- void **test\_method** ()

### 6.5.1 Detailed Description

Definition at line 37 of file FragCounter\_t.cc.

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

- artdaq/test/DAQrate/FragCounter\_t.cc

## 6.6 FragCounter\_test::ApplyWithOffset\_id Struct Reference

### 6.6.1 Detailed Description

Definition at line 37 of file FragCounter\_t.cc.

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

- artdaq/test/DAQrate/FragCounter\_t.cc

## 6.7 artdaq::art\_config\_file Class Reference

[art\\_config\\_file](#) wraps a temporary file used to configure art

```
#include <artdaq/DAQrate/SharedMemoryEventManager.hh>
```

## Public Member Functions

- [art\\_config\\_file](#) (fhicl::ParameterSet ps, uint32\_t shm\_key=0, uint32\_t broadcast\_key=0)  
*art\_config\_file Constructor*
- std::string [getFileName](#) () const  
*Get the path of the temporary file.*

### 6.7.1 Detailed Description

[art\\_config\\_file](#) wraps a temporary file used to configure art  
Definition at line 33 of file SharedMemoryEventManager.hh.

### 6.7.2 Constructor & Destructor Documentation

6.7.2.1 `artdaq::art_config_file::art_config_file ( fhicl::ParameterSet ps, uint32_t shm_key = 0, uint32_t broadcast_key = 0 )`  
[inline], [explicit]

[art\\_config\\_file](#) Constructor

Parameters

<i>ps</i>	ParameterSet to write to temporary file
<i>shm_key</i>	Shared Memory key to use (if 0, child program will use parent PID to generate)
<i>broadcast_key</i>	Shared Memory key to use for broadcasts (if 0, child program will use parent PID to generate)

Definition at line 42 of file SharedMemoryEventManager.hh.

### 6.7.3 Member Function Documentation

6.7.3.1 `std::string artdaq::art_config_file::getFileName ( ) const` [inline]

Get the path of the temporary file.

Returns

The path of the temporary file

Definition at line 86 of file SharedMemoryEventManager.hh.

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

- artdaq/artdaq/DAQrate/SharedMemoryEventManager.hh

## 6.8 artdaq::artdaqapp Class Reference

Class representing an artdaq application. Used by all "main" functions to start artdaq.

```
#include <proto/artdaqapp.hh>
```

## Classes

- struct [Config](#)  
*Configuration of artdaqapp. May be used for parameter validation*

## Public Types

- using [Parameters](#) = fhicl::WrappedTable< [Config](#) >  
*Used for ParameterSet validation (if desired)*

## Static Public Member Functions

- static void [runArtdaqApp](#) ([detail::TaskType](#) task, fhicl::ParameterSet const &config\_ps)  
*Run an artdaq Application*

### 6.8.1 Detailed Description

Class representing an artdaq application. Used by all "main" functions to start artdaq.

Definition at line 23 of file artdaqapp.hh.

### 6.8.2 Member Function Documentation

6.8.2.1 static void artdaq::artdaqapp::runArtdaqApp ( [detail::TaskType](#) task, fhicl::ParameterSet const & *config\_ps* )  
[inline], [static]

Run an artdaq Application

#### Parameters

<i>task</i>	Task type of the application
<i>config_ps</i>	"Startup" parameter set. See <a href="#">artdaq::artdaqapp::Config</a>

Definition at line 54 of file artdaqapp.hh.

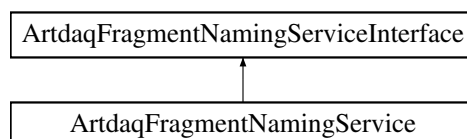
The documentation for this class was generated from the following file:

- artdaq/proto/artdaqapp.hh

## 6.9 ArtdaqFragmentNamingService Class Reference

[ArtdaqFragmentNamingService](#) extends [ArtdaqFragmentNamingServiceInterface](#). This implementation uses the default SystemTypeMap and directly assigns names based on it.

Inheritance diagram for ArtdaqFragmentNamingService:



## Public Member Functions

- virtual [~ArtdaqFragmentNamingService](#) ()  
*DefaultArtdaqFragmentNamingService Destructor.*
- [ArtdaqFragmentNamingService](#) (fhicl::ParameterSet const &pset, art::ActivityRegistry &)  
*NetMonTransportService Constructor.*

## Additional Inherited Members

### 6.9.1 Detailed Description

[ArtdaqFragmentNamingService](#) extends [ArtdaqFragmentNamingServiceInterface](#). This implementation uses the default SystemTypeMap and directly assigns names based on it.

Definition at line 13 of file [ArtdaqFragmentNamingService\\_service.cc](#).

### 6.9.2 Constructor & Destructor Documentation

#### 6.9.2.1 [ArtdaqFragmentNamingService::ArtdaqFragmentNamingService](#) ( fhicl::ParameterSet const & pset, art::ActivityRegistry & )

NetMonTransportService Constructor.

##### Parameters

<i>pset</i>	ParameterSet used to configure NetMonTransportService and DataSenderManager. See NetMonTransportService::Config
-------------	---

Definition at line 36 of file [ArtdaqFragmentNamingService\\_service.cc](#).

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

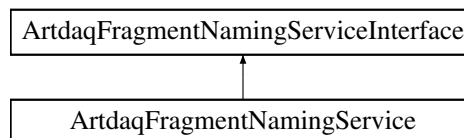
- [artdaq/artdaq/ArtModules/ArtdaqFragmentNamingService\\_service.cc](#)

## 6.10 ArtdaqFragmentNamingServiceInterface Class Reference

Interface for [ArtdaqFragmentNamingService](#). This interface is declared to art as part of the required registration of an art Service.

```
#include <artdaq/ArtModules/ArtdaqFragmentNamingService.h>
```

Inheritance diagram for [ArtdaqFragmentNamingServiceInterface](#):



## Public Member Functions

- virtual [~ArtdaqFragmentNamingServiceInterface](#) ()=default



*Default virtual destructor.*

- [ArtdaqFragmentNamingServiceInterface](#) (fhicl::ParameterSet const &ps)  
*ArtdaqFragmentNamingServiceInterface constructor.*
- std::string [GetInstanceNameForType](#) (artdaq::Fragment::type\_t type\_id) const  
*Returns the basic translation for the specified type. Must be implemented by derived classes.*
- std::set< std::string > [GetAllProductInstanceNames](#) () const  
*Returns the full set of product instance names which may be present in the data, based on the types that have been specified in the SetBasicTypes() and AddExtraType() methods. This does include "container" types, if the container type mapping is part of the basic types. Must be implemented by derived classes.*
- std::pair< bool, std::string > [GetInstanceNameForFragment](#) (artdaq::Fragment const &fragment) const  
*Returns the product instance name for the specified fragment, based on the types that have been specified in the SetBasicTypes() and AddExtraType() methods. This does include the use of "container" types, if the container type mapping is part of the basic types. If no mapping is found, the specified unidentified\_instance\_name is returned. Must be implemented by derived classes.*
- std::string [GetUnidentifiedInstanceName](#) () const  
*Get the name used for unidentified Fragment types.*

## Protected Attributes

- std::shared\_ptr  
< artdaq::FragmentNameHelper > [nameHelper\\_](#)  
*FragmentNameHelper plugin used to resolve Fragment names.*

## 6.10.1 Detailed Description

Interface for [ArtdaqFragmentNamingService](#). This interface is declared to art as part of the required registration of an art Service.

Definition at line 14 of file ArtdaqFragmentNamingService.h.

## 6.10.2 Constructor & Destructor Documentation

6.10.2.1 [ArtdaqFragmentNamingServiceInterface::ArtdaqFragmentNamingServiceInterface](#) ( fhicl::ParameterSet const & ps )  
[inline]

[ArtdaqFragmentNamingServiceInterface](#) constructor.

Parameters

<i>ps</i>	ParameterSet used to configure <a href="#">ArtdaqFragmentNamingServiceInterface</a>
-----------	---

[ArtdaqFragmentNamingServiceInterface](#) accepts the following Parameters: "unidentified\_instance\_name" (Default: "unidentified"): Name to use for any Fragments which are not successfully translated by the [ArtdaqFragmentNamingServiceInterface](#) "fragment\_type\_map" (Default: []): A list of Fragment type\_t to string pairs for additional types to register with the [ArtdaqFragmentNamingServiceInterface](#)

Definition at line 30 of file ArtdaqFragmentNamingService.h.

## 6.10.3 Member Function Documentation

6.10.3.1 std::string [ArtdaqFragmentNamingServiceInterface::GetUnidentifiedInstanceName](#) ( ) const [inline]

Get the name used for unidentified Fragment types.

### Returns

The name used for unidentified Fragment types

Definition at line 67 of file ArtdaqFragmentNamingService.h.

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

- artdaq/artdaq/ArtModules/ArtdaqFragmentNamingService.h

## 6.11 art::ArtdaqFragmentNamingServiceInterfaceConfig Struct Reference

Configuration for the [ArtdaqFragmentNamingServiceInterface](#)

```
#include <artdaq/ArtModules/detail/ArtConfig.hh>
```

### Public Attributes

- fhicl::Atom< std::string > [service\\_provider](#) {fhicl::Name{"service\_provider"}, fhicl::Comment{"Name of the provider for the [ArtdaqFragmentNamingServiceInterface](#) (e.g. [ArtdaqFragmentNamingService](#))"}}  
*"service\_provider" (REQUIRED): Name of the provider for the [ArtdaqFragmentNamingServiceInterface](#) (e.g. [ArtdaqFragmentNamingService](#) helper\_plugin: "ArtdaqDemo")*
- fhicl::Atom< std::string > [helper\\_plugin](#) {fhicl::Name{"helper\_plugin"}, fhicl::Comment{"Name of the helper plugin used by [ArtdaqFragmentNamingService](#)", "Artdaq"}}  
*"helper\_plugin" (Default: "Artdaq"): Name of the helper plugin used by [ArtdaqFragmentNamingService](#)*
- fhicl::Atom< std::string > [unidentified\\_instance\\_name](#) {fhicl::Name{"unidentified\_instance\_name"}, fhicl::Comment{"Name to use for Fragment types which are not identified by the [ArtdaqFragmentNamingServiceInterface](#) implementation."}, "unidentified"}  
*"unidentified\_instance\_name" (Default: "unidentified"): Name to use for Fragment types which are not identified by the [ArtdaqFragmentNamingServiceInterface](#) implementation.*
- fhicl::OptionalSequence  
 < fhicl::Tuple  
 < artdaq::Fragment::type\_t,  
 std::string > > [fragment\\_type\\_map](#) {fhicl::Name{"fragment\_type\_map"}, fhicl::Comment{"Additional types to register with the [ArtdaqFragmentNamingServiceInterface](#)"}}  
*"fragment\_type\_map" (OPTIONAL): Additional types to register with the [ArtdaqFragmentNamingServiceInterface](#)*

### 6.11.1 Detailed Description

Configuration for the [ArtdaqFragmentNamingServiceInterface](#)

Definition at line 17 of file ArtConfig.hh.

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

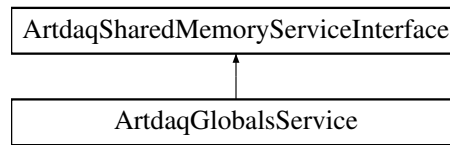
- artdaq/artdaq/ArtModules/detail/ArtConfig.hh

## 6.12 ArtdaqGlobalsService Class Reference

[ArtdaqGlobalsService](#) extends [ArtdaqSharedMemoryServiceInterface](#). It manages the artdaq Global variables my\_rank and app\_name, and initializes MetricManager. Users should retrieve a ServiceHandle to this class before using artdaq Globals to ensure the correct values are used.

```
#include <artdaq/ArtModules/ArtdaqGlobalsService.h>
```

Inheritance diagram for ArtdaqGlobalsService:



## Classes

- struct [Config](#)

*Allowed Configuration parameters of [ArtdaqGlobalsService](#). May be used for configuration validation*

## Public Types

- using [Parameters](#) = fhicl::WrappedTable< [Config](#) >

*Used for [ParameterSet](#) validation (if desired)*

## Public Member Functions

- virtual [~ArtdaqGlobalsService](#) ()  
*[ArtdaqGlobalsService](#) Destructor. Calls [disconnect\(\)](#).*
- [ArtdaqGlobalsService](#) (fhicl::ParameterSet const &pset, art::ActivityRegistry &)  
*[ArtdaqGlobalsService](#) Constructor.*
- std::unordered\_map  
  < artdaq::Fragment::type\_t,  
  std::unique\_ptr  
  < artdaq::Fragments > > [ReceiveEvent](#) (bool) override  
*Pretend to receive an event from the shared memory.*
- size\_t [GetQueueSize](#) () override  
*Get the number of events which are ready to be read (0)*
- size\_t [GetQueueCapacity](#) () override  
*Get the maximum number of events which can be stored in the shared memory (0)*
- std::shared\_ptr  
  < artdaq::detail::RawEventHeader > [GetEventHeader](#) () override  
*Get a shared\_ptr to the current event header, if any.*
- size\_t [GetMyId](#) () override  
*Get the ID of this art process. Always 0 since [ArtdaqGlobalsService](#) does not connect to shared memory.*

### 6.12.1 Detailed Description

[ArtdaqGlobalsService](#) extends [ArtdaqSharedMemoryServiceInterface](#). It manages the artdaq Global variables my\_rank and app\_name, and initializes MetricManager. Users should retrieve a ServiceHandle to this class before using artdaq Globals to ensure the correct values are used.

Definition at line 16 of file ArtdaqGlobalsService.h.

## 6.12.2 Constructor & Destructor Documentation

### 6.12.2.1 `ArtdaqGlobalsService::ArtdaqGlobalsService ( fhicl::ParameterSet const & pset, art::ActivityRegistry & )`

[ArtdaqGlobalsService](#) Constructor.

Parameters

<code>pset</code>	ParameterSet used to configure <a href="#">ArtdaqGlobalsService</a> . See <a href="#">ArtdaqGlobalsService::Config</a>
-------------------	--

Definition at line 16 of file `ArtdaqGlobalsService_service.cc`.

## 6.12.3 Member Function Documentation

### 6.12.3.1 `std::shared_ptr<artdaq::detail::RawEventHeader> ArtdaqGlobalsService::GetEventHeader ( ) [inline], [override], [virtual]`

Get a `shared_ptr` to the current event header, if any.

Returns

`Nullptr`

Implements [ArtdaqSharedMemoryServiceInterface](#).

Definition at line 62 of file `ArtdaqGlobalsService.h`.

### 6.12.3.2 `size_t ArtdaqGlobalsService::GetMyId ( ) [inline], [override], [virtual]`

Get the ID of this art process. Always 0 since [ArtdaqGlobalsService](#) does not connect to shared memory.

Returns

0

Implements [ArtdaqSharedMemoryServiceInterface](#).

Definition at line 68 of file `ArtdaqGlobalsService.h`.

### 6.12.3.3 `size_t ArtdaqGlobalsService::GetQueueCapacity ( ) [inline], [override], [virtual]`

Get the maximum number of events which can be stored in the shared memory (0)

Returns

The maximum number of events which can be stored in the shared memory (0)

Implements [ArtdaqSharedMemoryServiceInterface](#).

Definition at line 57 of file `ArtdaqGlobalsService.h`.

### 6.12.3.4 `size_t ArtdaqGlobalsService::GetQueueSize ( ) [inline], [override], [virtual]`

Get the number of events which are ready to be read (0)

**Returns**

The number of events which can be read (0)

Implements [ArtdaqSharedMemoryServiceInterface](#).

Definition at line 52 of file ArtdaqGlobalsService.h.

```
6.12.3.5 std::unordered_map<artdaq::Fragment::type_t, std::unique_ptr<artdaq::Fragments> >
        ArtdaqGlobalsService::ReceiveEvent( bool ) [inline],[override],[virtual]
```

Pretend to receive an event from the shared memory.

**Returns**

Map of Fragment types retrieved from shared memory

Implements [ArtdaqSharedMemoryServiceInterface](#).

Definition at line 43 of file ArtdaqGlobalsService.h.

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

- artdaq/artdaq/ArtModules/ArtdaqGlobalsService.h
- artdaq/artdaq/ArtModules/ArtdaqGlobalsService\_service.cc

## 6.13 art::ArtdaqInputHelper< U > Class Template Reference

This template class provides a unified interface for reading data into art.

```
#include <artdaq/ArtModules/ArtdaqInputHelper.hh>
```

**Public Member Functions**

- [ArtdaqInputHelper](#) (const [ArtdaqInputHelper](#) &)=delete  
*Copy Constructor is deleted.*
- [ArtdaqInputHelper](#) & [operator=](#) (const [ArtdaqInputHelper](#) &)=delete  
*Copy Assignment operator is deleted.*
- [~ArtdaqInputHelper](#) ()  
*ArtdaqInputHelper Destructor.*
- [ArtdaqInputHelper](#) (const fhicl::ParameterSet &ps, art::ProductRegistryHelper &helper, art::SourceHelper const &pm)  
*ArtdaqInputHelper Constructor.*
- void [closeCurrentFile](#) ()  
*Called by art to close the input source. No-Op.*
- void [readFile](#) (const std::string &, art::FileBlock \*&fb)  
*Emulate reading a file.*
- bool [hasMoreData](#) () const  
*Whether additional events are expected from the source.*
- bool [readNext](#) (art::RunPrincipal \*const inR, art::SubRunPrincipal \*const inSR, art::RunPrincipal \*&outR, art::SubRunPrincipal \*&outSR, art::EventPrincipal \*&outE)  
*Read the next event from the communication wrapper.*

### 6.13.1 Detailed Description

```
template<typename U> class art::ArtdaqInputHelper< U >
```

This template class provides a unified interface for reading data into art.

#### Template Parameters

<i>U</i>	The class responsible for delivering data
----------	---

JCF, May-27-2016 [ArtdaqInputHelper](#) is a template class which takes, as a parameter, a class which it uses to receive data; the instance of this class is called "communicationWrapper\_". As of this writing, this wrapper class is implemented by NetMonWrapper (for reading data into the aggregator from the eventbuilder) and TransferWrapper (for reading data into an art process). This class presents a unified approach to handling art provenance, regardless of the communication protocol used to read data in.

Definition at line 69 of file ArtdaqInputHelper.hh.

### 6.13.2 Constructor & Destructor Documentation

```
6.13.2.1 template<typename U > art::ArtdaqInputHelper< U >::ArtdaqInputHelper ( const fhicl::ParameterSet & ps,  
art::ProductRegistryHelper & helper, art::SourceHelper const & pm )
```

[ArtdaqInputHelper](#) Constructor.

#### Parameters

<i>ps</i>	ParameterSet used to configure communication wrapper class
<i>helper</i>	An art::ProductRegistryHelper for registering products
<i>pm</i>	An art::SourceHelper for handling provenance

Definition at line 182 of file ArtdaqInputHelper.hh.

### 6.13.3 Member Function Documentation

```
6.13.3.1 template<typename U > bool art::ArtdaqInputHelper< U >::hasMoreData ( ) const
```

Whether additional events are expected from the source.

#### Returns

True if [ArtdaqInputHelper](#) has not been shut down

Definition at line 441 of file ArtdaqInputHelper.hh.

```
6.13.3.2 template<typename U > ArtdaqInputHelper& art::ArtdaqInputHelper< U >::operator= ( const  
ArtdaqInputHelper< U > & ) [delete]
```

Copy Assignment operator is deleted.

#### Returns

[ArtdaqInputHelper](#) copy

6.13.3.3 `template<typename U> void art::ArtdaqInputHelper< U >::readFile ( const std::string & , art::FileBlock *& fb )`

Emulate reading a file.

## Parameters

<i>fb</i>	Output art::FileBlock object
-----------	------------------------------

Definition at line 431 of file ArtdaqInputHelper.hh.

```
6.13.3.4  template<typename U > bool art::ArtdaqInputHelper< U >::readNext ( art::RunPrincipal *const inR,
    art::SubRunPrincipal *const inSR, art::RunPrincipal *& outR, art::SubRunPrincipal *& outSR, art::EventPrincipal *& outE
    )
```

Read the next event from the communication wrapper.

## Parameters

<i>inR</i>	RunPrincipal input pointer
<i>inSR</i>	SubRunPrincipal input pointer
<i>outR</i>	RunPrincipal output pointer
<i>outSR</i>	SubRunPrincipal output pointer
<i>outE</i>	EventPrincipal output pointer

## Returns

Whether an event was successfully read from the communication wrapper

Definition at line 903 of file ArtdaqInputHelper.hh.

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

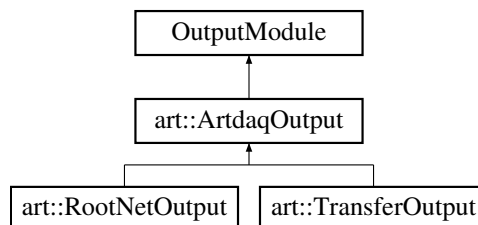
- artdaq/artdaq/ArtModules/ArtdaqInputHelper.hh

## 6.14 art::ArtdaqOutput Class Reference

This is the base class for artdaq OutputModules, providing the serialization interface for art Events.

```
#include <artdaq/ArtModules/ArtdaqOutput.hh>
```

Inheritance diagram for art::ArtdaqOutput:



### Public Member Functions

- [ArtdaqOutput](#) (fhicl::ParameterSet const &ps)  
*ArtdaqOutput Constructor*
- virtual [~ArtdaqOutput](#) ()=default  
*Destructor*



## Protected Member Functions

- virtual void [openFile](#) (FileBlock const &)  
*Perform actions necessary for opening files. No-op, but derived classes may override*
- virtual void [closeFile](#) ()  
*Perform actions necessary for closing files. No-op, but derived classes may override*
- virtual void [respondToCloseInputFile](#) (FileBlock const &)  
*Perform actions necessary after closing the input file. No-op, but derived classes may override*
- virtual void [respondToCloseOutputFiles](#) (FileBlock const &)  
*Perform actions necessary after closing the output file(s). No-op, but derived classes may override*
- virtual void [endJob](#) ()  
*Perform End-of-Job actions. No-op, but derived classes may override*
- void [beginRun](#) (RunPrincipal const &rp) final  
*Perform Begin Run actions. Derived classes should implement beginRun\_ instead.*
- virtual void [beginRun\\_](#) (RunPrincipal const &)  
*Perform Begin Run actions. No-op, but derived classes may override*
- void [beginSubRun](#) (SubRunPrincipal const &srp) final  
*Perform Begin SubRun actions. Derived classes should implement beginSubRun\_ instead.*
- virtual void [beginSubRun\\_](#) (SubRunPrincipal const &)  
*Perform Begin SubRun actions. No-op, but derived classes may override*
- void [event](#) (EventPrincipal const &ep) final  
*Perform actions for each event. Derived classes should implement event\_ instead.*
- virtual void [event\\_](#) (EventPrincipal const &)  
*Perform actions for each event. No-op, but derived classes may override*
- void [write](#) (EventPrincipal &ep) final  
*Write an EventPrincipal to TBufferFile and send*
- void [writeRun](#) (RunPrincipal &rp) final  
*Write a RunPrincipal to TBufferFile and send*
- void [writeSubRun](#) (SubRunPrincipal &srp) final  
*Write a SubRunPrincipal to TBufferFile and send*
- void [writeDataProducts](#) (std::unique\_ptr< TBufferFile > &msg, const Principal &principal, std::vector< Branch-Key \* > &bkv)  
*Extract the data products from a Principal and write them to the TBufferFile*
- void [extractProducts\\_](#) (Principal const &principal)  
*Extract the list of Products from the given Principal*
- void [send\\_init\\_message](#) ()  
*Send an init message downstream.*
- virtual void [SendMessage](#) (artdaq::FragmentPtr &msg)=0  
*Send the serialized art Event downstream. Artdaq output modules should define this function.*

### 6.14.1 Detailed Description

This is the base class for artdaq OutputModules, providing the serialization interface for art Events.

Definition at line 88 of file ArtdaqOutput.hh.

## 6.14.2 Constructor & Destructor Documentation

6.14.2.1 `art::ArtdaqOutput::ArtdaqOutput ( fhicl::ParameterSet const & ps )` `[inline],[explicit]`

[ArtdaqOutput](#) Constructor

## Parameters

<i>ps</i>	ParameterSet used to configure art::OutputModule
-----------	--

Definition at line 95 of file ArtdaqOutput.hh.

6.14.2.2 `virtual art::ArtdaqOutput::~~ArtdaqOutput ( ) [virtual],[default]`

Destructor

### 6.14.3 Member Function Documentation

6.14.3.1 `void art::ArtdaqOutput::beginRun ( RunPrincipal const & rp ) [inline],[final],[protected]`

Perform Begin Run actions. Derived classes should implement beginRun\_ instead.

## Parameters

<i>rp</i>	RunPrincipal of new run
-----------	-------------------------

Definition at line 150 of file ArtdaqOutput.hh.

6.14.3.2 `virtual void art::ArtdaqOutput::beginRun_ ( RunPrincipal const & ) [inline],[protected],[virtual]`

Perform Begin Run actions. No-op, but derived classes may override

Definition at line 158 of file ArtdaqOutput.hh.

6.14.3.3 `void art::ArtdaqOutput::beginSubRun ( SubRunPrincipal const & srp ) [inline],[final],[protected]`

Perform Begin SubRun actions. Derived classes should implement beginSubRun\_ instead.

## Parameters

<i>srp</i>	SubRunPrincipal of new subrun
------------	-------------------------------

Definition at line 164 of file ArtdaqOutput.hh.

6.14.3.4 `virtual void art::ArtdaqOutput::beginSubRun_ ( SubRunPrincipal const & ) [inline],[protected],[virtual]`

Perform Begin SubRun actions. No-op, but derived classes may override

Definition at line 172 of file ArtdaqOutput.hh.

6.14.3.5 `virtual void art::ArtdaqOutput::closeFile ( ) [inline],[protected],[virtual]`

Perform actions necessary for closing files. No-op, but derived classes may override

Definition at line 118 of file ArtdaqOutput.hh.

**6.14.3.6** `virtual void art::ArtdaqOutput::endJob ( ) [inline],[protected],[virtual]`

Perform End-of-Job actions. No-op, but derived classes may override

Definition at line 141 of file ArtdaqOutput.hh.

**6.14.3.7** `void art::ArtdaqOutput::event ( EventPrincipal const & ep ) [inline],[final],[protected]`

Perform actions for each event. Derived classes should implement event\_ instead.

Parameters

<i>ep</i>	EventPrincipal of event
-----------	-------------------------

Definition at line 178 of file ArtdaqOutput.hh.

**6.14.3.8** `virtual void art::ArtdaqOutput::event_ ( EventPrincipal const & ) [inline],[protected],[virtual]`

Perform actions for each event. No-op, but derived classes may override

Definition at line 186 of file ArtdaqOutput.hh.

**6.14.3.9** `void art::ArtdaqOutput::extractProducts_ ( Principal const & principal ) [inline],[protected]`

Extract the list of Products from the given Principal

Parameters

<i>principal</i>	Principal to extract products from
------------------	------------------------------------

Definition at line 790 of file ArtdaqOutput.hh.

**6.14.3.10** `virtual void art::ArtdaqOutput::openFile ( FileBlock const & ) [inline],[protected],[virtual]`

Perform actions necessary for opening files. No-op, but derived classes may override

Definition at line 110 of file ArtdaqOutput.hh.

**6.14.3.11** `virtual void art::ArtdaqOutput::respondToCloseInputFile ( FileBlock const & ) [inline],[protected],[virtual]`

Perform actions necessary after closing the input file. No-op, but derived classes may override

Definition at line 123 of file ArtdaqOutput.hh.

**6.14.3.12** `virtual void art::ArtdaqOutput::respondToCloseOutputFiles ( FileBlock const & ) [inline],[protected],[virtual]`

Perform actions necessary after closing the output file(s). No-op, but derived classes may override

Definition at line 132 of file ArtdaqOutput.hh.

6.14.3.13 `void art::ArtdaqOutput::send_init_message ( ) [inline],[protected]`

Send an init message downstream.

Definition at line 268 of file ArtdaqOutput.hh.

6.14.3.14 `virtual void art::ArtdaqOutput::SendMessage ( artdaq::FragmentPtr & msg ) [protected],[pure virtual]`

Send the serialized art Event downstream. Artdaq output modules should define this function.

Parameters

<i>msg</i>	Serialized art Event
------------	----------------------

Implemented in [art::RootNetOutput](#), and [art::TransferOutput](#).

6.14.3.15 `void art::ArtdaqOutput::write ( EventPrincipal & ep ) [inline],[final],[protected]`

Write an EventPrincipal to TBufferFile and send

Parameters

<i>ep</i>	EventPrincipal to write
-----------	-------------------------

Definition at line 526 of file ArtdaqOutput.hh.

6.14.3.16 `void art::ArtdaqOutput::writeDataProducts ( std::unique_ptr< TBufferFile > & msg, const Principal & principal, std::vector< BranchKey * > & bkv ) [inline],[protected]`

Extract the data products from a Principal and write them to the TBufferFile

Parameters

<i>msg</i>	Output TBufferFile
<i>principal</i>	Principal from which to extract products
<i>bkv</i>	Branch Keys for data products

Definition at line 411 of file ArtdaqOutput.hh.

6.14.3.17 `void art::ArtdaqOutput::writeRun ( RunPrincipal & rp ) [inline],[final],[protected]`

Write a RunPrincipal to TBufferFile and send

Parameters

<i>rp</i>	RunPrincipal to write
-----------	-----------------------

Definition at line 633 of file ArtdaqOutput.hh.

6.14.3.18 `void art::ArtdaqOutput::writeSubRun ( SubRunPrincipal & srp ) [inline],[final],[protected]`

Write a SubRunPrincipal to TBufferFile and send

## Parameters

<i>srp</i>	SubRunPrincipal to write
------------	--------------------------

Definition at line 689 of file ArtdaqOutput.hh.

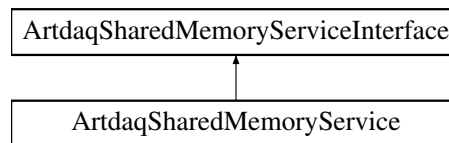
The documentation for this class was generated from the following file:

- artdaq/artdaq/ArtModules/ArtdaqOutput.hh

## 6.15 ArtdaqSharedMemoryService Class Reference

[ArtdaqSharedMemoryService](#) extends [ArtdaqSharedMemoryServiceInterface](#). It receives events from shared memory using [SharedMemoryEventReceiver](#). It also manages the artdaq Global variables `my_rank` and `app_name`. Users should retrieve a `ServiceHandle` to this class before using artdaq Globals to ensure the correct values are used.

Inheritance diagram for [ArtdaqSharedMemoryService](#):



### Classes

- struct [Config](#)

*Allowed Configuration parameters of [NetMonTransportService](#). May be used for configuration validation*

### Public Types

- using [Parameters](#) = `fhicl::WrappedTable< Config >`

*Used for [ParameterSet](#) validation (if desired)*

### Public Member Functions

- virtual `~ArtdaqSharedMemoryService ()`  
*[NetMonTransportService](#) Destructor. Calls `disconnect()`.*
- `ArtdaqSharedMemoryService (fhicl::ParameterSet const &pset, art::ActivityRegistry &)`  
*[NetMonTransportService](#) Constructor.*
- `std::unordered_map< artdaq::Fragment::type_t, std::unique_ptr< artdaq::Fragments > >` `ReceiveEvent (bool broadcast)` override  
*Receive an event from the shared memory.*
- `size_t GetQueueSize ()` override  
*Get the number of events which are ready to be read.*
- `size_t GetQueueCapacity ()` override  
*Get the maximum number of events which can be stored in the shared memory.*

- `std::shared_ptr`  
`< artdaq::detail::RawEventHeader > GetEventHeader ()` override  
*Get a shared\_ptr to the current event header, if any.*
- `size_t GetMyId ()` override  
*Get the ID of this art process.*

### 6.15.1 Detailed Description

[ArtdaqSharedMemoryService](#) extends [ArtdaqSharedMemoryServiceInterface](#). It receives events from shared memory using `SharedMemoryEventReceiver`. It also manages the artdaq Global variables `my_rank` and `app_name`. Users should retrieve a `ServiceHandle` to this class before using artdaq Globals to ensure the correct values are used.

Definition at line 27 of file `ArtdaqSharedMemoryService_service.cc`.

### 6.15.2 Constructor & Destructor Documentation

6.15.2.1 `ArtdaqSharedMemoryService::ArtdaqSharedMemoryService ( fhicl::ParameterSet const & pset, art::ActivityRegistry & )`

NetMonTransportService Constructor.

Parameters

<i>pset</i>	ParameterSet used to configure NetMonTransportService and DataSenderManager. See <code>NetMonTransportService::Config</code>
-------------	--

Definition at line 105 of file `ArtdaqSharedMemoryService_service.cc`.

### 6.15.3 Member Function Documentation

6.15.3.1 `std::shared_ptr<artdaq::detail::RawEventHeader> ArtdaqSharedMemoryService::GetEventHeader ( )` `[inline]`, `[override]`, `[virtual]`

Get a shared\_ptr to the current event header, if any.

Returns

`std::shared_ptr` to current event header. May be nullptr if no event is currently being read

Implements [ArtdaqSharedMemoryServiceInterface](#).

Definition at line 77 of file `ArtdaqSharedMemoryService_service.cc`.

6.15.3.2 `size_t ArtdaqSharedMemoryService::GetMyId ( )` `[inline]`, `[override]`, `[virtual]`

Get the ID of this art process.

Returns

The ID of this art process from the shared memory segment

Implements [ArtdaqSharedMemoryServiceInterface](#).

Definition at line 83 of file `ArtdaqSharedMemoryService_service.cc`.

**6.15.3.3** `size_t ArtdaqSharedMemoryService::GetQueueCapacity ( ) [inline],[override],[virtual]`

Get the maximum number of events which can be stored in the shared memory.

#### Returns

The maximum number of events which can be stored in the shared memory

Implements [ArtdaqSharedMemoryServiceInterface](#).

Definition at line 72 of file `ArtdaqSharedMemoryService_service.cc`.

**6.15.3.4** `size_t ArtdaqSharedMemoryService::GetQueueSize ( ) [inline],[override],[virtual]`

Get the number of events which are ready to be read.

#### Returns

The number of events which can be read

Implements [ArtdaqSharedMemoryServiceInterface](#).

Definition at line 67 of file `ArtdaqSharedMemoryService_service.cc`.

**6.15.3.5** `std::unordered_map< artdaq::Fragment::type_t, std::unique_ptr< artdaq::Fragments > > ArtdaqSharedMemoryService::ReceiveEvent ( bool broadcast ) [override],[virtual]`

Receive an event from the shared memory.

#### Parameters

<i>broadcast</i>	Whether to only attempt to receive a broadcast (broadcasts are always preferentially received over data)
------------------	--

#### Returns

Map of Fragment types retrieved from shared memory

Implements [ArtdaqSharedMemoryServiceInterface](#).

Definition at line 161 of file `ArtdaqSharedMemoryService_service.cc`.

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

- `artdaq/artdaq/ArtModules/ArtdaqSharedMemoryService_service.cc`

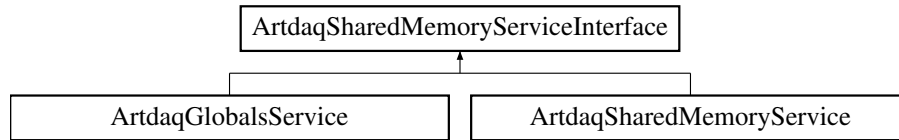
## 6.16 ArtdaqSharedMemoryServiceInterface Class Reference

Interface for [ArtdaqSharedMemoryService](#). This interface is declared to art as part of the required registration of an art Service.

```
#include <artdaq/ArtModules/ArtdaqSharedMemoryServiceInterface.h>
```

Inheritance diagram for `ArtdaqSharedMemoryServiceInterface`:





## Public Member Functions

- virtual `~ArtdaqSharedMemoryServiceInterface()`=default  
*Default virtual destructor.*
- virtual `std::unordered_map< artdaq::Fragment::type_t, std::unique_ptr< artdaq::Fragments > > ReceiveEvent` (bool broadcast)=0  
*Receive an event from the shared memory.*
- virtual `size_t GetQueueSize()`=0  
*Get the number of events which are ready to be read.*
- virtual `size_t GetQueueCapacity()`=0  
*Get the maximum number of events which can be stored in the shared memory.*
- virtual `std::shared_ptr< artdaq::detail::RawEventHeader > GetEventHeader()`=0  
*Get a shared\_ptr to the current event header, if any.*
- virtual `size_t GetMyId()`=0  
*Get the ID of this art process.*

### 6.16.1 Detailed Description

Interface for [ArtdaqSharedMemoryService](#). This interface is declared to art as part of the required registration of an art Service.

Definition at line 11 of file `ArtdaqSharedMemoryServiceInterface.h`.

### 6.16.2 Member Function Documentation

**6.16.2.1** `virtual std::shared_ptr<artdaq::detail::RawEventHeader> ArtdaqSharedMemoryServiceInterface::GetEventHeader ( )`  
[pure virtual]

Get a shared\_ptr to the current event header, if any.

#### Returns

`std::shared_ptr` to current event header. May be nullptr if no event is currently being read

Implemented in [ArtdaqSharedMemoryService](#), and [ArtdaqGlobalsService](#).

**6.16.2.2** `virtual size_t ArtdaqSharedMemoryServiceInterface::GetMyId ( )` [pure virtual]

Get the ID of this art process.

**Returns**

The ID of this art process (from shm->[GetMyId\(\)](#))

Implemented in [ArtdaqSharedMemoryService](#), and [ArtdaqGlobalsService](#).

6.16.2.3 `virtual size_t ArtdaqSharedMemoryServiceInterface::GetQueueCapacity ( ) [pure virtual]`

Get the maximum number of events which can be stored in the shared memory.

**Returns**

The maximum number of events which can be stored in the shared memory

Implemented in [ArtdaqSharedMemoryService](#), and [ArtdaqGlobalsService](#).

6.16.2.4 `virtual size_t ArtdaqSharedMemoryServiceInterface::GetQueueSize ( ) [pure virtual]`

Get the number of events which are ready to be read.

**Returns**

The number of events which can be read

Implemented in [ArtdaqSharedMemoryService](#), and [ArtdaqGlobalsService](#).

6.16.2.5 `virtual std::unordered_map<artdaq::Fragment::type_t, std::unique_ptr<artdaq::Fragments> > ArtdaqSharedMemoryServiceInterface::ReceiveEvent ( bool broadcast ) [pure virtual]`

Receive an event from the shared memory.

**Parameters**

<i>broadcast</i>	Whether to only attempt to receive a broadcast (broadcasts are always preferentially received over data)
------------------	--

**Returns**

Map of Fragment types retrieved from shared memory

Implemented in [ArtdaqSharedMemoryService](#), and [ArtdaqGlobalsService](#).

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

- `artdaq/artdaq/ArtModules/ArtdaqSharedMemoryServiceInterface.h`

## 6.17 `art::ArtdaqSharedMemoryServiceInterfaceConfig` Struct Reference

Configuration for the [ArtdaqSharedMemoryServiceInterface](#)

```
#include <artdaq/ArtModules/detail/ArtConfig.hh>
```

## Public Attributes

- fhicl::Atom< std::string > [service\\_provider](#) {fhicl::Name{"service\_provider"}, fhicl::Comment{"Name of the provider for the [ArtdaqSharedMemoryServiceInterface](#) (e.g. [ArtdaqSharedMemoryService](#))"}}  
*"service\_provider" (REQUIRED): Name of the provider for the [ArtdaqSharedMemoryServiceInterface](#) (e.g. [ArtdaqSharedMemoryService](#))*
- fhicl::Atom< size\_t > [read\\_timeout\\_us](#) {fhicl::Name{"read\_timeout\_us"}, fhicl::Comment{"Amount of time (in us) to wait for events from shared memory. Defaults to [waiting\\_time](#) if unspecified."}, 600000000}  
*"read\_timeout\_us" (Default: "waiting\_time" \* 1000000): Amount of time (in us) to wait for events from shared memory. Defaults to waiting\_time if unspecified.*
- fhicl::Atom< double > [waiting\\_time](#) {fhicl::Name{"waiting\_time"}, fhicl::Comment{"Amount of time (in s) to wait for events from shared memory. Overridden by [read\\_timeout\\_us](#) if specified."}, 600.0}  
*"waiting\_time" (Default: 600.0): Amount of time (in s) to wait for events from shared memory. Overridden by read\_timeout\_us if specified.*
- fhicl::Atom< bool > [resume\\_after\\_timeout](#) {fhicl::Name{"resume\_after\_timeout"}, fhicl::Comment{"Whether to continue to attempt to receive events after a timeout occurs"}, true}  
*"resume\_after\_timeout" (Default: true): Whether to continue to attempt to receive events after a timeout occurs*
- fhicl::OptionalAtom< int > [shared\\_memory\\_key](#) {fhicl::Name{"shared\_memory\_key"}, fhicl::Comment{"Key to use for Data shared memory segment. Automatically generated using parent PID."}}  
*"shared\_memory\_key" (OPTIONAL): Key to use for Data shared memory segment. Automatically generated using parent PID.*
- fhicl::OptionalAtom< int > [broadcast\\_shared\\_memory\\_key](#) {fhicl::Name{"broadcast\_shared\_memory\_key"}, fhicl::Comment{"Key to use for Broadcast shared memory segment. Automatically generated using parent PID."}}  
*"broadcast\_shared\_memory\_key" (OPTIONAL): Key to use for Broadcast shared memory segment. Automatically generated using parent PID.*
- fhicl::OptionalTable  
 < artdaq::MetricManager::Config > [metrics](#) {fhicl::Name{"metrics"}, fhicl::Comment{"Configuration for artdaq Metrics"}}  
*"metrics" (OPTIONAL): Configuration for artdaq Metrics*

### 6.17.1 Detailed Description

Configuration for the [ArtdaqSharedMemoryServiceInterface](#)

Definition at line 32 of file ArtConfig.hh.

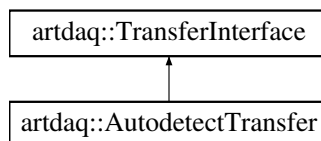
The documentation for this struct was generated from the following file:

- artdaq/artdaq/ArtModules/detail/ArtConfig.hh

## 6.18 artdaq::AutodetectTransfer Class Reference

The [AutodetectTransfer TransferInterface](#) plugin sets up a Shmem\_transfer plugin or TCPSocket\_transfer plugin depending if the source and destination are on the same host, to maximize throughput.

Inheritance diagram for artdaq::AutodetectTransfer:



## Public Member Functions

- [AutodetectTransfer](#) (const fhicl::ParameterSet &pset, [Role](#) role)  
*AutodetectTransfer Constructor.*
- [~AutodetectTransfer](#) () override=default  
*AutodetectTransfer default Destructor.*
- int [receiveFragment](#) (artdaq::Fragment &fragment, size\_t receiveTimeout) override  
*Receive a Fragment, using the underlying transfer plugin.*
- int [receiveFragmentHeader](#) (detail::RawFragmentHeader &header, size\_t receiveTimeout) override  
*Receive a Fragment Header from the transport mechanism.*
- int [receiveFragmentData](#) (RawDataType \*destination, size\_t wordCount) override  
*Receive the body of a Fragment to the given destination pointer.*
- [CopyStatus transfer\\_fragment\\_min\\_blocking\\_mode](#) (artdaq::Fragment const &fragment, size\_t send\_timeout\_ - usec) override  
*Send a Fragment in non-reliable mode, using the underlying transfer plugin.*
- [CopyStatus transfer\\_fragment\\_reliable\\_mode](#) (artdaq::Fragment &&fragment) override  
*Send a Fragment in reliable mode, using the underlying transfer plugin.*
- bool [isRunning](#) () override  
*Determine whether the [TransferInterface](#) plugin is able to send/receive data.*
- void [flush\\_buffers](#) () override  
*Flush any in-flight data. This should be used by the receiver after the receive loop has ended.*

## Additional Inherited Members

### 6.18.1 Detailed Description

The [AutodetectTransfer TransferInterface](#) plugin sets up a Shmem\_transfer plugin or TCPSocket\_transfer plugin depending if the source and destination are on the same host, to maximize throughput.

Definition at line 17 of file Autodetect\_transfer.cc.

### 6.18.2 Constructor & Destructor Documentation

#### 6.18.2.1 artdaq::AutodetectTransfer::AutodetectTransfer ( const fhicl::ParameterSet & pset, [Role](#) role )

[AutodetectTransfer](#) Constructor.

Parameters

<i>pset</i>	ParameterSet used to configure <a href="#">AutodetectTransfer</a>
<i>role</i>	Role of this <a href="#">TransferInterface</a> , either kReceive or kSend

Definition at line 110 of file Autodetect\_transfer.cc.

### 6.18.3 Member Function Documentation

#### 6.18.3.1 bool artdaq::AutodetectTransfer::isRunning ( ) [inline],[override],[virtual]

Determine whether the [TransferInterface](#) plugin is able to send/receive data.

**Returns**

True if the [TransferInterface](#) plugin is currently able to send/receive data

Reimplemented from [artdaq::TransferInterface](#).

Definition at line 91 of file Autodetect\_transfer.cc.

**6.18.3.2** `int artdaq::AutodetectTransfer::receiveFragment ( artdaq::Fragment & fragment, size_t receiveTimeout ) [inline], [override], [virtual]`

Receive a Fragment, using the underlying transfer plugin.

**Parameters**

<i>fragment</i>	Output Fragment
<i>receiveTimeout</i>	Time to wait before returning <a href="#">TransferInterface::RECV_TIMEOUT</a>

**Returns**

Rank of sender

Reimplemented from [artdaq::TransferInterface](#).

Definition at line 38 of file Autodetect\_transfer.cc.

**6.18.3.3** `int artdaq::AutodetectTransfer::receiveFragmentData ( RawDataType * destination, size_t wordCount ) [inline], [override], [virtual]`

Receive the body of a Fragment to the given destination pointer.

**Parameters**

<i>destination</i>	Pointer to memory region where Fragment data should be stored
<i>wordCount</i>	Number of words of Fragment data to receive

**Returns**

The rank the Fragment was received from (should be `source_rank`), or `RECV_TIMEOUT`

Implements [artdaq::TransferInterface](#).

Definition at line 61 of file Autodetect\_transfer.cc.

**6.18.3.4** `int artdaq::AutodetectTransfer::receiveFragmentHeader ( detail::RawFragmentHeader & header, size_t receiveTimeout ) [inline], [override], [virtual]`

Receive a Fragment Header from the transport mechanism.

**Parameters**

<i>out</i>	<i>header</i>	Received Fragment Header
------------	---------------	--------------------------

	<i>receiveTimeout</i>	<a href="#">Timeout</a> for receive
--	-----------------------	-------------------------------------

**Returns**

The rank the Fragment was received from (should be `source_rank`), or `RECV_TIMEOUT`

Implements [artdaq::TransferInterface](#).

Definition at line 50 of file Autodetect\_transfer.cc.

**6.18.3.5** **CopyStatus** `artdaq::AutodetectTransfer::transfer_fragment_min_blocking_mode ( artdaq::Fragment const & fragment, size_t send_timeout_usec )` `[inline]`, `[override]`, `[virtual]`

Send a Fragment in non-reliable mode, using the underlying transfer plugin.

**Parameters**

<i>fragment</i>	The Fragment to send
<i>send_timeout_usec</i>	How long to wait before aborting. Defaults to <code>size_t::MAX_VALUE</code>

**Returns**

A [TransferInterface::CopyStatus](#) result variable

Implements [artdaq::TransferInterface](#).

Definition at line 72 of file Autodetect\_transfer.cc.

**6.18.3.6** **CopyStatus** `artdaq::AutodetectTransfer::transfer_fragment_reliable_mode ( artdaq::Fragment && fragment )` `[inline]`, `[override]`, `[virtual]`

Send a Fragment in reliable mode, using the underlying transfer plugin.

**Parameters**

<i>fragment</i>	The Fragment to send
-----------------	----------------------

**Returns**

A [TransferInterface::CopyStatus](#) result variable

Implements [artdaq::TransferInterface](#).

Definition at line 82 of file Autodetect\_transfer.cc.

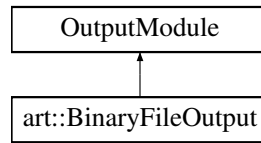
The documentation for this class was generated from the following file:

- `artdaq/artdaq/TransferPlugins/Autodetect_transfer.cc`

## 6.19 art::BinaryFileOutput Class Reference

The [BinaryFileOutput](#) module streams art Events to a binary file, bypassing ROOT.

Inheritance diagram for art::BinaryFileOutput:



## Public Member Functions

- [BinaryFileOutput](#) (ParameterSet const &ps)  
*BinaryFileOutput Constructor.*
- [~BinaryFileOutput](#) () override  
*BinaryFileOutput Destructor.*

### 6.19.1 Detailed Description

The [BinaryFileOutput](#) module streams art Events to a binary file, bypassing ROOT.

Definition at line 43 of file BinaryFileOutput\_module.cc.

### 6.19.2 Constructor & Destructor Documentation

6.19.2.1 `art::BinaryFileOutput::BinaryFileOutput ( ParameterSet const & ps ) [explicit]`

[BinaryFileOutput](#) Constructor.

Parameters

<i>ps</i>	ParameterSet used to configure <a href="#">BinaryFileOutput</a>
-----------	---

[BinaryFileOutput](#) accepts the same configuration parameters as art::OutputModule. It has the same name substitution code that RootOutput uses to unify names.

[BinaryFileOutput](#) also expects the following Parameters: "fileName" (REQUIRED): Name of the file to write "directIO" (Default: false): Whether to use O\_DIRECT (Ref. issue #24437)

Definition at line 94 of file BinaryFileOutput\_module.cc.

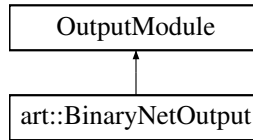
The documentation for this class was generated from the following file:

- artdaq/artdaq/ArtModules/BinaryFileOutput\_module.cc

## 6.20 art::BinaryNetOutput Class Reference

An art::OutputModule which sends Fragments using DataSenderManager. This module produces output identical to that of a BoardReader, for use in systems which have multiple layers of EventBuilders.

Inheritance diagram for art::BinaryNetOutput:



## Public Member Functions

- [BinaryNetOutput](#) (ParameterSet const &ps)  
*BinaryNetOutput Constructor.*
- [~BinaryNetOutput](#) () override  
*BinaryNetOutput Destructor.*

### 6.20.1 Detailed Description

An `art::OutputModule` which sends Fragments using `DataSenderManager`. This module produces output identical to that of a `BoardReader`, for use in systems which have multiple layers of `EventBuilders`.

Definition at line 42 of file `BinaryNetOutput_module.cc`.

### 6.20.2 Constructor & Destructor Documentation

6.20.2.1 `art::BinaryNetOutput::BinaryNetOutput ( ParameterSet const & ps ) [explicit]`

[BinaryNetOutput](#) Constructor.

Parameters

<i>ps</i>	ParameterSet used to configure <a href="#">BinaryNetOutput</a>
-----------	--

[BinaryNetOutput](#) forwards its `ParameterSet` to `art::OutputModule`, so any Parameters it requires are also required by [BinaryNetOutput](#). [BinaryNetOutput](#) also forwards its `ParameterSet` to `DataSenderManager`, so any Parameters *it* requires are *also* required by `BinaryMPIOutput`. Finally, [BinaryNetOutput](#) accpets the following parameters: "rt\_priority" (Default: 0): Priority for this thread "module\_name" (Default: [BinaryNetOutput](#)): Friendly name for this module (MessageFacility Category)

Definition at line 92 of file `BinaryNetOutput_module.cc`.

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

- `artdaq/artdaq/ArtModules/BinaryNetOutput_module.cc`

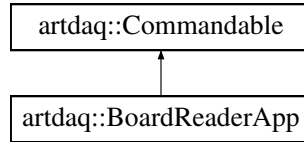
## 6.21 artdaq::BoardReaderApp Class Reference

[BoardReaderApp](#) is an `artdaq::Commandable` derived class which controls the [BoardReaderCore](#) state machine.

```
#include <artdaq/Application/BoardReaderApp.hh>
```

Inheritance diagram for `artdaq::BoardReaderApp`:





## Public Member Functions

- [BoardReaderApp](#) ()  
*BoardReaderApp Constructor.*
- [BoardReaderApp](#) ([BoardReaderApp](#) const &)=delete  
*Copy Constructor is deleted.*
- virtual [~BoardReaderApp](#) ()=default  
*Default Destructor.*
- [BoardReaderApp](#) & [operator=](#) ([BoardReaderApp](#) const &)=delete  
*Copy Assignment Operator is deleted.*
- [BoardReaderApp](#) ([BoardReaderApp](#) &&)=delete  
*Move Constructor is deleted.*
- [BoardReaderApp](#) & [operator=](#) ([BoardReaderApp](#) &&)=delete  
*Move Assignment Operator is deleted.*
- bool [do\\_initialize](#) (fhicl::ParameterSet const &pset, uint64\_t timeout, uint64\_t timestamp) override  
*Initialize the BoardReaderCore.*
- bool [do\\_start](#) (art::RunID id, uint64\_t timeout, uint64\_t timestamp) override  
*Start the BoardReaderCore.*
- bool [do\\_stop](#) (uint64\_t timeout, uint64\_t timestamp) override  
*Stop the BoardReaderCore.*
- bool [do\\_pause](#) (uint64\_t timeout, uint64\_t timestamp) override  
*Pause the BoardReaderCore.*
- bool [do\\_resume](#) (uint64\_t timeout, uint64\_t timestamp) override  
*Resume the BoardReaderCore.*
- bool [do\\_shutdown](#) (uint64\_t timeout) override  
*Shutdown the BoardReaderCore.*
- bool [do\\_soft\\_initialize](#) (fhicl::ParameterSet const &pset, uint64\_t timeout, uint64\_t timestamp) override  
*Soft-Initialize the BoardReaderCore.*
- bool [do\\_reinitialize](#) (fhicl::ParameterSet const &pset, uint64\_t timeout, uint64\_t timestamp) override  
*Reinitialize the BoardReaderCore.*
- void [BootedEnter](#) () override  
*Action taken upon entering the "Booted" state.*
- bool [do\\_meta\\_command](#) (std::string const &command, std::string const &arg) override  
*Perform a user-defined command (passed to CommandableFragmentGenerator)*
- std::string [report](#) (std::string const &which) const override  
*If which is "transition\_status", report the status of the last transition. Otherwise pass through to BoardReaderCore.*
- [CommandableFragmentGenerator](#)  
const \* [GetGeneratorPointer](#) ()

## Additional Inherited Members

### 6.21.1 Detailed Description

[BoardReaderApp](#) is an [artdaq::Commandable](#) derived class which controls the [BoardReaderCore](#) state machine.

Definition at line 14 of file BoardReaderApp.hh.

### 6.21.2 Member Function Documentation

**6.21.2.1** `void artdaq::BoardReaderApp::BootedEnter( ) [override],[virtual]`

Action taken upon entering the "Booted" state.

This resets the [BoardReaderCore](#) pointer

Reimplemented from [artdaq::Commandable](#).

Definition at line 271 of file BoardReaderApp.cc.

**6.21.2.2** `bool artdaq::BoardReaderApp::do_initialize ( fhicl::ParameterSet const & pset, uint64_t timeout, uint64_t timestamp ) [override],[virtual]`

Initialize the [BoardReaderCore](#).

Parameters

<i>pset</i>	ParameterSet used to configure the <a href="#">BoardReaderCore</a>
<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 21 of file BoardReaderApp.cc.

**6.21.2.3** `bool artdaq::BoardReaderApp::do_meta_command ( std::string const & command, std::string const & arg ) [override],[virtual]`

Perform a user-defined command (passed to [CommandableFragmentGenerator](#))

Parameters

<i>command</i>	Name of the command
<i>arg</i>	Argument for the command

Returns

Whether the command succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 282 of file BoardReaderApp.cc.

6.21.2.4 `bool artdaq::BoardReaderApp::do_pause ( uint64_t timeout, uint64_t timestamp )` `[override]`, `[virtual]`

Pause the [BoardReaderCore](#).

## Parameters

<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

## Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 150 of file BoardReaderApp.cc.

```
6.21.2.5  bool artdaq::BoardReaderApp::do_reinitialize ( fhicl::ParameterSet const & pset, uint64_t timeout, uint64_t timestamp )  
          [override],[virtual]
```

Reinitialize the [BoardReaderCore](#).

## Parameters

<i>pset</i>	ParameterSet used to configure the <a href="#">BoardReaderCore</a>
<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

## Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 259 of file BoardReaderApp.cc.

```
6.21.2.6  bool artdaq::BoardReaderApp::do_resume ( uint64_t timeout, uint64_t timestamp ) [override],[virtual]
```

Resume the [BoardReaderCore](#).

## Parameters

<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

## Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 170 of file BoardReaderApp.cc.

```
6.21.2.7  bool artdaq::BoardReaderApp::do_shutdown ( uint64_t timeout ) [override],[virtual]
```

Shutdown the [BoardReaderCore](#).

## Parameters

<i>timeout</i>	<a href="#">Timeout</a> for transition
----------------	--

## Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 230 of file BoardReaderApp.cc.

**6.21.2.8** `bool artdaq::BoardReaderApp::do_soft_initialize ( fhicl::ParameterSet const & pset, uint64_t timeout, uint64_t timestamp ) [override], [virtual]`

Soft-Initialize the [BoardReaderCore](#).

## Parameters

<i>pset</i>	ParameterSet used to configure the <a href="#">BoardReaderCore</a>
<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

## Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 246 of file BoardReaderApp.cc.

**6.21.2.9** `bool artdaq::BoardReaderApp::do_start ( art::RunID id, uint64_t timeout, uint64_t timestamp ) [override], [virtual]`

Start the [BoardReaderCore](#).

## Parameters

<i>id</i>	Run ID of new run
<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

## Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 47 of file BoardReaderApp.cc.

**6.21.2.10** `bool artdaq::BoardReaderApp::do_stop ( uint64_t timeout, uint64_t timestamp ) [override], [virtual]`

Stop the [BoardReaderCore](#).

## Parameters

<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

## Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 116 of file BoardReaderApp.cc.

6.21.2.11 **BoardReaderApp& artdaq::BoardReaderApp::operator= ( BoardReaderApp const & )** `[delete]`

Copy Assignment Operator is deleted.

## Returns

[BoardReaderApp](#) copy

6.21.2.12 **std::string artdaq::BoardReaderApp::report ( std::string const & *which* ) const** `[override],[virtual]`

If *which* is "transition\_status", report the status of the last transition. Otherwise pass through to [BoardReaderCore](#).

## Parameters

<i>which</i>	What to report on
--------------	-------------------

## Returns

Report string. Empty for unknown "which" parameter

Reimplemented from [artdaq::Commandable](#).

Definition at line 294 of file BoardReaderApp.cc.

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

- artdaq/artdaq/Application/BoardReaderApp.hh
- artdaq/artdaq/Application/BoardReaderApp.cc

## 6.22 artdaq::BoardReaderCore Class Reference

[BoardReaderCore](#) implements the state machine for the BoardReader artdaq application. It contains a [Commandable-FragmentGenerator](#), which generates Fragments which are then sent to a [DataSenderManager](#) by [BoardReaderCore](#).

```
#include <artdaq/Application/BoardReaderCore.hh>
```

### Public Member Functions

- [BoardReaderCore](#) ([Commandable](#) &parent\_application)

- [BoardReaderCore](#) Constructor.  
*Copy Constructor is Deleted.*
- virtual [~BoardReaderCore](#) ()  
[BoardReaderCore](#) Destructor.
- [BoardReaderCore](#) & [operator=](#) ([BoardReaderCore](#) const &)=delete  
*Copy Assignment Operator is deleted.*
- [BoardReaderCore](#) ([BoardReaderCore](#) &&)=delete  
*Move Constructor is deleted.*
- [BoardReaderCore](#) & [operator=](#) ([BoardReaderCore](#) &&)=delete  
*Move Assignment Operator is deleted.*
- bool [initialize](#) (fhicl::ParameterSet const &pset, uint64\_t timeout, uint64\_t timestamp)  
*Initialize the [BoardReaderCore](#).*
- bool [start](#) (art::RunID id, uint64\_t timeout, uint64\_t timestamp)  
*Start the BoardReader, and the [CommandableFragmentGenerator](#).*
- bool [stop](#) (uint64\_t timeout, uint64\_t timestamp)  
*Stop the BoardReader, and the [CommandableFragmentGenerator](#).*
- bool [pause](#) (uint64\_t timeout, uint64\_t timestamp)  
*Pause the BoardReader, and the [CommandableFragmentGenerator](#).*
- bool [resume](#) (uint64\_t timeout, uint64\_t timestamp)  
*Resume the BoardReader, and the [CommandableFragmentGenerator](#).*
- bool [shutdown](#) (uint64\_t timeout)  
*Shutdown the BoardReader, and the [CommandableFragmentGenerator](#).*
- bool [soft\\_initialize](#) (fhicl::ParameterSet const &pset, uint64\_t timeout, uint64\_t timestamp)  
*Soft-Initialize the BoardReader. No-Op.*
- bool [reinitialize](#) (fhicl::ParameterSet const &pset, uint64\_t timeout, uint64\_t timestamp)  
*Reinitialize the BoardReader. No-Op.*
- void [receive\\_fragments](#) ()  
*Main working loop of the [BoardReaderCore](#).*
- void [send\\_fragments](#) ()  
*Main working loop of the [BoardReaderCore](#), pt. 2.*
- std::string [report](#) (std::string const &which) const  
*Send a report on a given run-time quantity.*
- bool [metaCommand](#) (std::string const &command, std::string const &arg)  
*Run a user-defined command on the [CommandableFragmentGenerator](#).*
- size\_t [GetFragmentsProcessed](#) ()  
*Get the number of Fragments processed this run*
- [CommandableFragmentGenerator](#)  
const \* [GetGeneratorPointer](#) ()
- bool [GetSenderThreadActive](#) ()  
*Get whether the sender thread is still running.*
- bool [GetReceiverThreadActive](#) ()  
*Get whether the receiver thread is still running.*
- void [SetStartTransitionTimeout](#) (double timeout)  
*Set the timeout for starting the sender and receiver threads.*

## Static Public Member Functions

- static [DataSenderManager](#) \* [GetDataSenderManagerPtr](#) ()  
*Gets a handle to the [DataSenderManager](#).*

## Static Public Attributes

- static const std::string [FRAGMENTS\\_PROCESSED\\_STAT\\_KEY](#)  
*Key for the Fragments Processed MonitoredQuantity.*
- static const std::string [INPUT\\_WAIT\\_STAT\\_KEY](#)  
*Key for the Input Wait MonitoredQuantity.*
- static const std::string [BUFFER\\_WAIT\\_STAT\\_KEY](#)  
*Key for the Fragment Buffer Wait MonitoredQuantity.*
- static const std::string [REQUEST\\_WAIT\\_STAT\\_KEY](#)  
*Key for the Request Buffer Wait MonitoredQuantity.*
- static const std::string [BRSYNC\\_WAIT\\_STAT\\_KEY](#)  
*Key for the Sync Wait MonitoredQuantity.*
- static const std::string [OUTPUT\\_WAIT\\_STAT\\_KEY](#)  
*Key for the Output Wait MonitoredQuantity.*
- static const std::string [FRAGMENTS\\_PER\\_READ\\_STAT\\_KEY](#)  
*Key for the Fragments Per Read MonitoredQuantity.*

### 6.22.1 Detailed Description

[BoardReaderCore](#) implements the state machine for the BoardReader artdaq application. It contains a [Commandable-FragmentGenerator](#), which generates Fragments which are then sent to a [DataSenderManager](#) by [BoardReaderCore](#).

Definition at line 24 of file BoardReaderCore.hh.

### 6.22.2 Constructor & Destructor Documentation

#### 6.22.2.1 artdaq::BoardReaderCore::BoardReaderCore ( [Commandable](#) & *parent\_application* )

[BoardReaderCore](#) Constructor.

Parameters

<i>parent - application</i>	Reference to parent <a href="#">Commandable</a> object, for in_run_failure notification
-----------------------------	---

Definition at line 35 of file BoardReaderCore.cc.

### 6.22.3 Member Function Documentation

#### 6.22.3.1 static [DataSenderManager](#)\* artdaq::BoardReaderCore::GetDataSenderManagerPtr ( ) [inline],[static]

Gets a handle to the [DataSenderManager](#).



**Returns**

Pointer to the [DataSenderManager](#)

Definition at line 175 of file BoardReaderCore.hh.

**6.22.3.2** `size_t artdaq::BoardReaderCore::GetFragmentsProcessed ( ) [inline]`

Get the number of Fragments processed this run

**Returns**

The number of Fragments processed this run

Definition at line 181 of file BoardReaderCore.hh.

**6.22.3.3** `bool artdaq::BoardReaderCore::GetReceiverThreadActive ( ) [inline]`

Get whether the receiver thread is still running.

**Returns**

Whether the receiver thread (calling CFG::getNext and putting Fragments into the [FragmentBuffer](#)) is still running

Definition at line 200 of file BoardReaderCore.hh.

**6.22.3.4** `bool artdaq::BoardReaderCore::GetSenderThreadActive ( ) [inline]`

Get whether the sender thread is still running.

**Returns**

Whether the sender thread (applying requests to [FragmentBuffer](#) and sending to [DataSenderManager](#)) is still running

Definition at line 195 of file BoardReaderCore.hh.

**6.22.3.5** `bool artdaq::BoardReaderCore::initialize ( fhicl::ParameterSet const & pset, uint64_t timeout, uint64_t timestamp )`

Initialize the [BoardReaderCore](#).

**Parameters**

<i>pset</i>	ParameterSet used to configure the <a href="#">BoardReaderCore</a>
<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

**Returns**

True if the initialize attempt succeeded

```
* BoardReaderCore accepts the following Parameters:
* "daq" (REQUIRED): FHiCL table containing DAQ configuration.
* "fragment_receiver" (REQUIRED): FHiCL table containing Fragment Receiver configuration.
*   See CommandableFragmentGenerator for configuration options.
* "generator" (Default: ""): The plugin name of the generator to load
* "rt_priority" (Default: 0): The unix priority to attempt to assign to the process
* "verbose" (Default: true): Whether to print transition messages
* "metrics": FHiCL table containing MetricManager configuration.
*   See MetricManager for configuration options.
*
```

Definition at line 61 of file BoardReaderCore.cc.

### 6.22.3.6 `bool artdaq::BoardReaderCore::metaCommand ( std::string const & command, std::string const & arg )`

Run a user-defined command on the [CommandableFragmentGenerator](#).

**Parameters**

<i>command</i>	Command name to run
<i>arg</i>	Argument(s) for command

**Returns**

Whether command completed successfully. (By convention, unsupported commands should return true)

Definition at line 607 of file BoardReaderCore.cc.

### 6.22.3.7 `BoardReaderCore& artdaq::BoardReaderCore::operator= ( BoardReaderCore const & ) [delete]`

Copy Assignment Operator is deleted.

**Returns**

[BoardReaderCore](#) copy

### 6.22.3.8 `bool artdaq::BoardReaderCore::pause ( uint64_t timeout, uint64_t timestamp )`

Pause the BoardReader, and the [CommandableFragmentGenerator](#).

**Parameters**

<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

**Returns**

True unless exception occurred

Definition at line 258 of file BoardReaderCore.cc.

## 6.22.3.9 void artdaq::BoardReaderCore::receive\_fragments ( )

Main working loop of the [BoardReaderCore](#).

This loop calls the [CommandableFragmentGenerator::getNext](#) method and gives the result to the [FragmentBuffer](#)

Definition at line 303 of file BoardReaderCore.cc.

## 6.22.3.10 bool artdaq::BoardReaderCore::reinitialize ( fhicl::ParameterSet const &amp; pset, uint64\_t timeout, uint64\_t timestamp )

Reinitialize the BoardReader. No-Op.

## Parameters

<i>pset</i>	ParameterSet used to configure the <a href="#">BoardReaderCore</a>
<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

## Returns

True unless exception occurred

Definition at line 295 of file BoardReaderCore.cc.

## 6.22.3.11 std::string artdaq::BoardReaderCore::report ( std::string const &amp; which ) const

Send a report on a given run-time quantity.

## Parameters

<i>which</i>	Which quantity to report
--------------	--------------------------

## Returns

A string containing the requested quantity.

If the [CommandableFragmentGenerator](#) has been initialized, [CommandableFragmentGenerator::report](#)(std::string const& which) will be called. Otherwise, the [BoardReaderCore](#) will return the current run number and an error message.

Definition at line 570 of file BoardReaderCore.cc.

## 6.22.3.12 bool artdaq::BoardReaderCore::resume ( uint64\_t timeout, uint64\_t timestamp )

Resume the BoardReader, and the [CommandableFragmentGenerator](#).

## Parameters

<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

## Returns

True unless exception occurred

Definition at line 267 of file BoardReaderCore.cc.

### 6.22.3.13 void artdaq::BoardReaderCore::send\_fragments ( )

Main working loop of the [BoardReaderCore](#), pt. 2.

This loop calls the [FragmentBuffer::applyRequests](#) method and sends the result using [DataSenderManager](#)

Definition at line 398 of file BoardReaderCore.cc.

### 6.22.3.14 void artdaq::BoardReaderCore::SetStartTransitionTimeout ( double *timeout* ) [inline]

Set the timeout for starting the sender and receiver threads.

#### Parameters

<i>timeout</i>	<a href="#">Timeout</a> , in seconds, for starting the sender and receiver threads
----------------	--

This function is used to communicate the start transition timeout from [BoardReaderApp](#) to the [BoardReaderCore](#) threads

Definition at line 207 of file BoardReaderCore.hh.

### 6.22.3.15 bool artdaq::BoardReaderCore::shutdown ( uint64\_t *timeout* )

Shutdown the BoardReader, and the [CommandableFragmentGenerator](#).

#### Parameters

<i>timeout</i>	<a href="#">Timeout</a> for transition
----------------	--

#### Returns

True unless exception occurred

Definition at line 277 of file BoardReaderCore.cc.

### 6.22.3.16 bool artdaq::BoardReaderCore::soft\_initialize ( fhicl::ParameterSet const & *pset*, uint64\_t *timeout*, uint64\_t *timestamp* )

Soft-Initialize the BoardReader. No-Op.

#### Parameters

<i>pset</i>	ParameterSet used to configure the <a href="#">BoardReaderCore</a>
<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

#### Returns

True unless exception occurred

Definition at line 287 of file BoardReaderCore.cc.

### 6.22.3.17 bool artdaq::BoardReaderCore::start ( art::RunID *id*, uint64\_t *timeout*, uint64\_t *timestamp* )

Start the BoardReader, and the [CommandableFragmentGenerator](#).

## Parameters

<i>id</i>	Run ID of new run
<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

## Returns

True unless exception occurred

Definition at line 206 of file BoardReaderCore.cc.

6.22.3.18 `bool artdaq::BoardReaderCore::stop ( uint64_t timeout, uint64_t timestamp )`

Stop the BoardReader, and the [CommandableFragmentGenerator](#).

## Parameters

<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

## Returns

True unless exception occurred

Definition at line 231 of file BoardReaderCore.cc.

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

- artdaq/artdaq/Application/BoardReaderCore.hh
- artdaq/artdaq/Application/BoardReaderCore.cc

## 6.23 artdaqtest::BrokenTransferTest Class Reference

A class which simulates several failure modes for TransferPlugins such as sender pause/restart and receiver pause/restart

```
#include <test/TransferPlugins/BrokenTransferTest.hh>
```

## Classes

- struct [Config](#)  
Configuration parameters for [BrokenTransferTest](#)

## Public Member Functions

- [BrokenTransferTest](#) (const fhicl::ParameterSet &ps)  
[BrokenTransferTest](#) Constructor
- void [TestSenderPause](#) ()  
Run the "Sender Paused" test
- void [TestReceiverPause](#) ()

*Run the "Receiver Paused" test*

- void [TestSenderReconnect](#) ( )

*Run the "Sender Reconnect" test*

- void [TestReceiverReconnect](#) (int send\_throttle\_factor=0)

*Run the "Receiver Reconnect" test*

### 6.23.1 Detailed Description

A class which simulates several failure modes for TransferPlugins such as sender pause/restart and receiver pause/restart

Definition at line 27 of file BrokenTransferTest.hh.

### 6.23.2 Constructor & Destructor Documentation

6.23.2.1 [artdaqtest::BrokenTransferTest::BrokenTransferTest](#) ( const fhicl::ParameterSet & *ps* )

[BrokenTransferTest](#) Constructor

Parameters

<i>ps</i>	ParameterSet containing <a href="#">BrokenTransferTest</a> configuration
-----------	--

Definition at line 18 of file BrokenTransferTest.cc.

### 6.23.3 Member Function Documentation

6.23.3.1 void [artdaqtest::BrokenTransferTest::TestReceiverPause](#) ( )

Run the "Receiver Paused" test

Definition at line 66 of file BrokenTransferTest.cc.

6.23.3.2 void [artdaqtest::BrokenTransferTest::TestReceiverReconnect](#) ( int *send\_throttle\_factor* = 0 )

Run the "Receiver Reconnect" test

Parameters

<i>send_throttle_factor</i>	Amount of time Sender should wait, in units of 1/fragment_rate
-----------------------------	--

Definition at line 124 of file BrokenTransferTest.cc.

6.23.3.3 void [artdaqtest::BrokenTransferTest::TestSenderPause](#) ( )

Run the "Sender Paused" test

Definition at line 46 of file BrokenTransferTest.cc.

## 6.23.3.4 void artdaqtest::BrokenTransferTest::TestSenderReconnect ( )

Run the "Sender Reconnect" test

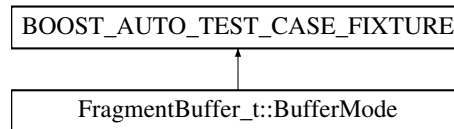
Definition at line 86 of file BrokenTransferTest.cc.

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

- artdaq/test/TransferPlugins/BrokenTransferTest.hh
- artdaq/test/TransferPlugins/BrokenTransferTest.cc

## 6.24 FragmentBuffer\_t::BufferMode Struct Reference

Inheritance diagram for FragmentBuffer\_t::BufferMode:



### Public Member Functions

- void **test\_method** ( )

#### 6.24.1 Detailed Description

Definition at line 232 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.25 FragmentBuffer\_t::BufferMode\_id Struct Reference

#### 6.25.1 Detailed Description

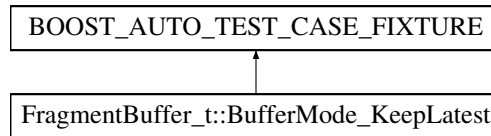
Definition at line 232 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.26 FragmentBuffer\_t::BufferMode\_KeepLatest Struct Reference

Inheritance diagram for FragmentBuffer\_t::BufferMode\_KeepLatest:



## Public Member Functions

- void **test\_method** ()

### 6.26.1 Detailed Description

Definition at line 320 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.27 FragmentBuffer\_t::BufferMode\_KeepLatest\_id Struct Reference

### 6.27.1 Detailed Description

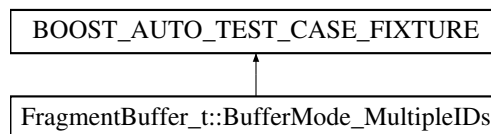
Definition at line 320 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.28 FragmentBuffer\_t::BufferMode\_MultipleIDs Struct Reference

Inheritance diagram for FragmentBuffer\_t::BufferMode\_MultipleIDs:



## Public Member Functions

- void **test\_method** ()

### 6.28.1 Detailed Description

Definition at line 1345 of file FragmentBuffer\_t.cc.

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



- `artdaq/test/DAQrate/FragmentBuffer_t.cc`

## 6.29 FragmentBuffer\_t::BufferMode\_MultipleIDs\_id Struct Reference

### 6.29.1 Detailed Description

Definition at line 1345 of file `FragmentBuffer_t.cc`.

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

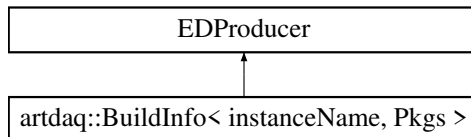
- `artdaq/test/DAQrate/FragmentBuffer_t.cc`

## 6.30 artdaq::BuildInfo< instanceName, Pkgs > Class Template Reference

[BuildInfo](#) is an `art::EDProducer` which saves information about package builds to the data file.

```
#include <artdaq/ArtModules/BuildInfo_module.hh>
```

Inheritance diagram for `artdaq::BuildInfo< instanceName, Pkgs >`:



### Public Member Functions

- [BuildInfo](#) (`fhicl::ParameterSet const &p`)  
*BuildInfo module Constructor.*
- `virtual ~BuildInfo ()=default`  
*Default Destructor.*
- `void beginRun (art::Run &r) override`  
*Perform actions at the beginning of the Run.*
- `void produce (art::Event &e) finaloverride`  
*Perform actions for each event.*

### 6.30.1 Detailed Description

```
template<std::string * instanceName, typename... Pkgs>class artdaq::BuildInfo< instanceName, Pkgs >
```

[BuildInfo](#) is an `art::EDProducer` which saves information about package builds to the data file.

#### Template Parameters

---

<i>instanceName</i>	Tag which the <a href="#">BuildInfo</a> objects will be saved under
<i>Pkgs</i>	List of package <a href="#">BuildInfo</a> types

Definition at line 26 of file BuildInfo\_module.hh.

## 6.30.2 Constructor & Destructor Documentation

6.30.2.1 `template<std::string * instanceName, typename... Pkgs> artdaq::BuildInfo< instanceName, Pkgs >::BuildInfo ( fhicl::ParameterSet const & p ) [explicit]`

[BuildInfo](#) module Constructor.

Parameters

<i>p</i>	ParameterSet used to configure <a href="#">BuildInfo</a> module
----------	---

BuildInfo\_module expects the following Parameters: "instance\_name": Name which the [BuildInfo](#) information will be saved under

Definition at line 93 of file BuildInfo\_module.hh.

## 6.30.3 Member Function Documentation

6.30.3.1 `template<std::string * instanceName, typename... Pkgs> void artdaq::BuildInfo< instanceName, Pkgs >::beginRun ( art::Run & r ) [override]`

Perform actions at the beginning of the Run.

Parameters

<i>r</i>	art::Run object
----------	-----------------

The [BuildInfo](#) information is stored in the Run-level provenance, so this method performs most of the "work" for this module.

Definition at line 104 of file BuildInfo\_module.hh.

6.30.3.2 `template<std::string * instanceName, typename... Pkgs> void artdaq::BuildInfo< instanceName, Pkgs >::produce ( art::Event & e ) [final],[override]`

Perform actions for each event.

Parameters

<i>e</i>	art::Event object
----------	-------------------

This function is a required override for EDProducer, and is a No-Op in BuildInfo\_module.

Definition at line 128 of file BuildInfo\_module.hh.

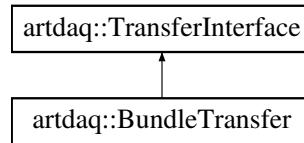
The documentation for this class was generated from the following file:

- artdaq/artdaq/ArtModules/BuildInfo\_module.hh

## 6.31 artdaq::BundleTransfer Class Reference

The [BundleTransfer TransferInterface](#) plugin sets up a Shmem\_transfer plugin or TCPSocket\_transfer plugin depending if the source and destination are on the same host, to maximize throughput.

Inheritance diagram for artdaq::BundleTransfer:



### Public Member Functions

- [BundleTransfer](#) (const fhicl::ParameterSet &pset, [Role](#) role)  
*BundleTransfer* Constructor.
- [~BundleTransfer](#) () override  
*BundleTransfer* default Destructor.
- int [receiveFragment](#) (artdaq::Fragment &fragment, size\_t receiveTimeout) override  
*Receive a Fragment, using the underlying transfer plugin.*
- int [receiveFragmentHeader](#) (detail::RawFragmentHeader &header, size\_t receiveTimeout) override  
*Receive a Fragment Header from the transport mechanism.*
- int [receiveFragmentData](#) (RawDataType \*destination, size\_t) override  
*Receive the body of a Fragment to the given destination pointer.*
- [CopyStatus](#) transfer\_fragment\_min\_blocking\_mode (artdaq::Fragment const &fragment, size\_t send\_timeout\_ - usec) override  
*Send a Fragment in non-reliable mode, using the underlying transfer plugin.*
- [CopyStatus](#) transfer\_fragment\_reliable\_mode (artdaq::Fragment &&fragment) override  
*Send a Fragment in reliable mode, using the underlying transfer plugin.*
- bool [isRunning](#) () override  
*Determine whether the [TransferInterface](#) plugin is able to send/receive data.*
- void [flush\\_buffers](#) () override  
*Flush any in-flight data. This should be used by the receiver after the receive loop has ended.*

### Additional Inherited Members

#### 6.31.1 Detailed Description

The [BundleTransfer TransferInterface](#) plugin sets up a Shmem\_transfer plugin or TCPSocket\_transfer plugin depending if the source and destination are on the same host, to maximize throughput.

Definition at line 19 of file Bundle\_transfer.cc.

#### 6.31.2 Constructor & Destructor Documentation

##### 6.31.2.1 artdaq::BundleTransfer::BundleTransfer ( const fhicl::ParameterSet & pset, [Role](#) role )

[BundleTransfer](#) Constructor.

## Parameters

<i>pset</i>	ParameterSet used to configure <a href="#">BundleTransfer</a>
<i>role</i>	Role of this <a href="#">TransferInterface</a> , either kReceive or kSend

Definition at line 220 of file Bundle\_transfer.cc.

### 6.31.3 Member Function Documentation

**6.31.3.1** `bool artdaq::BundleTransfer::isRunning ( )` `[inline]`, `[override]`, `[virtual]`

Determine whether the [TransferInterface](#) plugin is able to send/receive data.

## Returns

True if the [TransferInterface](#) plugin is currently able to send/receive data

Reimplemented from [artdaq::TransferInterface](#).

Definition at line 177 of file Bundle\_transfer.cc.

**6.31.3.2** `int artdaq::BundleTransfer::receiveFragment ( artdaq::Fragment & fragment, size_t receiveTimeout )` `[inline]`, `[override]`, `[virtual]`

Receive a Fragment, using the underlying transfer plugin.

## Parameters

<i>fragment</i>	Output Fragment
<i>receiveTimeout</i>	Time to wait before returning <a href="#">TransferInterface::RECV_TIMEOUT</a>

## Returns

Rank of sender

Reimplemented from [artdaq::TransferInterface](#).

Definition at line 40 of file Bundle\_transfer.cc.

**6.31.3.3** `int artdaq::BundleTransfer::receiveFragmentData ( RawDataType * destination, size_t )` `[inline]`, `[override]`, `[virtual]`

Receive the body of a Fragment to the given destination pointer.

## Parameters

<i>destination</i>	Pointer to memory region where Fragment data should be stored
<i>wordCount</i>	Number of words of Fragment data to receive

## Returns

The rank the Fragment was received from (should be `source_rank`), or `RECV_TIMEOUT`

Implements [artdaq::TransferInterface](#).

Definition at line 86 of file Bundle\_transfer.cc.

```
6.31.3.4  int artdaq::BundleTransfer::receiveFragmentHeader ( detail::RawFragmentHeader & header, size_t receiveTimeout )  
          [inline],[override],[virtual]
```

Receive a Fragment Header from the transport mechanism.

## Parameters

out	header	Received Fragment Header
	receiveTimeout	Timeout for receive

## Returns

The rank the Fragment was received from (should be source\_rank), or RECV\_TIMEOUT

Implements [artdaq::TransferInterface](#).

Definition at line 67 of file Bundle\_transfer.cc.

**6.31.3.5 CopyStatus** `artdaq::BundleTransfer::transfer_fragment_min_blocking_mode ( artdaq::Fragment const & fragment, size_t send_timeout_usec )` [inline],[override],[virtual]

Send a Fragment in non-reliable mode, using the underlying transfer plugin.

## Parameters

fragment	The Fragment to send
send_timeout_ - usec	How long to wait before aborting. Defaults to size_t::MAX_VALUE

## Returns

A [TransferInterface::CopyStatus](#) result variable

Implements [artdaq::TransferInterface](#).

Definition at line 109 of file Bundle\_transfer.cc.

**6.31.3.6 CopyStatus** `artdaq::BundleTransfer::transfer_fragment_reliable_mode ( artdaq::Fragment && fragment )` [inline],[override],[virtual]

Send a Fragment in reliable mode, using the underlying transfer plugin.

## Parameters

fragment	The Fragment to send
----------	----------------------

## Returns

A [TransferInterface::CopyStatus](#) result variable

Implements [artdaq::TransferInterface](#).

Definition at line 147 of file Bundle\_transfer.cc.

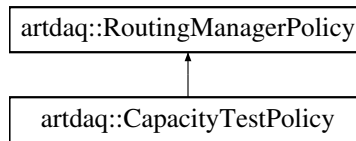
The documentation for this class was generated from the following file:

- `artdaq/artdaq/TransferPlugins/Bundle_transfer.cc`

## 6.32 artdaq::CapacityTestPolicy Class Reference

A [RoutingManagerPolicy](#) which tries to fully load the first receiver, then the second, and so on.

Inheritance diagram for artdaq::CapacityTestPolicy:



## Public Member Functions

- [CapacityTestPolicy](#) (const fhicl::ParameterSet &ps)  
*CapacityTestPolicy Constructor.*
- [~CapacityTestPolicy](#) () override=default  
*Default virtual Destructor.*
- virtual void [CreateRoutingTable](#) (detail::RoutingPacket &output) override  
*Add entries to the given RoutingPacket using currently-held tokens.*
- virtual [detail::RoutingPacketEntry](#) [CreateRouteForSequenceID](#) (artdaq::Fragment::sequence\_id\_t seq, int requesting\_rank) override  
*Get an [artdaq::detail::RoutingPacketEntry](#) for a given sequence ID and rank. Used by RequestBasedEventBuilder and DataFlow RoutingManagerMode.*

## Additional Inherited Members

### 6.32.1 Detailed Description

A [RoutingManagerPolicy](#) which tries to fully load the first receiver, then the second, and so on.

Definition at line 10 of file CapacityTest\_policy.cc.

### 6.32.2 Constructor & Destructor Documentation

6.32.2.1 [artdaq::CapacityTestPolicy::CapacityTestPolicy](#) ( const fhicl::ParameterSet & ps ) [explicit]

[CapacityTestPolicy](#) Constructor.

Parameters

<i>ps</i>	ParameterSet used to configure the <a href="#">CapacityTestPolicy</a>
-----------	---

\* CapacityTestPolicy accepts the following Parameters:  
 \* "tokens\_used\_per\_table\_percent" (Default: 50): Percentage of available tokens to be used on each iteration.  
 \*

Definition at line 60 of file CapacityTest\_policy.cc.

### 6.32.3 Member Function Documentation

6.32.3.1 **detail::RoutingPacketEntry** `artdaq::CapacityTestPolicy::CreateRouteForSequenceID (`  
`artdaq::Fragment::sequence_id_t seq, int requesting_rank )` `[override],[virtual]`

Get an [artdaq::detail::RoutingPacketEntry](#) for a given sequence ID and rank. Used by RequestBasedEventBuilder and DataFlow RoutingManagerMode.



## Parameters

<i>seq</i>	Sequence Number to get route for
<i>requesting_rank</i>	Rank to route for

## Returns

[artdaq::detail::RoutingPacketEntry](#) connecting sequence ID to destination rank

Implements [artdaq::RoutingManagerPolicy](#).

Definition at line 95 of file CapacityTest\_policy.cc.

**6.32.3.2** `void artdaq::CapacityTestPolicy::CreateRoutingTable ( detail::RoutingPacket & output ) [override], [virtual]`

Add entries to the given RoutingPacket using currently-held tokens.

## Parameters

<i>output</i>	RoutingPacket to add entries to
---------------	---------------------------------

[CapacityTestPolicy](#) will assign available tokens from the first receiver, then the second, and so on until it has assigned tokens equal to the `inital_token_count * tokens_used_per_table_percent / 100`. The idea is that in steady-state, the load on the receivers should reflect the workload relative to the capacity of the system. (i.e. if you have 5 receivers, and 3 of them are 100% busy, then your load factor is approximately 60%.)

Implements [artdaq::RoutingManagerPolicy](#).

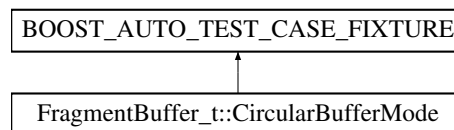
Definition at line 65 of file CapacityTest\_policy.cc.

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

- `artdaq/artdaq/RoutingPolicies/CapacityTest_policy.cc`

## 6.33 FragmentBuffer\_t::CircularBufferMode Struct Reference

Inheritance diagram for `FragmentBuffer_t::CircularBufferMode`:



## Public Member Functions

- `void test_method ()`

### 6.33.1 Detailed Description

Definition at line 407 of file `FragmentBuffer_t.cc`.

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

- `artdaq/test/DAQrate/FragmentBuffer_t.cc`

## 6.34 `FragmentBuffer_t::CircularBufferMode_id` Struct Reference

### 6.34.1 Detailed Description

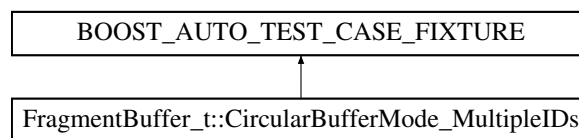
Definition at line 407 of file `FragmentBuffer_t.cc`.

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

- `artdaq/test/DAQrate/FragmentBuffer_t.cc`

## 6.35 `FragmentBuffer_t::CircularBufferMode_MultipleIDs` Struct Reference

Inheritance diagram for `FragmentBuffer_t::CircularBufferMode_MultipleIDs`:



### Public Member Functions

- `void test_method ()`

### 6.35.1 Detailed Description

Definition at line 1469 of file `FragmentBuffer_t.cc`.

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

- `artdaq/test/DAQrate/FragmentBuffer_t.cc`

## 6.36 `FragmentBuffer_t::CircularBufferMode_MultipleIDs_id` Struct Reference

### 6.36.1 Detailed Description

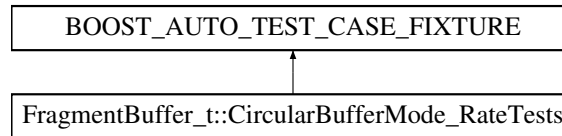
Definition at line 1469 of file `FragmentBuffer_t.cc`.

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

- `artdaq/test/DAQrate/FragmentBuffer_t.cc`

## 6.37 FragmentBuffer\_t::CircularBufferMode\_RateTests Struct Reference

Inheritance diagram for FragmentBuffer\_t::CircularBufferMode\_RateTests:



### Public Member Functions

- void **test\_method** ()

#### 6.37.1 Detailed Description

Definition at line 2250 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.38 FragmentBuffer\_t::CircularBufferMode\_RateTests\_id Struct Reference

#### 6.38.1 Detailed Description

Definition at line 2250 of file FragmentBuffer\_t.cc.

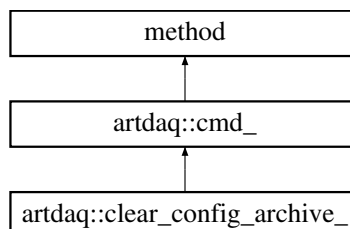
The documentation for this struct was generated from the following file:

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.39 artdaq::clear\_config\_archive\_ Class Reference

[clear\\_config\\_archive\\_](#) Command class

Inheritance diagram for artdaq::clear\_config\_archive\_:



## Public Member Functions

- [clear\\_config\\_archive\\_](#) ([xmlrpc\\_commander](#) &c)

[clear\\_config\\_archive\\_](#) Constructor

## Additional Inherited Members

### 6.39.1 Detailed Description

[clear\\_config\\_archive\\_](#) Command class

Definition at line 1270 of file `xmlrpc_commander.cc`.

### 6.39.2 Constructor & Destructor Documentation

6.39.2.1 `artdaq::clear_config_archive_::clear_config_archive_ ( xmlrpc_commander & c )` `[inline]`

[clear\\_config\\_archive\\_](#) Constructor

#### Parameters

<code>c</code>	<a href="#">xmlrpc_commander</a> to send transition commands to
----------------	---

Definition at line 1277 of file `xmlrpc_commander.cc`.

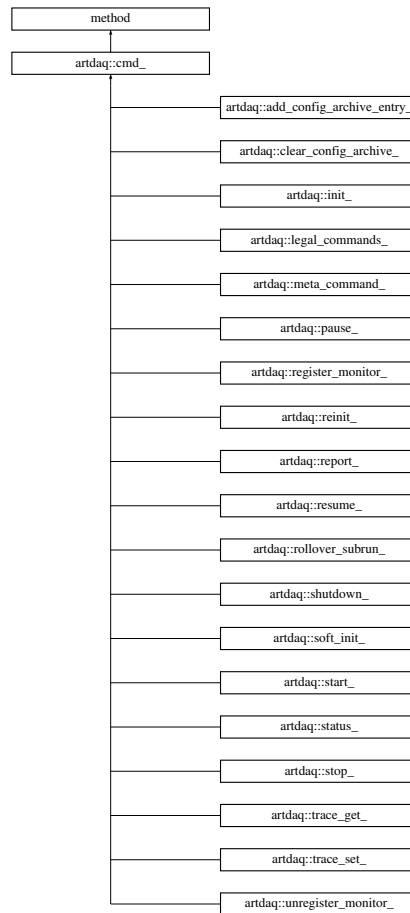
The documentation for this class was generated from the following file:

- `artdaq/artdaq/ExternalComms/xmlrpc_commander.cc`

## 6.40 artdaq::cmd\_ Class Reference

The "cmd\_" class serves as the base class for all artdaq's XML-RPC commands.

Inheritance diagram for `artdaq::cmd_`:



## Public Member Functions

- `cmd_ (xmlrpc_commander &c, const std::string &signature, const std::string &description)`  
*cmd\_ Constructor*
- `void execute (const xmlrpc_c::paramList &paramList, xmlrpc_c::value *retValP) final`  
*Execute the command with the given parameters.*

## Protected Member Functions

- `virtual bool execute_ (const xmlrpc_c::paramList &, xmlrpc_c::value *retValP)=0`  
*"execute\_" is a wrapper function around the call to the commandable object's function*
- `template<typename T >`  
`T getParam (const xmlrpc_c::paramList &paramList, int index)`  
*Get a parameter from the parameter list.*
- `template<typename T >`  
`T getParam (const xmlrpc_c::paramList &paramList, int index, T default_value)`  
*Get a parameter from the parameter list, returning a default value if not found at specified location.*
- `template<>`  
`uint64_t getParam (const xmlrpc_c::paramList &paramList, int index)`  
*Get a parameter from the parameter list.*

- `template<>`  
`uint32_t getParam (const xmlrpc_c::paramList &paramList, int index)`  
*Get a parameter from the parameter list.*

## Protected Attributes

- `xmlrpc\_commander & \_c`  
*The [xmlrpc\\_commander](#) instance that the command will be sent to.*

### 6.40.1 Detailed Description

The "cmd\_" class serves as the base class for all artdaq's XML-RPC commands.

JCF, 9/5/14

The "cmd\_" class serves as the base class for all artdaq's XML-RPC commands, all of which use the code in the "execute()" function; each specific command type deriving from `cmd_` is implemented in the `execute_()` function which `execute()` calls (notice the underscore), and optionally sets the `retvalP` parameter

`cmd_` contains a set of template functions, `getParam<T>()`, which are designed to prevent implementors of derived classes from having to worry about interfacing directly with `xmlrpc_c`'s parameter-getting functionality

Definition at line 499 of file `xmlrpc_commander.cc`.

### 6.40.2 Constructor & Destructor Documentation

6.40.2.1 `artdaq::cmd_::cmd_ ( xmlrpc\_commander & c, const std::string & signature, const std::string & description )`  
`[inline]`

`cmd_` Constructor

Parameters

<i>c</i>	<a href="#">xmlrpc_commander</a> instance
<i>signature</i>	Signature of the command
<i>description</i>	Description of the command

Definition at line 509 of file `xmlrpc_commander.cc`.

### 6.40.3 Member Function Documentation

6.40.3.1 `void artdaq::cmd_::execute ( const xmlrpc_c::paramList & paramList, xmlrpc_c::value * retvalP )` `[final]`

Execute the command with the given parameters.

Parameters

<i>paramList</i>	List of parameters for the command (i.e. a fhicl string for init transitions)
<i>retvalP</i>	Pointer to the return value (usually a string describing result of command)

Definition at line 718 of file `xmlrpc_commander.cc`.

6.40.3.2 `virtual bool artdaq::cmd_::execute_ ( const xmlrpc_c::paramList & , xmlrpc_c::value * retvalP )` [protected],  
[pure virtual]

"execute\_" is a wrapper function around the call to the commandable object's function

## Parameters

<i>retvalP</i>	Pointer to the return value (usually a string describing result of command)
----------------	---

## Returns

Whether the command succeeded

6.40.3.3 `template<typename T > T artdaq::cmd_::getParam ( const xmlrpc_c::paramList & paramList, int index )`  
`[protected]`

Get a parameter from the parameter list.

## Template Parameters

<i>T</i>	Type of the parameter
----------	-----------------------

## Parameters

<i>paramList</i>	The parameter list
<i>index</i>	Index of the parameter in the parameter list

## Returns

The requested parameter

Template specilization is used to provide valid overloads

Definition at line 576 of file xmlrpc\_commander.cc.

6.40.3.4 `template<typename T > T artdaq::cmd_::getParam ( const xmlrpc_c::paramList & paramList, int index, T default_value )`  
`[protected]`

Get a parameter from the parameter list, returning a default value if not found at specified location.

## Template Parameters

<i>T</i>	Type of the parameter
----------	-----------------------

## Parameters

<i>paramList</i>	The parameter list
<i>index</i>	Index of the parameter in the parameter list
<i>default_value</i>	Default value to return if exception retrieving parameter

## Returns

The requested parameter, or the default value if there was an exception retrieving the parameter

JCF, 9/5/14

Here, if getParam throws an exception due to a lack of an existing parameter, swallow the exception and return the default value passed to the function

Surprisingly, if an invalid index is supplied, although getParam throws an exception that exception is neither xmlrpc\_c's girrr:error nor boost::bad\_lexical\_cast. Although it's less than ideal, we'll swallow almost all exceptions in the call to



getParam, as an invalid index value simply means the user wishes to employ the default\_value. I say "almost" because the only exception we don't swallow here is if an invalid parameter type "T" was supplied

Definition at line 706 of file xmlrpc\_commander.cc.

6.40.3.5 `template<> uint64_t artdaq::cmd_::getParam ( const xmlrpc_c::paramList & paramList, int index )` [protected]

Get a parameter from the parameter list.

#### Parameters

<i>paramList</i>	The parameter list
<i>index</i>	Index of the parameter in the parameter list

#### Returns

The requested parameter

This specialized cmd\_getParam for the uint64\_t type

Definition at line 590 of file xmlrpc\_commander.cc.

6.40.3.6 `template<> uint32_t artdaq::cmd_::getParam ( const xmlrpc_c::paramList & paramList, int index )` [protected]

Get a parameter from the parameter list.

#### Parameters

<i>paramList</i>	The parameter list
<i>index</i>	Index of the parameter in the parameter list

#### Returns

The requested parameter

This specialized cmd\_getParam for the uint64\_t type

Definition at line 614 of file xmlrpc\_commander.cc.

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

- artdaq/artdaq/ExternalComms/xmlrpc\_commander.cc

## 6.41 artdaq::Commandable Class Reference

[Commandable](#) is the base class for all artdaq components which implement the artdaq state machine.

```
#include <artdaq/Application/Commandable.hh>
```

Inheritance diagram for artdaq::Commandable:



## Public Member Functions

- [Commandable](#) ()
- [Commandable](#) ([Commandable](#) const &)=delete  
*Copy Constructor is deleted.*
- virtual [~Commandable](#) ()=default  
*Default Destructor.*
- [Commandable](#) & [operator=](#) ([Commandable](#) const &)=delete  
*Copy Assignment operator is deleted.*
- [Commandable](#) ([Commandable](#) &&)=delete  
*Move Constructor is deleted.*
- [Commandable](#) & [operator=](#) ([Commandable](#) &&)=delete  
*Move Assignment Operator is deleted.*
- bool [initialize](#) (fhicl::ParameterSet const &pset, uint64\_t timeout, uint64\_t timestamp)  
*Processes the initialize request.*
- bool [start](#) (art::RunID id, uint64\_t timeout, uint64\_t timestamp)  
*Processes the start transition.*
- bool [stop](#) (uint64\_t timeout, uint64\_t timestamp)  
*Processes the stop transition.*
- bool [pause](#) (uint64\_t timeout, uint64\_t timestamp)  
*Processes the pause transition.*
- bool [resume](#) (uint64\_t timeout, uint64\_t timestamp)  
*Processes the resume transition.*
- bool [shutdown](#) (uint64\_t timeout)  
*Processes the shutdown transition.*
- bool [soft\\_initialize](#) (fhicl::ParameterSet const &pset, uint64\_t timeout, uint64\_t timestamp)  
*Processes the soft-initialize request.*
- bool [reinitialize](#) (fhicl::ParameterSet const &pset, uint64\_t timeout, uint64\_t timestamp)  
*Processes the reinitialize request.*
- bool [in\\_run\\_failure](#) ()  
*Actions taken when the in\_run\_failure state is set.*
- virtual std::string [report](#) (std::string const &) const  
*Default report implementation returns current report\_string.*
- std::string [status](#) () const  
*Returns the current state of the [Commandable](#).*
- virtual std::string [register\\_monitor](#) (fhicl::ParameterSet const &)  
*Perform the register\_monitor action.*
- virtual std::string [unregister\\_monitor](#) (std::string const &)  
*Perform the unregister\_monitor action.*
- std::vector< std::string > [legal\\_commands](#) () const  
*Get the legal transition commands from the current state.*
- virtual bool [do\\_initialize](#) (fhicl::ParameterSet const &, uint64\_t, uint64\_t)  
*Perform the initialize transition.*
- virtual bool [do\\_start](#) (art::RunID, uint64\_t, uint64\_t)  
*Perform the start transition.*
- virtual bool [do\\_stop](#) (uint64\_t, uint64\_t)  
*Perform the stop transition.*

- virtual bool `do_pause` (uint64\_t, uint64\_t)  
*Perform the pause transition.*
- virtual bool `do_resume` (uint64\_t, uint64\_t)  
*Perform the resume transition.*
- virtual bool `do_shutdown` (uint64\_t)  
*Perform the shutdown transition.*
- virtual bool `do_reinitialize` (fhicl::ParameterSet const &, uint64\_t, uint64\_t)  
*Perform the reinitialize transition.*
- virtual bool `do_soft_initialize` (fhicl::ParameterSet const &, uint64\_t, uint64\_t)  
*Perform the soft\_initialize transition.*
- virtual bool `do_rollover_subrun` (uint64\_t eventNum, uint32\_t subrunNum)  
*Perform the rollover\_subrun transition.*
- virtual void `badTransition` (const std::string &trans)  
*This function is called when an attempt is made to call an illegal transition.*
- virtual void `BootedEnter` ()  
*Perform actions upon entering the Booted state.*
- virtual void `InRunExit` ()  
*Perform actions upon leaving the InRun state.*
- virtual std::string `do_trace_get` (std::string const &name)  
*Get the TRACE mask for the given TRACE name. If name is "ALL", then all TRACE masks will be printed.*
- virtual bool `do_trace_set` (std::string const &name, std::string const &type, std::string const &mask\_in\_string\_  
form)  
*Set the given TRACE mask for the given TRACE name.*
- virtual bool `do_meta_command` (std::string const &cmd, std::string const &args)  
*Run a module-defined command with the given parameter string.*
- virtual bool `do_add_config_archive_entry` (std::string const &, std::string const &)  
*Add the specified key-value pair to the configuration archive list.*
- virtual bool `do_clear_config_archive` ()  
*Clears the configuration archive list.*

## Protected Member Functions

- std::string `current_state` () const  
*Return the name of the current state.*

## Protected Attributes

- CommandableContext `fsm_`  
*The generated State Machine (using smc\_compiler)*
- bool `external_request_status_`  
*Whether the last command succeeded.*
- std::string `report_string_`  
*Status information about the last command.*

### 6.41.1 Detailed Description

[Commandable](#) is the base class for all artdaq components which implement the artdaq state machine.

Definition at line 22 of file Commandable.hh.

### 6.41.2 Constructor & Destructor Documentation

#### 6.41.2.1 `artdaq::Commandable::Commandable ( )`

Default constructor.

Definition at line 24 of file Commandable.cc.

### 6.41.3 Member Function Documentation

#### 6.41.3.1 `void artdaq::Commandable::badTransition ( const std::string & trans ) [virtual]`

This function is called when an attempt is made to call an illegal transition.

Parameters

<i>trans</i>	The transition that was attempted
--------------	-----------------------------------

Definition at line 403 of file Commandable.cc.

#### 6.41.3.2 `void artdaq::Commandable::BootedEnter ( ) [virtual]`

Perform actions upon entering the Booted state.

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::RoutingManagerApp](#), [artdaq::BoardReaderApp](#), and [artdaq::EventBuilderApp](#).

Definition at line 422 of file Commandable.cc.

#### 6.41.3.3 `std::string artdaq::Commandable::current_state ( ) const [protected]`

Return the name of the current state.

Returns

The name of the current state

Definition at line 556 of file Commandable.cc.

#### 6.41.3.4 `bool artdaq::Commandable::do_add_config_archive_entry ( std::string const & key, std::string const & value ) [virtual]`

Add the specified key-value pair to the configuration archive list.

**Returns**

Whether the command succeeded (always true)

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::EventBuilderApp](#), and [artdaq::DataLoggerApp](#).

Definition at line 539 of file Commandable.cc.

**6.41.3.5 bool artdaq::Commandable::do\_clear\_config\_archive ( ) [virtual]**

Clears the configuration archive list.

**Returns**

Whether the command succeeded (always true)

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::EventBuilderApp](#), and [artdaq::DataLoggerApp](#).

Definition at line 545 of file Commandable.cc.

**6.41.3.6 bool artdaq::Commandable::do\_initialize ( fhicl::ParameterSet const &, uint64\_t, uint64\_t ) [virtual]**

Perform the initialize transition.

**Returns**

Whether the transition succeeded

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::RoutingManagerApp](#), [artdaq::DispatcherApp](#), [artdaq::BoardReaderApp](#), [artdaq::DataLoggerApp](#), and [artdaq::EventBuilderApp](#).

Definition at line 347 of file Commandable.cc.

**6.41.3.7 bool artdaq::Commandable::do\_meta\_command ( std::string const & cmd, std::string const & args ) [virtual]**

Run a module-defined command with the given parameter string.

This function is a No-Op. Derived classes should override it.

**Parameters**

<i>cmd</i>	Name of the command to run (implementation-defined)
<i>args</i>	Any arguments for the command (implementation-defined)

**Returns**

Whether the command succeeded (always true)

Reimplemented in [artdaq::BoardReaderApp](#).

Definition at line 526 of file Commandable.cc.

6.41.3.8 `bool artdaq::Commandable::do_pause ( uint64_t, uint64_t ) [virtual]`

Perform the pause transition.

#### Returns

Whether the transition succeeded

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::RoutingManagerApp](#), [artdaq::BoardReaderApp](#), [artdaq::DispatcherApp](#), [artdaq::Data-LoggerApp](#), and [artdaq::EventBuilderApp](#).

Definition at line 368 of file Commandable.cc.

6.41.3.9 `bool artdaq::Commandable::do_reinitialize ( fhicl::ParameterSet const &, uint64_t, uint64_t ) [virtual]`

Perform the reinitialize transition.

#### Returns

Whether the transition succeeded

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::RoutingManagerApp](#), [artdaq::BoardReaderApp](#), [artdaq::DispatcherApp](#), [artdaq::Event-BuilderApp](#), and [artdaq::DataLoggerApp](#).

Definition at line 389 of file Commandable.cc.

6.41.3.10 `bool artdaq::Commandable::do_resume ( uint64_t, uint64_t ) [virtual]`

Perform the resume transition.

#### Returns

Whether the transition succeeded

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::RoutingManagerApp](#), [artdaq::BoardReaderApp](#), [artdaq::DispatcherApp](#), [artdaq::Data-LoggerApp](#), and [artdaq::EventBuilderApp](#).

Definition at line 375 of file Commandable.cc.

6.41.3.11 `bool artdaq::Commandable::do_rollover_subrun ( uint64_t eventNum, uint32_t subrunNum ) [virtual]`

Perform the rollover\_subrun transition.

#### Parameters

<i>eventNum</i>	Sequence ID of boundary
-----------------	-------------------------

<i>subrunNum</i>	New Subrun Number
------------------	-------------------

**Returns**

Whether the transition succeeded

Reimplemented in [artdaq::EventBuilderApp](#).

Definition at line 532 of file Commandable.cc.

**6.41.3.12 bool artdaq::Commandable::do\_shutdown ( uint64\_t ) [virtual]**

Perform the shutdown transition.

**Returns**

Whether the transition succeeded

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::RoutingManagerApp](#), [artdaq::BoardReaderApp](#), [artdaq::DispatcherApp](#), [artdaq::Data-LoggerApp](#), and [artdaq::EventBuilderApp](#).

Definition at line 382 of file Commandable.cc.

**6.41.3.13 bool artdaq::Commandable::do\_soft\_initialize ( fhicl::ParameterSet const &, uint64\_t, uint64\_t ) [virtual]**

Perform the soft\_initialize transition.

**Returns**

Whether the transition succeeded

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::RoutingManagerApp](#), [artdaq::BoardReaderApp](#), [artdaq::DispatcherApp](#), [artdaq::Event-BuilderApp](#), and [artdaq::DataLoggerApp](#).

Definition at line 396 of file Commandable.cc.

**6.41.3.14 bool artdaq::Commandable::do\_start ( art::RunID, uint64\_t, uint64\_t ) [virtual]**

Perform the start transition.

**Returns**

Whether the transition succeeded

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::RoutingManagerApp](#), [artdaq::BoardReaderApp](#), [artdaq::DispatcherApp](#), [artdaq::Data-LoggerApp](#), and [artdaq::EventBuilderApp](#).

Definition at line 354 of file Commandable.cc.

6.41.3.15 `bool artdaq::Commandable::do_stop ( uint64_t, uint64_t ) [virtual]`

Perform the stop transition.

#### Returns

Whether the transition succeeded

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::RoutingManagerApp](#), [artdaq::BoardReaderApp](#), [artdaq::DispatcherApp](#), [artdaq::Data-LoggerApp](#), and [artdaq::EventBuilderApp](#).

Definition at line 361 of file Commandable.cc.

6.41.3.16 `std::string artdaq::Commandable::do_trace_get ( std::string const & name ) [virtual]`

Get the TRACE mask for the given TRACE name If name is "ALL", then all TRACE masks will be printed.

This function is implemented in [Commandable](#), derived classes may override if necessary.

#### Parameters

<i>name</i>	TRACE name to print mask for. "ALL" prints all TRACE masks
-------------	--

#### Returns

TRACE mask of the given TRACE name

Definition at line 436 of file Commandable.cc.

6.41.3.17 `bool artdaq::Commandable::do_trace_set ( std::string const & name, std::string const & type, std::string const & mask_in_string_form ) [virtual]`

Set the given TRACE mask for the given TRACE name.

This function is implemented in [Commandable](#), derived classes may override if necessary.

#### Parameters

<i>name</i>	Name of the TRACE level to set mask for
<i>type</i>	Type of TRACE mask to set (either M, S, or T)
<i>mask_in_string_form</i>	Mask to set

#### Returns

Whether the command succeeded (always true)

Definition at line 473 of file Commandable.cc.

6.41.3.18 `bool artdaq::Commandable::in_run_failure ( )`

Actions taken when the in\_run\_failure state is set.



**Returns**

Whether the notification was properly processed

Definition at line 270 of file Commandable.cc.

**6.41.3.19** `bool artdaq::Commandable::initialize ( fhicl::ParameterSet const & pset, uint64_t timeout, uint64_t timestamp )`

Processes the initialize request.

**Parameters**

<i>pset</i>	ParameterSet used to configure the <a href="#">Commandable</a>
<i>timeout</i>	<a href="#">Timeout</a> for init step
<i>timestamp</i>	Timestamp of init step

**Returns**

Whether the transition was successful

Definition at line 33 of file Commandable.cc.

**6.41.3.20** `void artdaq::Commandable::InRunExit ( ) [virtual]`

Perform actions upon leaving the InRun state.

This function is a No-Op. Derived classes should override it.

Definition at line 427 of file Commandable.cc.

**6.41.3.21** `std::vector< std::string > artdaq::Commandable::legal_commands ( ) const`

Get the legal transition commands from the current state.

**Returns**

A list of legal transitions from the current state

Definition at line 313 of file Commandable.cc.

**6.41.3.22** `Commandable& artdaq::Commandable::operator= ( Commandable const & ) [delete]`

Copy Assignment operator is deleted.

**Returns**

[Commandable](#) copy

**6.41.3.23** `bool artdaq::Commandable::pause ( uint64_t timeout, uint64_t timestamp )`

Processes the pause transition.

## Parameters

<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

## Returns

Whether the transition was successful

Definition at line 123 of file Commandable.cc.

6.41.3.24 `virtual std::string artdaq::Commandable::register_monitor ( fhicl::ParameterSet const & ) [inline],[virtual]`

Perform the register\_monitor action.

## Returns

A report on the status of the command

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::DispatcherApp](#).

Definition at line 144 of file Commandable.hh.

6.41.3.25 `bool artdaq::Commandable::reinitialize ( fhicl::ParameterSet const & pset, uint64_t timeout, uint64_t timestamp )`

Processes the reinitialize request.

## Parameters

<i>pset</i>	ParameterSet used to configure the <a href="#">Commandable</a>
<i>timeout</i>	<a href="#">Timeout</a> for init step
<i>timestamp</i>	Timestamp of init step

## Returns

Whether the transition was successful

Definition at line 240 of file Commandable.cc.

6.41.3.26 `virtual std::string artdaq::Commandable::report ( std::string const & ) const [inline],[virtual]`

Default report implementation returns current report\_string.

## Returns

Current report\_string (may be set by derived classes)

Reimplemented in [artdaq::BoardReaderApp](#), [artdaq::RoutingManagerApp](#), [artdaq::EventBuilderApp](#), [artdaq::DispatcherApp](#), and [artdaq::DataLoggerApp](#).

Definition at line 126 of file Commandable.hh.

6.41.3.27 `bool artdaq::Commandable::resume ( uint64_t timeout, uint64_t timestamp )`

Processes the resume transition.

## Parameters

<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

## Returns

Whether the transition was successful

Definition at line 153 of file Commandable.cc.

#### 6.41.3.28 bool artdaq::Commandable::shutdown ( uint64\_t *timeout* )

Processes the shutdown transition.

## Parameters

<i>timeout</i>	<a href="#">Timeout</a> for transition
----------------	--

## Returns

Whether the transition was successful

Definition at line 182 of file Commandable.cc.

#### 6.41.3.29 bool artdaq::Commandable::soft\_initialize ( fhicl::ParameterSet const & *pset*, uint64\_t *timeout*, uint64\_t *timestamp* )

Processes the soft-initialize request.

## Parameters

<i>pset</i>	ParameterSet used to configure the <a href="#">Commandable</a>
<i>timeout</i>	<a href="#">Timeout</a> for init step
<i>timestamp</i>	Timestamp of init step

## Returns

Whether the transition was successful

Definition at line 210 of file Commandable.cc.

#### 6.41.3.30 bool artdaq::Commandable::start ( art::RunID *id*, uint64\_t *timeout*, uint64\_t *timestamp* )

Processes the start transition.

## Parameters

<i>id</i>	Run number of new run
<i>timeout</i>	<a href="#">Timeout</a> for transition

<i>timestamp</i>	Timestamp of transition
------------------	-------------------------

#### Returns

Whether the transition was successful

Definition at line 63 of file Commandable.cc.

#### 6.41.3.31 `std::string artdaq::Commandable::status ( ) const`

Returns the current state of the [Commandable](#).

#### Returns

The current state of the [Commandable](#)

Definition at line 300 of file Commandable.cc.

#### 6.41.3.32 `bool artdaq::Commandable::stop ( uint64_t timeout, uint64_t timestamp )`

Processes the stop transition.

#### Parameters

<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

#### Returns

Whether the transition was successful

Definition at line 93 of file Commandable.cc.

#### 6.41.3.33 `virtual std::string artdaq::Commandable::unregister_monitor ( std::string const & ) [inline], [virtual]`

Perform the unregister\_monitor action.

#### Returns

A report on the status of the command

This function is a No-Op. Derived classes should override it.

Reimplemented in [artdaq::DispatcherApp](#).

Definition at line 155 of file Commandable.hh.

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

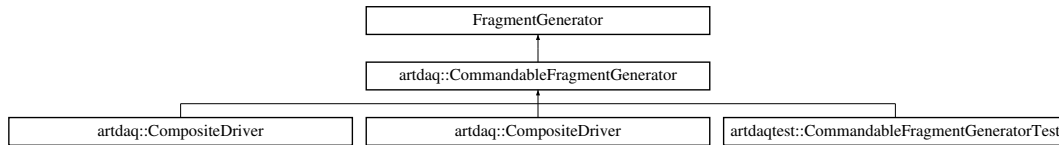
- `artdaq/artdaq/Application/Commandable.hh`
- `artdaq/artdaq/Application/Commandable.cc`

## 6.42 artdaq::CommandableFragmentGenerator Class Reference

[CommandableFragmentGenerator](#) is a [FragmentGenerator](#)-derived abstract class that defines the interface for a [FragmentGenerator](#) designed as a state machine with start, stop, etc., transition commands.

```
#include <artdaq/Generators/CommandableFragmentGenerator.hh>
```

Inheritance diagram for [artdaq::CommandableFragmentGenerator](#):



### Classes

- struct [Config](#)  
Configuration of the [CommandableFragmentGenerator](#). May be used for parameter validation

### Public Types

- using [Parameters](#) = [fhicl::WrappedTable](#)< [Config](#) >  
Used for [ParameterSet](#) validation (if desired)

### Public Member Functions

- [CommandableFragmentGenerator](#) (const [fhicl::ParameterSet](#) &ps)  
*CommandableFragmentGenerator Constructor.*
- virtual [~CommandableFragmentGenerator](#) ()  
*CommandableFragmentGenerator Destructor.*
- void [joinThreads](#) ()  
Join any data-taking threads. Should be called when destructing [CommandableFragmentGenerator](#).
- bool [getNext](#) ([FragmentPtrs](#) &output) overridefinal  
*getNext calls either applyRequests or getNext\_ to get any data that is ready to be sent to the EventBuilders*
- void [startMonitoringThread](#) ()  
Function that launches the monitoring thread ([getMonitoringDataLoop\(\)](#))
- void [getMonitoringDataLoop](#) ()  
This function regularly calls [checkHWStatus\\_\(\)](#), and sets the isHardwareOK flag accordingly.
- std::vector  
< [Fragment::fragment\\_id\\_t](#) > [fragmentIDs](#) () override  
Get the list of [Fragment](#) IDs handled by this [CommandableFragmentGenerator](#).
- size\_t [ev\\_counter](#) () const  
Get the current value of the event counter.
- void [StartCmd](#) (int run, uint64\_t timeout, uint64\_t timestamp)  
Start the [CommandableFragmentGenerator](#).
- void [StopCmd](#) (uint64\_t timeout, uint64\_t timestamp)  
Stop the [CommandableFragmentGenerator](#).

- void `PauseCmd` (uint64\_t `timeout`, uint64\_t `timestamp`)  
*Pause the `CommandableFragmentGenerator`.*
- void `ResumeCmd` (uint64\_t `timeout`, uint64\_t `timestamp`)  
*Resume the `CommandableFragmentGenerator`.*
- std::string `ReportCmd` (std::string const &`which`="")  
*Get a report about a user-specified run-time quantity.*
- virtual std::string `metricsReportingInstanceName` () const  
*Get the name used when reporting metrics.*
- bool `exception` () const  
*Get the current value of the exception flag.*
- virtual bool `metaCommand` (std::string const &`command`, std::string const &`arg`)  
*The meta-command is used for implementing user-specific commands in a `CommandableFragmentGenerator`.*
- void `SetRequestBuffer` (std::shared\_ptr< `RequestBuffer` > `buffer`)  
*Set the shared\_ptr to the `RequestBuffer`.*
- void `SetFragmentBuffer` (std::shared\_ptr< `FragmentBuffer` > `buffer`)

## Protected Member Functions

- int `run_number` () const  
*Get the current Run number.*
- int `subrun_number` () const  
*Get the current Subrun number.*
- uint64\_t `timeout` () const  
*Timeout of last command.*
- uint64\_t `timestamp` () const  
*Timestamp of last command.*
- artdaq::Fragment::fragment\_id\_t `fragment_id` () const  
*Get the Fragment ID of this Fragment generator.*
- bool `should_stop` () const  
*Get the current value of the should\_stop flag.*
- bool `check_stop` ()  
*Routine used by `applyRequests` to make sure that all outstanding requests have been fulfilled before returning.*
- size\_t `ev_counter_inc` (size\_t `step`=1)  
*Increment the event counter.*
- void `set_exception` (bool `exception`)  
*Control the exception flag.*
- void `metricsReportingInstanceName` (std::string const &`name`)  
*Sets the name for metrics reporting.*
- std::shared\_ptr< `RequestBuffer` > `GetRequestBuffer` ()  
*Get the shared\_ptr to the `RequestBuffer`.*
- std::shared\_ptr< `FragmentBuffer` > `GetFragmentBuffer` ()
- virtual bool `getNext_` (FragmentPtrs &`output`)=0  
*Obtain the next group of Fragments, if any are available. Return false if readout cannot continue, if we are 'stopped', or if we are not running in state-machine mode. Note that `getNext_()` must return n of each fragmentID declared by `fragmentIDs_()`.*
- virtual bool `checkHWStatus_` ()  
*Check any relevant hardware status registers. Return false if an error condition exists that should halt data-taking. This function should probably make `MetricManager` calls.*

- virtual void `start ()=0`  
*If a `CommandableFragmentGenerator` subclass is reading from a file, and `start()` is called, any run-, subrun-, and event-numbers in the data read from the file must be over-written by the specified run number, etc. After a call to `StartCmd()`, and until a call to `StopCmd()`, `getNext_()` is expected to return true as long as datataking is intended.*
- virtual void `stopNoMutex ()=0`  
*On call to `StopCmd`, `stopNoMutex()` is called prior to `StopCmd` acquiring the mutex*
- virtual void `stop ()=0`  
*If a `CommandableFragmentGenerator` subclass is reading from a file, calling `stop()` should arrange that the next call to `getNext_()` returns false, rather than allowing `getNext_()` to read to the end of the file.*
- virtual void `pauseNoMutex ()`  
*On call to `PauseCmd`, `pauseNoMutex()` is called prior to `PauseCmd` acquiring the mutex*
- virtual void `pause ()`  
*If a `CommandableFragmentGenerator` subclass is reading from hardware, the implementation of `pause()` should tell the hardware to stop sending data.*
- virtual void `resume ()`  
*The subrun number will be incremented before a call to resume.*
- virtual std::string `report ()`  
*Let's say that the contract with the `report()` functions is that they return a non-empty string if they have something useful to report, but if they don't know how to handle a given request, they simply return an empty string and the `ReportCmd()` takes care of saying "the xyz command is not currently supported". For backward compatibility, we keep the report function that takes no arguments and add one that takes a "which" argument. In the `ReportCmd` function, we'll call the more specific one first.*
- virtual std::string `reportSpecific (std::string const &what)`  
*Report the status of a specific quantity*

## Protected Attributes

- std::mutex `mutex_`  
*Mutex used to ensure that multiple transition commands do not run at the same time.*

### 6.42.1 Detailed Description

`CommandableFragmentGenerator` is a `FragmentGenerator`-derived abstract class that defines the interface for a `FragmentGenerator` designed as a state machine with start, stop, etc., transition commands.

Users of classes derived from `CommandableFragmentGenerator` will call these transitions via the publically defined `StartCmd()`, `StopCmd()`, etc.; these public functions contain functionality considered properly universal to all `CommandableFragmentGenerator`-derived classes, including calls to private virtual functions meant to be overridden in derived classes. The same applies to this class's implementation of the `FragmentGenerator::getNext()` pure virtual function, which is declared final (i.e., non-overridable in derived classes) and which itself calls a pure virtual `getNext_()` function to be implemented in derived classes.

State-machine related interface functions will be called only from a single thread. `getNext()` will be called only from a single thread. The thread from which state-machine interfaces functions are called may be a different thread from the one that calls `getNext()`.

John F., 3/24/14

After some discussion with Kurt, `CommandableFragmentGenerator` has been updated such that it now contains a member vector `fragment_ids_`; if "fragment\_id" is set in the FHiCL document controlling a class derived from `CommandableFragmentGenerator`, `fragment_ids_` will be booked as a length-1 vector, and the value in this vector will be returned by `fragment_id()`. `fragment_id()` will throw an exception if the length of the vector isn't 1. If "fragment\_ids" is set in the FHiCL document, then `fragment_ids_` is filled with the values in the list which "fragment\_ids" refers to, otherwise it is set to the

empty vector (this is what should happen if the user sets the "fragment\_id" variable in the FHiCL document, otherwise exceptions will end up thrown due to the logical conflict). If neither "fragment\_id" nor "fragment\_ids" is set in the FHiCL document, writers of classes derived from this one will be expected to override the virtual [fragmentIDs\(\)](#) function with their own code (the [CompositeDriver](#) class is an example of this)

Definition at line 83 of file CommandableFragmentGenerator.hh.

## 6.42.2 Constructor & Destructor Documentation

6.42.2.1 `artdaq::CommandableFragmentGenerator::CommandableFragmentGenerator ( const fhicl::ParameterSet & ps )`  
`[explicit]`

[CommandableFragmentGenerator](#) Constructor.

Parameters

<i>ps</i>	ParameterSet used to configure <a href="#">CommandableFragmentGenerator</a> . See <a href="#">artdaq::CommandableFragmentGenerator::Config</a> .
-----------	--

Definition at line 48 of file CommandableFragmentGenerator.cc.

6.42.2.2 `artdaq::CommandableFragmentGenerator::~~CommandableFragmentGenerator ( )` `[virtual]`

[CommandableFragmentGenerator](#) Destructor.

Joins all threads before returning

Definition at line 95 of file CommandableFragmentGenerator.cc.

## 6.42.3 Member Function Documentation

6.42.3.1 `bool artdaq::CommandableFragmentGenerator::check_stop ( )` `[protected]`

Routine used by applyRequests to make sure that all outstanding requests have been fulfilled before returning.

Returns

The logical AND of should\_stop, mode is not Ignored, and requests list size equal to 0

Definition at line 243 of file CommandableFragmentGenerator.cc.

6.42.3.2 `bool artdaq::CommandableFragmentGenerator::checkHWStatus_ ( )` `[protected]`, `[virtual]`

Check any relevant hardware status registers. Return false if an error condition exists that should halt data-taking. This function should probably make MetricManager calls.

Returns

False if a condition exists that should halt all data-taking, true otherwise

Reimplemented in [artdaqtest::CommandableFragmentGeneratorTest](#).

Definition at line 393 of file CommandableFragmentGenerator.cc.



6.42.3.3 `size_t artdaq::CommandableFragmentGenerator::ev_counter ( ) const` `[inline]`

Get the current value of the event counter.

#### Returns

The current value of the event counter

Definition at line 171 of file CommandableFragmentGenerator.hh.

6.42.3.4 `size_t artdaq::CommandableFragmentGenerator::ev_counter_inc ( size_t step = 1 )` `[protected]`

Increment the event counter.

#### Parameters

<i>step</i>	Amount to increment the event counter by
-------------	--

#### Returns

The previous value of the event counter

Definition at line 252 of file CommandableFragmentGenerator.cc.

6.42.3.5 `bool artdaq::CommandableFragmentGenerator::exception ( ) const` `[inline]`

Get the current value of the exception flag.

#### Returns

The current value of the exception flag

Definition at line 266 of file CommandableFragmentGenerator.hh.

6.42.3.6 `artdaq::Fragment::fragment_id_t artdaq::CommandableFragmentGenerator::fragment_id ( ) const` `[inline]`,  
`[protected]`

Get the Fragment ID of this Fragment generator.

#### Exceptions

<code>cet::exception("FragmentID")</code>	if there is more than one Fragment ID configured for this Fragment Generator
---	--

#### Returns

Fragment ID for the Fragment Generator

Definition at line 319 of file CommandableFragmentGenerator.hh.

6.42.3.7 `std::vector<Fragment::fragment_id_t> artdaq::CommandableFragmentGenerator::fragmentIDs ( )` `[inline]`,  
`[override]`

Get the list of Fragment IDs handled by this [CommandableFragmentGenerator](#).

**Returns**

A `std::vector<Fragment::fragment_id_t>` containing the Fragment IDs handled by this [CommandableFragmentGenerator](#)

Definition at line 155 of file `CommandableFragmentGenerator.hh`.

**6.42.3.8** `bool artdaq::CommandableFragmentGenerator::getNext ( FragmentPtrs & output )` `[final], [override]`

`getNext` calls either `applyRequests` or `getNext_` to get any data that is ready to be sent to the EventBuilders

**Parameters**

<i>output</i>	FragmentPtrs object containing Fragments ready for transmission
---------------	---

**Returns**

Whether `getNext` completed without exceptions

Definition at line 118 of file `CommandableFragmentGenerator.cc`.

**6.42.3.9** `virtual bool artdaq::CommandableFragmentGenerator::getNext_ ( FragmentPtrs & output )` `[protected], [pure virtual]`

Obtain the next group of Fragments, if any are available. Return false if readout cannot continue, if we are 'stopped', or if we are not running in state-machine mode. Note that `getNext_()` must return n of each fragmentID declared by `fragmentIDs_()`.

**Parameters**

<i>output</i>	Reference to list of Fragment pointers to which additional Fragments should be added
---------------	--

**Returns**

True if readout should continue, false otherwise

**6.42.3.10** `std::shared_ptr<RequestBuffer> artdaq::CommandableFragmentGenerator::GetRequestBuffer ( )` `[inline], [protected]`

Get the `shared_ptr` to the [RequestBuffer](#).

**Returns**

`shared_ptr` to the [RequestBuffer](#)

Definition at line 373 of file `CommandableFragmentGenerator.hh`.

**6.42.3.11** `void artdaq::CommandableFragmentGenerator::joinThreads ( )`

Join any data-taking threads. Should be called when destructing [CommandableFragmentGenerator](#).

Join any data-taking threads. Should be called when destructing [CommandableFragmentGenerator](#) Sets flags so that threads stop operations.

Definition at line 100 of file `CommandableFragmentGenerator.cc`.

6.42.3.12 `bool artdaq::CommandableFragmentGenerator::metaCommand ( std::string const & command, std::string const & arg )`  
`[virtual]`

The meta-command is used for implementing user-specific commands in a [CommandableFragmentGenerator](#).

## Parameters

<i>command</i>	Name of the command to run
<i>arg</i>	Argument(s) for command

## Returns

true if command succeeded or if command not supported

Definition at line 399 of file CommandableFragmentGenerator.cc.

6.42.3.13 `virtual std::string artdaq::CommandableFragmentGenerator::metricsReportingInstanceName ( ) const [inline], [virtual]`

Get the name used when reporting metrics.

## Returns

The name used when reporting metrics

Definition at line 241 of file CommandableFragmentGenerator.hh.

6.42.3.14 `void artdaq::CommandableFragmentGenerator::metricsReportingInstanceName ( std::string const & name ) [inline], [protected]`

Sets the name for metrics reporting.

## Parameters

<i>name</i>	The new name for metrics reporting
-------------	------------------------------------

Definition at line 354 of file CommandableFragmentGenerator.hh.

6.42.3.15 `void artdaq::CommandableFragmentGenerator::pause ( ) [protected], [virtual]`

If a [CommandableFragmentGenerator](#) subclass is reading from hardware, the implementation of [pause\(\)](#) should tell the hardware to stop sending data.

Reimplemented in [artdaqtest::CommandableFragmentGeneratorTest](#), [artdaq::CompositeDriver](#), and [artdaq::CompositeDriver](#).

Definition at line 372 of file CommandableFragmentGenerator.cc.

6.42.3.16 `void artdaq::CommandableFragmentGenerator::PauseCmd ( uint64_t timeout, uint64_t timestamp )`

Pause the [CommandableFragmentGenerator](#).

## Parameters

<i>timeout</i>	<a href="#">Timeout</a> for transition
----------------	--

<i>timestamp</i>	Timestamp of transition
------------------	-------------------------

A call to [PauseCmd\(\)](#) is advisory. It is an indication that the BoardReader should stop the incoming flow of data, if it can do so.

Definition at line 300 of file CommandableFragmentGenerator.cc.

6.42.3.17 `void artdaq::CommandableFragmentGenerator::pauseNoMutex ( ) [protected],[virtual]`

On call to PauseCmd, [pauseNoMutex\(\)](#) is called prior to PauseCmd acquiring the mutex

Definition at line 367 of file CommandableFragmentGenerator.cc.

6.42.3.18 `std::string artdaq::CommandableFragmentGenerator::report ( ) [protected],[virtual]`

Let's say that the contract with the [report\(\)](#) functions is that they return a non-empty string if they have something useful to report, but if they don't know how to handle a given request, they simply return an empty string and the [ReportCmd\(\)](#) takes care of saying "the xyz command is not currently supported". For backward compatibility, we keep the report function that takes no arguments and add one that takes a "which" argument. In the ReportCmd function, we'll call the more specific one first.

#### Returns

Default report from the FragmentGenerator

Definition at line 381 of file CommandableFragmentGenerator.cc.

6.42.3.19 `std::string artdaq::CommandableFragmentGenerator::ReportCmd ( std::string const & which = " " )`

Get a report about a user-specified run-time quantity.

#### Parameters

<i>which</i>	Which quantity to report
--------------	--------------------------

#### Returns

The report about the specified quantity

[CommandableFragmentGenerator](#) only implements "latest\_exception", a report on the last exception received. However, child classes can override the reportSpecific function to provide additional reports.

Definition at line 331 of file CommandableFragmentGenerator.cc.

6.42.3.20 `std::string artdaq::CommandableFragmentGenerator::reportSpecific ( std::string const & what ) [protected],[virtual]`

Report the status of a specific quantity

#### Parameters

<i>what</i>	Name of the quantity to report
-------------	--------------------------------

**Returns**

Value of requested quantity. Null string if what is unsupported.

Definition at line 387 of file CommandableFragmentGenerator.cc.

6.42.3.21 `void artdaq::CommandableFragmentGenerator::resume ( )` `[protected]`, `[virtual]`

The subrun number will be incremented *before* a call to resume.

Reimplemented in [artdaqtest::CommandableFragmentGeneratorTest](#), [artdaq::CompositeDriver](#), and [artdaq::CompositeDriver](#).

Definition at line 377 of file CommandableFragmentGenerator.cc.

6.42.3.22 `void artdaq::CommandableFragmentGenerator::ResumeCmd ( uint64_t timeout, uint64_t timestamp )`

Resume the [CommandableFragmentGenerator](#).

**Parameters**

<i>timeout</i>	Timeout for transition
<i>timestamp</i>	Timestamp of transition

After a call to [ResumeCmd\(\)](#), the next Fragments returned from [getNext\(\)](#) will be part of a new SubRun.

Definition at line 313 of file CommandableFragmentGenerator.cc.

6.42.3.23 `int artdaq::CommandableFragmentGenerator::run_number ( ) const` `[inline]`, `[protected]`

Get the current Run number.

**Returns**

The current Run number

Definition at line 297 of file CommandableFragmentGenerator.hh.

6.42.3.24 `void artdaq::CommandableFragmentGenerator::set_exception ( bool exception )` `[inline]`, `[protected]`

Control the exception flag.

**Parameters**

<i>exception</i>	Whether an exception has occurred
------------------	-----------------------------------

Definition at line 348 of file CommandableFragmentGenerator.hh.

6.42.3.25 `void artdaq::CommandableFragmentGenerator::SetRequestBuffer ( std::shared_ptr< RequestBuffer > buffer )`  
`[inline]`

Set the `shared_ptr` to the [RequestBuffer](#).

## Parameters

<i>buffer</i>	shared_ptr to the <a href="#">RequestBuffer</a>
---------------	---

Definition at line 280 of file CommandableFragmentGenerator.hh.

6.42.3.26 `bool artdaq::CommandableFragmentGenerator::should_stop ( ) const` `[inline], [protected]`

Get the current value of the should\_stop flag.

## Returns

The current value of the should\_stop flag

Definition at line 329 of file CommandableFragmentGenerator.hh.

6.42.3.27 `virtual void artdaq::CommandableFragmentGenerator::start ( )` `[protected], [pure virtual]`

If a [CommandableFragmentGenerator](#) subclass is reading from a file, and [start\(\)](#) is called, any run-, subrun-, and event-numbers in the data read from the file must be over-written by the specified run number, etc. After a call to [StartCmd\(\)](#), and until a call to [StopCmd\(\)](#), [getNext\\_\(\)](#) is expected to return true as long as data taking is intended.

This is a pure virtual function, and must be overridden by Fragment Generator implementations

Implemented in [artdaqtest::CommandableFragmentGeneratorTest](#), [artdaq::CompositeDriver](#), and [artdaq::CompositeDriver](#).

6.42.3.28 `void artdaq::CommandableFragmentGenerator::StartCmd ( int run, uint64_t timeout, uint64_t timestamp )`

Start the [CommandableFragmentGenerator](#).

## Parameters

<i>run</i>	Run ID of the new run
<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

After a call to 'StartCmd', all Fragments returned by [getNext\(\)](#) will be marked as part of a Run with the given run number, and with subrun number 1. Calling StartCmd also resets the event number to 1. After a call to [StartCmd\(\)](#), and until a call to [StopCmd](#), [getNext\(\)](#) – and hence the virtual function it calls, [getNext\\_\(\)](#) – should return true as long as data taking is meant to take place, even if a particular call returns no fragments.

Definition at line 258 of file CommandableFragmentGenerator.cc.

6.42.3.29 `virtual void artdaq::CommandableFragmentGenerator::stop ( )` `[protected], [pure virtual]`

If a [CommandableFragmentGenerator](#) subclass is reading from a file, calling [stop\(\)](#) should arrange that the next call to [getNext\\_\(\)](#) returns false, rather than allowing [getNext\\_\(\)](#) to read to the end of the file.

This is a pure virtual function, and must be overridden by Fragment Generator implementations

Implemented in [artdaqtest::CommandableFragmentGeneratorTest](#), [artdaq::CompositeDriver](#), and [artdaq::CompositeDriver](#).

6.42.3.30 void artdaq::CommandableFragmentGenerator::StopCmd ( uint64\_t *timeout*, uint64\_t *timestamp* )

Stop the [CommandableFragmentGenerator](#).



## Parameters

<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

After a call to [StopCmd\(\)](#), [getNext\(\)](#) will eventually return false. This may not happen for several calls, if the implementation has data to be 'drained' from the system.

Definition at line 284 of file CommandableFragmentGenerator.cc.

6.42.3.31 `virtual void artdaq::CommandableFragmentGenerator::stopNoMutex ( ) [protected], [pure virtual]`

On call to StopCmd, [stopNoMutex\(\)](#) is called prior to StopCmd acquiring the mutex

This is a pure virtual function, and must be overridden by Fragment Generator implementations

Implemented in [artdaqtest::CommandableFragmentGeneratorTest](#), [artdaq::CompositeDriver](#), and [artdaq::CompositeDriver](#).

6.42.3.32 `int artdaq::CommandableFragmentGenerator::subrun_number ( ) const [inline], [protected]`

Get the current Subrun number.

## Returns

The current Subrun number

Definition at line 302 of file CommandableFragmentGenerator.hh.

6.42.3.33 `uint64_t artdaq::CommandableFragmentGenerator::timeout ( ) const [inline], [protected]`

[Timeout](#) of last command.

## Returns

[Timeout](#) of last command

Definition at line 307 of file CommandableFragmentGenerator.hh.

6.42.3.34 `uint64_t artdaq::CommandableFragmentGenerator::timestamp ( ) const [inline], [protected]`

Timestamp of last command.

## Returns

Timestamp of last command

Definition at line 312 of file CommandableFragmentGenerator.hh.

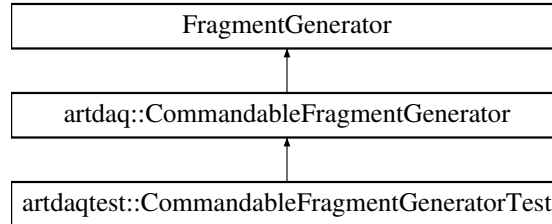
The documentation for this class was generated from the following files:

- `artdaq/artdaq/Generators/CommandableFragmentGenerator.hh`
- `artdaq/artdaq/Generators/CommandableFragmentGenerator.cc`

## 6.43 artdaqtest::CommandableFragmentGeneratorTest Class Reference

CommandableFragmentGenerator derived class for testing.

Inheritance diagram for artdaqtest::CommandableFragmentGeneratorTest:



### Public Member Functions

- [CommandableFragmentGeneratorTest](#) (const fhicl::ParameterSet &ps)  
*CommandableFragmentGeneratorTest Constructor.*
- void [setFireCount](#) (size\_t count)  
*Have getNext\_ generate count fragments.*
- void [setHwFail](#) ()  
*Set the hwFail flag.*
- void [setEnabledIds](#) (uint64\_t bitmask)  
*Set the enabled IDs mask for the Fragment Generator.*
- void [setTimestamp](#) (artdaq::Fragment::timestamp\_t ts)  
*Set the timestamp to be used for the next Fragment.*
- artdaq::Fragment::timestamp\_t [getTimestamp](#) ()  
*Get the timestamp that will be used for the next Fragment.*

### Protected Member Functions

- bool [getNext\\_](#) (artdaq::FragmentPtrs &frags) override  
*Generate data and return it to CommandableFragmentGenerator.*
- bool [checkHWStatus\\_](#) () override  
*Returns whether the hwFail flag has not been set.*
- void [start](#) () override  
*Perform start actions. No-Op.*
- void [stopNoMutex](#) () override  
*Perform immediate stop actions. No-Op.*
- void [stop](#) () override  
*Perform stop actions. No-Op.*
- void [pause](#) () override  
*Perform pause actions. No-Op.*
- void [resume](#) () override  
*Perform resume actions. No-Op.*

## Additional Inherited Members

### 6.43.1 Detailed Description

CommandableFragmentGenerator derived class for testing.

Definition at line 48 of file CommandableFragmentGenerator\_t.cc.

### 6.43.2 Member Function Documentation

**6.43.2.1** `bool artdaqtest::CommandableFragmentGeneratorTest::checkHWStatus_( ) [inline], [override], [protected], [virtual]`

Returns whether the hwFail flag has not been set.

#### Returns

If hwFail has been set, false, otherwise true

Reimplemented from [artdaq::CommandableFragmentGenerator](#).

Definition at line 78 of file CommandableFragmentGenerator\_t.cc.

**6.43.2.2** `bool artdaqtest::CommandableFragmentGeneratorTest::getNext_( artdaq::FragmentPtrs & frags ) [override], [protected]`

Generate data and return it to CommandableFragmentGenerator.

#### Parameters

<i>frags</i>	FragmentPtrs list that new Fragments should be added to
--------------	---

#### Returns

True if data was generated

[CommandableFragmentGeneratorTest](#) merely default-constructs Fragments, emplacing them on the frags list.

Definition at line 158 of file CommandableFragmentGenerator\_t.cc.

**6.43.2.3** `artdaq::Fragment::timestamp_t artdaqtest::CommandableFragmentGeneratorTest::getTimestamp( ) [inline]`

Get the timestamp that will be used for the next Fragment.

#### Returns

The timestamp that will be used for the next Fragment

Definition at line 136 of file CommandableFragmentGenerator\_t.cc.

**6.43.2.4** `void artdaqtest::CommandableFragmentGeneratorTest::setEnabledIds( uint64_t bitmask ) [inline]`

Set the enabled IDs mask for the Fragment Generator.

## Parameters

<i>bitmask</i>	Bitmask of enabled IDs for the Fragment Generator
----------------	---

For testing, this bitmask allows a configured Fragment ID to not be generated by a given call to `setFireCount`. This is used to create asymmetric response from the Fragment generator in the `_MultipleIDs` test cases

Definition at line 124 of file `CommandableFragmentGenerator_t.cc`.

6.43.2.5 `void artdaqtest::CommandableFragmentGeneratorTest::setFireCount ( size_t count ) [inline]`

Have `getNext_` generate count fragments.

## Parameters

<i>count</i>	Number of fragments to generate
--------------	---------------------------------

Definition at line 110 of file `CommandableFragmentGenerator_t.cc`.

6.43.2.6 `void artdaqtest::CommandableFragmentGeneratorTest::setTimestamp ( artdaq::Fragment::timestamp_t ts ) [inline]`

Set the timestamp to be used for the next Fragment.

## Parameters

<i>ts</i>	Timestamp to be used for the next Fragment
-----------	--

Definition at line 130 of file `CommandableFragmentGenerator_t.cc`.

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

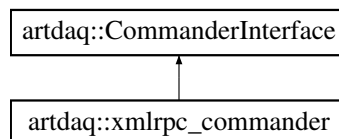
- `artdaq/test/Generators/CommandableFragmentGenerator_t.cc`

## 6.44 artdaq::CommanderInterface Class Reference

This interface defines the functions used to transfer data between artdaq applications.

```
#include <artdaq/ExternalComms/CommanderInterface.hh>
```

Inheritance diagram for `artdaq::CommanderInterface`:



## Classes

- struct [Config](#)

Configuration of the [CommanderInterface](#). May be used for parameter validation

## Public Types

- using [Parameters](#) = fhicl::WrappedTable< [Config](#) >  
*Used for ParameterSet validation (if desired)*

## Public Member Functions

- [CommanderInterface](#) (const fhicl::ParameterSet &ps, [artdaq::Commandable](#) &commandable)  
*CommanderInterface Constructor.*
- [CommanderInterface](#) (const [CommanderInterface](#) &)=delete  
*Copy Constructor is deleted.*
- [CommanderInterface](#) & operator= (const [CommanderInterface](#) &)=delete  
*Copy Assignment operator is deleted.*
- virtual [~CommanderInterface](#) ()  
*Default virtual Destructor.*
- virtual void [run\\_server](#) ()=0  
*run\_server is the main work loop for the Commander.*
- virtual std::string [send\\_init](#) (fhicl::ParameterSet const &ps, uint64\_t timeout, uint64\_t timestamp)  
*Using the transport mechanism, send an init command*
- virtual std::string [send\\_soft\\_init](#) (fhicl::ParameterSet const &ps, uint64\_t timeout, uint64\_t timestamp)  
*Using the transport mechanism, send a soft\_init command*
- virtual std::string [send\\_reinit](#) (fhicl::ParameterSet const &ps, uint64\_t timeout, uint64\_t timestamp)  
*Using the transport mechanism, send a reinit command*
- virtual std::string [send\\_start](#) (art::RunID runNumber, uint64\_t timeout, uint64\_t timestamp)  
*Using the transport mechanism, send a start command*
- virtual std::string [send\\_pause](#) (uint64\_t timeout, uint64\_t timestamp)  
*Using the transport mechanism, send a pause command*
- virtual std::string [send\\_resume](#) (uint64\_t timeout, uint64\_t timestamp)  
*Using the transport mechanism, send a resume command*
- virtual std::string [send\\_stop](#) (uint64\_t timeout, uint64\_t timestamp)  
*Using the transport mechanism, send a stop command*
- virtual std::string [send\\_shutdown](#) (uint64\_t timeout)  
*Using the transport mechanism, send a shutdown command*
- virtual std::string [send\\_status](#) ()  
*Using the transport mechanism, send a status command*
- virtual std::string [send\\_report](#) (std::string const &which)  
*Using the transport mechanism, send a report command*
- virtual std::string [send\\_legal\\_commands](#) ()  
*Using the transport mechanism, send a legal\_commands command*
- virtual std::string [send\\_register\\_monitor](#) (std::string const &monitor\_fhicl)  
*Using the transport mechanism, send a register\_monitor command*
- virtual std::string [send\\_unregister\\_monitor](#) (std::string const &label)  
*Using the transport mechanism, send an unregister\_monitor command*
- virtual std::string [send\\_trace\\_get](#) (std::string const &name)  
*Using the transport mechanism, send an send\_trace\_get command*
- virtual std::string [send\\_trace\\_set](#) (std::string const &name, std::string const &type, std::string const &mask)  
*Using the transport mechanism, send an send\_trace\_msgfacility\_set command*

- virtual std::string [send\\_meta\\_command](#) (std::string const &command, std::string const &argument)  
*Using the transport mechanism, send an send\_meta\_command command*
- virtual std::string [send\\_rollover\\_subrun](#) (uint64\_t seq, uint32\_t subrunNumber)  
*Using the transport mechanism, send a send\_rollover\_subrun command*
- bool [GetStatus](#) ()  
*Determine whether the Commander plugin is ready to accept commands*
- virtual std::string [add\\_config\\_archive\\_entry](#) (std::string const &key, std::string const &value)  
*Using the transport mechanism, send an add\_config\_archive\_entry command*
- virtual std::string [clear\\_config\\_archive](#) ()  
*Using the transport mechanism, send a clear\_config\_archive command*

## Public Attributes

- [artdaq::Commandable](#) & [\\_commandable](#)  
*Reference to the [Commandable](#) that this Commander Commands.*

## Protected Attributes

- int [\\_id](#)  
*ID Number of this Commander.*
- std::atomic< bool > [running\\_](#)  
*Whether the server is running and able to respond to requests.*

### 6.44.1 Detailed Description

This interface defines the functions used to transfer data between artdaq applications.

Definition at line 22 of file CommanderInterface.hh.

### 6.44.2 Constructor & Destructor Documentation

- 6.44.2.1 [artdaq::CommanderInterface::CommanderInterface](#) ( const fhicl::ParameterSet & *ps*, [artdaq::Commandable](#) & *commandable* ) [\[inline\]](#)

[CommanderInterface](#) Constructor.

Parameters

<i>ps</i>	ParameterSet used for configuring the <a href="#">CommanderInterface</a> . See <a href="#">artdaq::CommanderInterface::Config</a>
<i>commandable</i>	<a href="#">artdaq::Commandable</a> object to send transition commands to

Definition at line 43 of file CommanderInterface.hh.

### 6.44.3 Member Function Documentation

- 6.44.3.1 [std::string artdaq::CommanderInterface::add\\_config\\_archive\\_entry](#) ( std::string const & *key*, std::string const & *value* )  
[\[inline\]](#), [\[virtual\]](#)

Using the transport mechanism, send an add\_config\_archive\_entry command

This will cause the receiver to add the specified key and value to its list of configuration archive information, which is stored in the art/ROOT files that are written by various processes in the system. This command accepts configuration key and value strings.

EXAMPLE: xmlrpc <http://localhost:5235/RPC2> daq.add\_config\_archive\_entry "EventBuilder1" "daq:{verbose: true}"

#### Parameters

<i>key</i>	Key in the archive
<i>value</i>	Value to store in the archive

#### Returns

Command result: "SUCCESS" if succeeded

Definition at line 108 of file CommanderInterface.cc.

#### 6.44.3.2 std::string artdaq::CommanderInterface::clear\_config\_archive ( ) [inline],[virtual]

Using the transport mechanism, send a clear\_config\_archive command

This will cause the receiver to clear its list of configuration archive information.

EXAMPLE: xmlrpc <http://localhost:5235/RPC2> daq.clear\_config\_archive

#### Returns

Command result: "SUCCESS" if succeeded

Definition at line 114 of file CommanderInterface.cc.

#### 6.44.3.3 bool artdaq::CommanderInterface::GetStatus ( ) [inline]

Determine whether the Commander plugin is ready to accept commands

#### Returns

True if running, false otherwise

Definition at line 279 of file CommanderInterface.hh.

#### 6.44.3.4 CommanderInterface& artdaq::CommanderInterface::operator= ( const CommanderInterface & ) [delete]

Copy Assignment operator is deleted.

#### Returns

[CommanderInterface](#) Copy

#### 6.44.3.5 `virtual void artdaq::CommanderInterface::run_server ( ) [pure virtual]`

`run_server` is the main work loop for the Commander.

This function is expected to block and persist for the entire run of the application. It should accept and handle the following commands (subject to state-machine constraints, see [Commandable::legal\\_commands\(\)](#)): `init` `soft_init` `reinit` `start` `pause` `resume` `stop` `shutdown` `status` `report` `legal_commands` `register_monitor` `unregister_monitor` `trace_get` `trace_set` `meta_command` `rollover_subrun` `add_config_archive_entry` `clear_config_archive`

See the `send_*` functions for more details on each command. Not all commands are valid for all applications/states. `run_server` should return a string indicating success or failure to the transport mechanism when it is done processing a command.

Implemented in [artdaq::xmlrpc\\_commander](#).

#### 6.44.3.6 `std::string artdaq::CommanderInterface::send_init ( fhicl::ParameterSet const & ps, uint64_t timeout, uint64_t timestamp ) [inline],[virtual]`

Using the transport mechanism, send an `init` command

The `init` command is accepted by all artdaq processes that are in the booted state. It expects a `ParameterSet` for configuration, a timeout, and a timestamp.

Parameters

<i>ps</i>	ParameterSet for the command
<i>timeout</i>	<a href="#">Timeout</a> for the command
<i>timestamp</i>	Timestamp of the command

Returns

Command result: "SUCCESS" if succeeded

Reimplemented in [artdaq::xmlrpc\\_commander](#).

Definition at line 6 of file `CommanderInterface.cc`.

#### 6.44.3.7 `std::string artdaq::CommanderInterface::send_legal_commands ( ) [inline],[virtual]`

Using the transport mechanism, send a `legal_commands` command

This will query the artdaq process, and it will return the list of allowed transition commands from its current state.

Returns

Command result: a list of allowed transition commands from its current state

Reimplemented in [artdaq::xmlrpc\\_commander](#).

Definition at line 66 of file `CommanderInterface.cc`.

#### 6.44.3.8 `std::string artdaq::CommanderInterface::send_meta_command ( std::string const & command, std::string const & argument ) [inline],[virtual]`

Using the transport mechanism, send an `send_meta_command` command

This will cause the receiver to run the given command with the given argument in user code



## Parameters

<i>command</i>	Command name to send
<i>argument</i>	Argument for command

## Returns

Command result: "SUCCESS" if succeeded

Reimplemented in [artdaq::xmlrpc\\_commander](#).

Definition at line 96 of file CommanderInterface.cc.

6.44.3.9 `std::string artdaq::CommanderInterface::send_pause ( uint64_t timeout, uint64_t timestamp ) [inline], [virtual]`

Using the transport mechanism, send a pause command

The pause command pauses a Run. When the run resumes, the subrun number will be incremented. This command accepts a timeout parameter and a timestamp parameter.

## Parameters

<i>timeout</i>	<a href="#">Timeout</a> for the command
<i>timestamp</i>	Timestamp of the command

## Returns

Command result: "SUCCESS" if succeeded

Reimplemented in [artdaq::xmlrpc\\_commander](#).

Definition at line 30 of file CommanderInterface.cc.

6.44.3.10 `std::string artdaq::CommanderInterface::send_register_monitor ( std::string const & monitor_fhicl ) [inline], [virtual]`

Using the transport mechanism, send a register\_monitor command

This will cause a Dispatcher to start an art process with the given FHiCL configuration string

## Parameters

<i>monitor_fhicl</i>	FHiCL code used to configure the art process that the Dispatcher starts
----------------------	---

## Returns

Command result: "SUCCESS" if succeeded

Reimplemented in [artdaq::xmlrpc\\_commander](#).

Definition at line 72 of file CommanderInterface.cc.

6.44.3.11 `std::string artdaq::CommanderInterface::send_reinit ( fhicl::ParameterSet const & ps, uint64_t timeout, uint64_t timestamp ) [inline], [virtual]`

Using the transport mechanism, send a reinit command

The reinit command is accepted by all artdaq processes. It expects a ParameterSet for configuration, a timeout, and a timestamp.

## Parameters

<i>ps</i>	ParameterSet for the command
<i>timeout</i>	<a href="#">Timeout</a> for the command
<i>timestamp</i>	Timestamp of the command

## Returns

Command result: "SUCCESS" if succeeded

Reimplemented in [artdaq::xmlrpc\\_commander](#).

Definition at line 18 of file CommanderInterface.cc.

6.44.3.12 `std::string artdaq::CommanderInterface::send_report ( std::string const & which ) [inline], [virtual]`

Using the transport mechanism, send a report command

The report command returns the current value of the requested reportable quantity.

## Parameters

<i>which</i>	Reportable quantity to request
--------------	--------------------------------

## Returns

Command result: current value of the requested reportable quantity

Reimplemented in [artdaq::xmlrpc\\_commander](#).

Definition at line 60 of file CommanderInterface.cc.

6.44.3.13 `std::string artdaq::CommanderInterface::send_resume ( uint64_t timeout, uint64_t timestamp ) [inline], [virtual]`

Using the transport mechanism, send a resume command

The resume command resumes a paused Run. When the run resumes, the subrun number will be incremented. This command accepts a timeout parameter and a timestamp parameter.

## Parameters

<i>timeout</i>	<a href="#">Timeout</a> for the command
<i>timestamp</i>	Timestamp of the command

## Returns

Command result: "SUCCESS" if succeeded

Reimplemented in [artdaq::xmlrpc\\_commander](#).

Definition at line 36 of file CommanderInterface.cc.

6.44.3.14 `std::string artdaq::CommanderInterface::send_rollover_subrun ( uint64_t seq, uint32_t subrunNumber ) [inline], [virtual]`

Using the transport mechanism, send a send\_rollover\_subrun command

This will cause the receiver to rollover the subrun number at the given event. (Event with seqID == boundary will be in new subrun.) Should be sent to all EventBuilders before the given event is processed.

## Parameters

<i>seq</i>	Sequence ID of new subrun
<i>subrunNumber</i>	Subrun number of the new subrun

## Returns

Command result: "SUCCESS" if succeeded

Reimplemented in [artdaq::xmlrpc\\_commander](#).

Definition at line 102 of file CommanderInterface.cc.

6.44.3.15 `std::string artdaq::CommanderInterface::send_shutdown ( uint64_t timeout ) [inline],[virtual]`

Using the transport mechanism, send a shutdown command

The shutdown command shuts down the artdaq process. This command accepts a timeout parameter.

## Parameters

<i>timeout</i>	<a href="#">Timeout</a> for the command
----------------	---

## Returns

Command result: "SUCCESS" if succeeded

Reimplemented in [artdaq::xmlrpc\\_commander](#).

Definition at line 48 of file CommanderInterface.cc.

6.44.3.16 `std::string artdaq::CommanderInterface::send_soft_init ( fhicl::ParameterSet const & ps, uint64_t timeout, uint64_t timestamp ) [inline],[virtual]`

Using the transport mechanism, send a soft\_init command

The soft\_init command is accepted by all artdaq processes that are in the booted state. It expects a ParameterSet for configuration, a timeout, and a timestamp.

## Parameters

<i>ps</i>	ParameterSet for the command
<i>timeout</i>	<a href="#">Timeout</a> for the command
<i>timestamp</i>	Timestamp of the command

## Returns

Command result: "SUCCESS" if succeeded

Reimplemented in [artdaq::xmlrpc\\_commander](#).

Definition at line 12 of file CommanderInterface.cc.

6.44.3.17 `std::string artdaq::CommanderInterface::send_start ( art::RunID runNumber, uint64_t timeout, uint64_t timestamp ) [inline],[virtual]`

Using the transport mechanism, send a start command

The start command starts a Run using the given run number. This command also accepts a timeout parameter and a timestamp parameter.

## Parameters

<i>runNumber</i>	Run number of the new run
<i>timeout</i>	<a href="#">Timeout</a> for the command
<i>timestamp</i>	Timestamp of the command

## Returns

Command result: "SUCCESS" if succeeded

Reimplemented in [artdaq::xmlrpc\\_commander](#).

Definition at line 24 of file CommanderInterface.cc.

6.44.3.18 `std::string artdaq::CommanderInterface::send_status ( ) [inline],[virtual]`

Using the transport mechanism, send a status command

The status command returns the current status of the artdaq process.

## Returns

Command result: current status of the artdaq process

Reimplemented in [artdaq::xmlrpc\\_commander](#).

Definition at line 54 of file CommanderInterface.cc.

6.44.3.19 `std::string artdaq::CommanderInterface::send_stop ( uint64_t timeout, uint64_t timestamp ) [inline],[virtual]`

Using the transport mechanism, send a stop command

The stop command stops the current Run. This command accepts a timeout parameter and a timestamp parameter.

## Parameters

<i>timeout</i>	<a href="#">Timeout</a> for the command
<i>timestamp</i>	Timestamp of the command

## Returns

Command result: "SUCCESS" if succeeded

Reimplemented in [artdaq::xmlrpc\\_commander](#).

Definition at line 42 of file CommanderInterface.cc.

6.44.3.20 `std::string artdaq::CommanderInterface::send_trace_get ( std::string const & name ) [inline],[virtual]`

Using the transport mechanism, send an send\_trace\_get command

This will cause the receiver to get the TRACE level masks for the given name Use name == "ALL" to get ALL names

## Parameters

<i>name</i>	TRACE name to get the mask for ("ALL" to get all names)
-------------	---

## Returns

Command result: "SUCCESS" if succeeded

Reimplemented in [artdaq::xmlrpc\\_commander](#).

Definition at line 84 of file CommanderInterface.cc.

6.44.3.21 `std::string artdaq::CommanderInterface::send_trace_set ( std::string const & name, std::string const & type, std::string const & mask ) [inline], [virtual]`

Using the transport mechanism, send an send\_trace\_msgfacility\_set command

This will cause the receiver to set the given TRACE level mask for the given name to the given mask. Only the first character of the mask selection will be parsed, dial 'M' for Memory, or 'S' for Slow. Use name == "ALL" to set ALL names

EXAMPLE: xmlrpc <http://localhost:5235/RPC2> daq.trace\_set s/M s/ALL s/0x12345

## Parameters

<i>name</i>	TRACE name to set ("ALL" for all TRACE names)
<i>type</i>	Type of mask to set ('M', 'S', or 'T')
<i>mask</i>	64-bit mask, in string form

## Returns

Command result: "SUCCESS" if succeeded

Reimplemented in [artdaq::xmlrpc\\_commander](#).

Definition at line 90 of file CommanderInterface.cc.

6.44.3.22 `std::string artdaq::CommanderInterface::send_unregister_monitor ( std::string const & label ) [inline], [virtual]`

Using the transport mechanism, send an unregister\_monitor command

This will cause a Dispatcher to stop sending data to the monitor identified by the given label

## Parameters

<i>label</i>	Label of the monitor to unregister
--------------	------------------------------------

## Returns

Command result: "SUCCESS" if succeeded

Reimplemented in [artdaq::xmlrpc\\_commander](#).

Definition at line 78 of file CommanderInterface.cc.



### 6.44.4 Member Data Documentation

#### 6.44.4.1 artdaq::Commandable& artdaq::CommanderInterface::\_commandable

Reference to the [Commandable](#) that this Commander Commands.

Definition at line 314 of file CommanderInterface.hh.

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

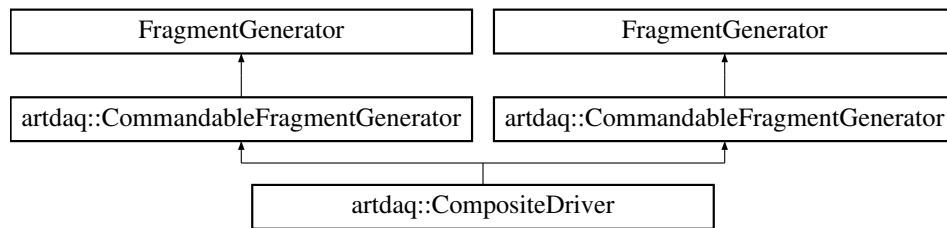
- artdaq/artdaq/ExternalComms/CommanderInterface.hh
- artdaq/artdaq/ExternalComms/CommanderInterface.cc

## 6.45 artdaq::CompositeDriver Class Reference

[CompositeDriver](#) handles a set of lower-level generators.

```
#include <artdaq/Generators/CompositeDriver.hh>
```

Inheritance diagram for artdaq::CompositeDriver:



### Public Member Functions

- [CompositeDriver](#) (fhicl::ParameterSet const &ps)  
*CompositeDriver Constructor.*
- virtual [~CompositeDriver](#) () noexcept  
*Destructor. Calls the destructors for each configured [CommandableFragmentGenerator](#).*
- void [start](#) () override  
*Start all configured CommandableFragmentGenerators.*
- void [stopNoMutex](#) () override  
*Call non-locked stop methods for all configured CommandableFragmentGenerators.*
- void [stop](#) () override  
*Call stop methods for all configured CommandableFragmentGenerators. Currently handled by stopNoMutex.*
- void [pause](#) () override  
*Pause all configured CommandableFragmentGenerators.*
- void [resume](#) () override  
*Resume all configured CommandableFragmentGenerators.*
- [CompositeDriver](#) (fhicl::ParameterSet const &ps)  
*CompositeDriver Constructor.*
- virtual [~CompositeDriver](#) () noexcept  
*Destructor. Calls the destructors for each configured [CommandableFragmentGenerator](#).*

- void [start](#) () override  
*Start all configured CommandableFragmentGenerators.*
- void [stopNoMutex](#) () override  
*Call non-locked stop methods for all configured CommandableFragmentGenerators.*
- void [stop](#) () override  
*Call stop methods for all configured CommandableFragmentGenerators. Currently handled by stopNoMutex.*
- void [pause](#) () override  
*Pause all configured CommandableFragmentGenerators.*
- void [resume](#) () override  
*Resume all configured CommandableFragmentGenerators.*

## Additional Inherited Members

### 6.45.1 Detailed Description

[CompositeDriver](#) handles a set of lower-level generators.

When multiple CommandableFragmentGenerators are needed by a single BoardReader, [CompositeDriver](#) may be used to provide a single interface to all of them.

Definition at line 18 of file CompositeDriver.hh.

### 6.45.2 Constructor & Destructor Documentation

#### 6.45.2.1 `artdaq::CompositeDriver::CompositeDriver ( fhicl::ParameterSet const & ps )` `[explicit]`

[CompositeDriver](#) Constructor.

Parameters

<i>ps</i>	ParameterSet used to configure <a href="#">CompositeDriver</a>
-----------	--

```
* CompositeDriver accepts the following Parameters:
* "generator_config_list" (REQUIRED): A FHiCL sequence of FHiCL tables, each one configuring
* a CommandableFragmentGenerator instance.
*
```

Definition at line 79 of file CompositeDriver\_generator.cc.

#### 6.45.2.2 `artdaq::CompositeDriver::CompositeDriver ( fhicl::ParameterSet const & ps )` `[explicit]`

[CompositeDriver](#) Constructor.

Parameters

<i>ps</i>	ParameterSet used to configure <a href="#">CompositeDriver</a>
-----------	--

```
* CompositeDriver accepts the following Parameters:
* "generator_config_list" (REQUIRED): A FHiCL sequence of FHiCL tables, each one configuring
* a CommandableFragmentGenerator instance.
*
```

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

- artdaq/artdaq/Generators/CompositeDriver.hh
- artdaq/artdaq/Generators/CompositeDriver\_generator.cc

## 6.46 artdaq::CommandableFragmentGenerator::Config Struct Reference

Configuration of the [CommandableFragmentGenerator](#). May be used for parameter validation

```
#include <artdaq/Generators/CommandableFragmentGenerator.hh>
```

### Public Attributes

- fhicl::Atom< std::string > [generator\\_type](#) {fhicl::Name{"generator"}, fhicl::Comment{"Name of the [Commandable-FragmentGenerator](#) plugin to load"}}  
*"generator" (REQUIRED) Name of the [CommandableFragmentGenerator](#) plugin to load*
- fhicl::Atom< Fragment::type\_t > [expected\\_fragment\\_type](#) {fhicl::Name{"expected\_fragment\_type"}, fhicl::Comment{"The type of Fragments this CFG will be generating. \"Empty\" will auto-detect type based on Fragments generated."}, Fragment::type\_t(Fragment::EmptyFragmentType)}  
*"expected\_fragment\_type" (Default: 231, EmptyFragmentType) : The type of Fragments this CFG will be generating. "Empty" will auto - detect type based on Fragments generated.*
- fhicl::Atom< size\_t > [sleep\\_on\\_no\\_data\\_us](#) {fhicl::Name{"sleep\_on\_no\_data\_us"}, fhicl::Comment{"How long to sleep after calling getNext\_ if no data is returned"}, 0}  
*"sleep\_on\_no\_data\_us" (Default: 0 (no sleep)) : How long to sleep after calling getNext\_ if no data is returned*
- fhicl::Atom< bool > [separate\\_monitoring\\_thread](#) {fhicl::Name{"separate\_monitoring\_thread"}, fhicl::Comment{"Whether a thread that calls the [checkHWStatus\\_](#) method should be created"}, false}  
*"separate\_monitoring\_thread" (Default: false) : Whether a thread that calls the checkHWStatus\_ method should be created*
- fhicl::Atom< int64\_t > [hardware\\_poll\\_interval\\_us](#) {fhicl::Name{"hardware\_poll\_interval\_us"}, fhicl::Comment{"If a separate monitoring thread is used, how often should it call [checkHWStatus\\_](#)"}, 0}  
*"hardware\_poll\_interval\_us" (Default: 0) : If a separate monitoring thread is used, how often should it call checkHWStatus\_*
- fhicl::Sequence  
 < Fragment::fragment\_id\_t > [fragment\\_ids](#) {fhicl::Name{"fragment\_ids"}, fhicl::Comment{"A list of Fragment IDs created by this [CommandableFragmentGenerator](#)"}}  
- fhicl::Atom< int > [fragment\\_id](#) {fhicl::Name{"fragment\_id"}, fhicl::Comment{"The Fragment ID created by this [CommandableFragmentGenerator](#)"}, -99}
- fhicl::Atom< int > [sleep\\_on\\_stop\\_us](#) {fhicl::Name{"sleep\_on\_stop\_us"}, fhicl::Comment{"How long to sleep before returning when [stop](#) transition is called"}, 0}  
*"sleep\_on\_stop\_us" (Default: 0) : How long to sleep before returning when stop transition is called*

### 6.46.1 Detailed Description

Configuration of the [CommandableFragmentGenerator](#). May be used for parameter validation

Definition at line 89 of file CommandableFragmentGenerator.hh.

### 6.46.2 Member Data Documentation

6.46.2.1 fhicl::Atom<int> artdaq::CommandableFragmentGenerator::Config::fragment\_id {fhicl::Name{"fragment\_id"}, fhicl::Comment{"The Fragment ID created by this CommandableFragmentGenerator"}, -99}

"fragment\_id" (Default: -99) : The Fragment ID created by this [CommandableFragmentGenerator](#) Note that only one of fragment\_ids and fragment\_id should be specified in the configuration

Definition at line 106 of file CommandableFragmentGenerator.hh.

6.46.2.2 fhicl::Sequence<Fragment::fragment\_id\_t> artdaq::CommandableFragmentGenerator::Config::fragment\_ids {fhicl::Name{"fragment\_ids"}, fhicl::Comment{"A list of Fragment IDs created by this CommandableFragmentGenerator"}}

"fragment\_ids" (Default: empty vector) : A list of Fragment IDs created by this [CommandableFragmentGenerator](#) Note that only one of fragment\_ids and fragment\_id should be specified in the configuration

Definition at line 103 of file CommandableFragmentGenerator.hh.

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

- artdaq/artdaq/Generators/CommandableFragmentGenerator.hh

## 6.47 artdaq::TransferInterface::Config Struct Reference

Configuration of the [TransferInterface](#). May be used for parameter validation

```
#include <artdaq/TransferPlugins/TransferInterface.hh>
```

### Public Attributes

- fhicl::Atom< int > [source\\_rank](#) {fhicl::Name{"source\_rank"}, fhicl::Comment{"The rank that data is coming from"}, my\_rank}  
*"source\_rank" (Default: my\_rank) : The rank that data is coming from*
- fhicl::Atom< int > [destination\\_rank](#) {fhicl::Name{"destination\_rank"}, fhicl::Comment{"The rank that data is going to"}, my\_rank}  
*"destination\_rank" (Default: my\_rank) : The rank that data is going to*
- fhicl::Atom< std::string > [unique\\_label](#) {fhicl::Name{"unique\_label"}, fhicl::Comment{"A label that uniquely identifies the [TransferInterface](#) instance"}, "transfer\_between\_[source\_rank]\_and\_[destination\_rank]"}  
*"unique\_label" (Default: "transfer\_between\_[source\_rank]\_and\_[destination\_rank]") : A label that uniquely identifies the [TransferInterface](#) instance*
- fhicl::Atom< size\_t > [buffer\\_count](#) {fhicl::Name{"buffer\_count"}, fhicl::Comment{"How many Fragments can the [TransferInterface](#) handle simultaneously"}, 10}  
*"buffer\_count" (Default: 10) : How many Fragments can the [TransferInterface](#) handle simultaneously*
- fhicl::Atom< size\_t > [max\\_fragment\\_size](#) {fhicl::Name{"max\_fragment\_size\_words"}, fhicl::Comment{"The maximum Fragment size expected.May be used for static memory allocation, and will cause errors if larger Fragments are sent."}, 1024}  
*"max\_fragment\_size\_words" (Default: 1024) : The maximum Fragment size expected.May be used for static memory allocation, and will cause errors if larger Fragments are sent.*

### 6.47.1 Detailed Description

Configuration of the [TransferInterface](#). May be used for parameter validation

Definition at line 38 of file TransferInterface.hh.

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

- artdaq/artdaq/TransferPlugins/TransferInterface.hh

## 6.48 artdaq::artdaqapp::Config Struct Reference

Configuration of artdaqapp. May be used for parameter validation

```
#include <proto/artdaqapp.hh>
```

### Public Attributes

- fhicl::Atom< std::string > [application\\_name](#) {fhicl::Name{"application\_name"}, fhicl::Comment{"Name to use for metrics and logging"}, "BoardReader"}  
*"application\_name" (Default: artdaq::detail::TaskTypeToString(task)): Name to use for metrics and logging*
- fhicl::Atom< bool > [replace\\_image\\_name](#) {fhicl::Name{"replace\_image\_name"}, fhicl::Comment{"Replace the application image name with [application\\_name](#)"}, false}  
*"replace\_image\_name" (Default: false): Replace the application image name with application\_name*
- fhicl::Atom< int > [rank](#) {fhicl::Name{"rank"}, fhicl::Comment{"The \"rank\" of the application, used for configuring data transfers"}}  
*"rank": The "rank" of the application, used for configuring data transfers*
- fhicl::TableFragment  
 < [artdaq::CommanderInterface::Config](#) > [commanderPluginConfig](#)
- fhicl::Atom< bool > [auto\\_run](#) {fhicl::Name{"auto\_run"}, fhicl::Comment{"Whether to automatically start a run"}, false}  
*"auto\_run" (Default: false): Whether to automatically start a run*
- fhicl::Atom< int > [run\\_number](#) {fhicl::Name{"run\_number"}, fhicl::Comment{"Run number to use for automatic run"}, 101}  
*"run\_number" (Default: 101): Run number to use for automatic run*
- fhicl::Atom< uint64\_t > [transition\\_timeout](#) {fhicl::Name{"transition\_timeout"}, fhicl::Comment{"Timeout to use for automatic transitions"}, 30}  
*"transition\_timeout" (Default: 30): [Timeout](#) to use for automatic transitions*
- fhicl::TableFragment  
 < [artdaq::PortManager::Config](#) > [portsConfig](#)  
*Configuration for artdaq Ports.*

### 6.48.1 Detailed Description

Configuration of artdaqapp. May be used for parameter validation

Definition at line 29 of file artdaqapp.hh.

## 6.48.2 Member Data Documentation

### 6.48.2.1 fhicl::TableFragment<artdaq::CommanderInterface::Config> artdaq::artdaqapp::Config::commanderPluginConfig

Configuration for the [CommanderInterface](#). See [artdaq::CommanderInterface::Config](#)

Definition at line 37 of file artdaqapp.hh.

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

- artdaq/proto/artdaqapp.hh

## 6.49 Config Struct Reference

Configuration for simple\_metric\_sender.

### Public Attributes

- fhicl::TableFragment  
< artdaq::MetricManager::Config > [metricmanager\\_config](#)  
*Configuration for MetricManager.*

### 6.49.1 Detailed Description

Configuration for simple\_metric\_sender.

Definition at line 9 of file simple\_metric\_sender.cc.

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

- artdaq/proto/simple\_metric\_sender.cc

## 6.50 ArtdaqGlobalsService::Config Struct Reference

Allowed Configuration parameters of [ArtdaqGlobalsService](#). May be used for configuration validation

```
#include <artdaq/ArtModules/ArtdaqGlobalsService.h>
```

### 6.50.1 Detailed Description

Allowed Configuration parameters of [ArtdaqGlobalsService](#). May be used for configuration validation

Definition at line 22 of file ArtdaqGlobalsService.h.

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

- artdaq/artdaq/ArtModules/ArtdaqGlobalsService.h

## 6.51 mfplugins::ELArtdaqMetric::Config Struct Reference

Configuration Parameters for [ELArtdaqMetric](#).

## Public Attributes

- fhicl::TableFragment  
< ELdestination::Config > [elDestConfig](#)  
*ELDestination common parameters.*
- fhicl::Atom< bool > [showDebug](#)  
*"showDebug" (Default: False): Send metrics for Debug messages*
- fhicl::Atom< bool > [showInfo](#)  
*"showInfo" (Default: False): Send metrics for Info messages*
- fhicl::Atom< bool > [showWarning](#)  
*"showWarning" (Default: true): Send metrics for Warning messages*
- fhicl::Atom< bool > [showError](#)  
*"showError" (Default: true): Send metrics for Error messages*
- fhicl::Atom< size\_t > [messageLength](#) = fhicl::Atom<size\_t>(fhicl::Name{"messageLength"}, fhicl::Comment{"Maximum length of metric titles (0 for unlimited)"}, 40)  
*"messageLength" (Default: 40): Number of characters to use for metric title*
- fhicl::Atom< size\_t > [metricLevelOffset](#) = fhicl::Atom<size\_t>(fhicl::Name{"metricLevelOffset"}, fhicl::Comment{"Offset for Metric Levels (+0: summary rates, +1 errors, ...)"}, 10)  
*"metricLevelOffset" (Default: 10): Offset for Metric Levels (+0: summary rates, +1 errors, ...)*

### 6.51.1 Detailed Description

Configuration Parameters for [ELArtdaqMetric](#).

Definition at line 29 of file ArtdaqMetric\_mfPlugin.cc.

### 6.51.2 Member Data Documentation

#### 6.51.2.1 fhicl::Atom<bool> mfplugins::ELArtdaqMetric::Config::showDebug

**Initial value:**

```
=
    fhicl::Atom<bool>{fhicl::Name{"showDebug"}, fhicl::Comment{"Send Metrics for Debug messages"},
    false}
```

"showDebug" (Default: False): Send metrics for Debug messages

Definition at line 34 of file ArtdaqMetric\_mfPlugin.cc.

#### 6.51.2.2 fhicl::Atom<bool> mfplugins::ELArtdaqMetric::Config::showError

**Initial value:**

```
=
    fhicl::Atom<bool>{fhicl::Name{"showError"}, fhicl::Comment{"Send Metrics for Error messages"},
    true}
```

"showError" (Default: true): Send metrics for Error messages

Definition at line 43 of file ArtdaqMetric\_mfPlugin.cc.

### 6.51.2.3 fhicl::Atom<bool> mfplugins::ELArtdaqMetric::Config::showInfo

#### Initial value:

```
=
    fhicl::Atom<bool>{fhicl::Name{"showInfo"}, fhicl::Comment{"Send Metrics for Info messages"},
    false}
```

"showInfo" (Default: False): Send metrics for Info messages

Definition at line 37 of file ArtdaqMetric\_mfPlugin.cc.

### 6.51.2.4 fhicl::Atom<bool> mfplugins::ELArtdaqMetric::Config::showWarning

#### Initial value:

```
=
    fhicl::Atom<bool>{fhicl::Name{"showWarning"}, fhicl::Comment{"Send Metrics for Warning messages"},
    true}
```

"showWarning" (Default: true): Send metrics for Warning messages

Definition at line 40 of file ArtdaqMetric\_mfPlugin.cc.

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

- artdaq/artdaq/ArtModules/ArtdaqMetric\_mfPlugin.cc

## 6.52 ArtdaqSharedMemoryService::Config Struct Reference

Allowed Configuration parameters of NetMonTransportService. May be used for configuration validation

### Public Attributes

- fhicl::Atom< uint32\_t > [shared\\_memory\\_key](#) {fhicl::Name{"shared\_memory\_key"}, fhicl::Comment{"Key to use when connecting to shared memory. Will default to 0xBEE70000 + getpid()."}, 0xBEE70000}  
*"shared\_memory\_key" (Default: 0xBEE70000 + pid): Key to use when connecting to shared memory. Will default to 0xBEE70000 + getpid().*
- fhicl::Atom< uint32\_t > [broadcast\\_shared\\_memory\\_key](#) {fhicl::Name{"broadcast\_shared\_memory\_key"}, fhicl::Comment{"Key to use when connecting to broadcast shared memory. Will default to 0xCEE70000 + getpid()."}, 0xCEE70000}  
*"shared\_memory\_key" (Default: 0xCEE70000 + pid): Key to use when connecting to broadcast shared memory. Will default to 0xCEE70000 + getpid().*
- fhicl::Atom< int > [rank](#) {fhicl::Name{"rank"}, fhicl::Comment{"Rank of this artdaq application. Used for data transfers"}}  
*"rank" (OPTIONAL) : The rank of this applicaiton, for use by non - artdaq applications running NetMonTransportService*

### 6.52.1 Detailed Description

Allowed Configuration parameters of NetMonTransportService. May be used for configuration validation

Definition at line 33 of file ArtdaqSharedMemoryService\_service.cc.

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



- `artdaq/artdaq/ArtModules/ArtdaqSharedMemoryService_service.cc`

## 6.53 art::Config Struct Reference

Required configuration for art processes started by artdaq, with artdaq-specific defaults where applicable

```
#include <artdaq/ArtModules/detail/ArtConfig.hh>
```

### Public Attributes

- `fhicl::Table< art::ServicesConfig > services {fhicl::Name{"services"}}`  
*Services block.*
- `fhicl::Table< art::PhysicsConfig > physics {fhicl::Name{"physics"}}`  
*Physics block.*
- `fhicl::Table< art::OutputsConfig > outputs {fhicl::Name{"outputs"}}`  
*Outputs block.*
- `fhicl::Table< art::SourceConfig > source {fhicl::Name{"source"}}`
- `fhicl::Atom< std::string > process_name {fhicl::Name{"process_name"}, fhicl::Comment{"Name of this art processing job"}, "DAQ"}`  
*"process\_name" (Default: "DAQ"): Name of this art processing job*

### 6.53.1 Detailed Description

Required configuration for art processes started by artdaq, with artdaq-specific defaults where applicable

Definition at line 154 of file ArtConfig.hh.

### 6.53.2 Member Data Documentation

6.53.2.1 `fhicl::Table<art::SourceConfig> art::Config::source {fhicl::Name{"source"}}`

Source block

Definition at line 159 of file ArtConfig.hh.

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

- `artdaq/artdaq/ArtModules/detail/ArtConfig.hh`

## 6.54 art::RootDAQOut::Config Struct Reference

### Classes

- struct [FileNameSubstitution](#)
- struct [KeysToIgnore](#)
- struct [NewSubStringForApp](#)

## Public Types

- using **Name** = fhicl::Name
- using **Comment** = fhicl::Comment
- template<typename T >  
using **Atom** = fhicl::Atom< T >
- template<typename T >  
using **OptionalAtom** = fhicl::OptionalAtom< T >

## Public Attributes

- fhicl::TableFragment  
  < OutputModule::Config > **omConfig**
- Atom< string > **catalog** {Name("catalog"), ""}
- OptionalAtom< bool > **dropAllEvents** {Name("dropAllEvents")}
- Atom< bool > **dropAllSubRuns** {Name("dropAllSubRuns"), false}
- OptionalAtom< bool > **fastCloning** {Name("fastCloning")}
- Atom< string > **tmpDir** {Name("tmpDir"), default\_tmpDir}
- Atom< int > **compressionLevel** {Name("compressionLevel"), 7}
- Atom< unsigned > **freePercent** {Name("freePercent"), 0}
- Atom< unsigned > **freeMB** {Name("freeMB"), 0}
- Atom< int64\_t > **saveMemoryObjectThreshold**
- Atom< int64\_t > **treeMaxVirtualSize** {Name("treeMaxVirtualSize"), -1}
- Atom< int > **splitLevel** {Name("splitLevel"), 1}
- Atom< int > **basketSize** {Name("basketSize"), 16384}
- Atom< bool > **dropMetaDataForDroppedData**
- Atom< string > **dropMetaData** {Name("dropMetaData"), "NONE"}
- Atom< bool > **writeParameterSets** {Name("writeParameterSets"), true}
- fhicl::Table  
  < ClosingCriteria::Config > **fileProperties**
- fhicl::TableFragment  
  < detail::SafeFileNameConfig > **safeFileName**
- Atom< int > **firstLoggerRank** {Name("firstLoggerRank"), -1}
- fhicl::OptionalSequence  
  < fhicl::Table  
  < [FileNameSubstitution](#) > > **fileNameSubstitutions** {Name("fileNameSubstitutions")}

### 6.54.1 Detailed Description

Definition at line 102 of file RootDAQOut\_module.cc.

### 6.54.2 Member Data Documentation

#### 6.54.2.1 Atom<bool> art::RootDAQOut::Config::dropMetaDataForDroppedData

##### Initial value:

```
{Name("dropMetaDataForDroppedData"),  
                                     false}
```

Definition at line 124 of file RootDAQOut\_module.cc.

## 6.54.2.2 fhicl::Table&lt;ClosingCriteria::Config&gt; art::RootDAQOut::Config::fileProperties

## Initial value:

```
{
    Name("fileProperties"),
    Comment("The 'fileProperties' parameter is specified to enable "
           "output-file switching.")}
```

Definition at line 128 of file RootDAQOut\_module.cc.

## 6.54.2.3 Atom&lt;int64\_t&gt; art::RootDAQOut::Config::saveMemoryObjectThreshold

## Initial value:

```
{Name("saveMemoryObjectThreshold"),
                                     -11}
```

Definition at line 119 of file RootDAQOut\_module.cc.

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

- artdaq/artdaq/ArtModules/RootDAQOutput-s124/RootDAQOut\_module.cc

## 6.55 artdaq::GenericFragmentSimulator::Config Struct Reference

Configuration of the [GenericFragmentSimulator](#). May be used for parameter validation

```
#include <artdaq/DAQdata/GenericFragmentSimulator.hh>
```

## Public Attributes

- fhicl::Atom< size\_t > [content\\_selection](#) {fhicl::Name{"content\_selection"}, fhicl::Comment{"What type of data to fill in generated Fragment payloads"}, 0}
- fhicl::Atom< size\_t > [payload\\_size](#) {fhicl::Name{"payload\_size"}, fhicl::Comment{"The size (in words) of the Fragment payload"}, 10240}
 

*"payload\_size" (Default: 10240) : The size(in words) of the Fragment payload*
- fhicl::Atom< bool > [want\\_random\\_payload\\_size](#) {fhicl::Name{"want\_random\_payload\_size"}, fhicl::Comment{"Whether payload size should be sampled from a random distribution"}, false}
 

*"want\_random\_payload\_size" (Default: false) : Whether payload size should be sampled from a random distribution*
- fhicl::Atom< int64\_t > [random\\_seed](#) {fhicl::Name{"random\_seed"}, fhicl::Comment{"Random seed for random number distributions"}, 314159}
 

*"random\_seed" (Default: 314159) : Random seed for random number distributions*
- fhicl::Atom< size\_t > [fragments\\_per\\_event](#) {fhicl::Name{"fragments\_per\_event"}, fhicl::Comment{"The number of Fragment objects to generate for each sequence ID"}, 5}
 

*"fragments\_per\_event" (Default: 5) : The number of Fragment objects to generate for each sequence ID*
- fhicl::Atom
  - < Fragment::fragment\_id\_t > [starting\\_fragment\\_id](#) {fhicl::Name{"starting\_fragment\_id"}, fhicl::Comment{"The first Fragment ID handled by this GenericFragmentSimulator."}, 0}

### 6.55.1 Detailed Description

Configuration of the [GenericFragmentSimulator](#). May be used for parameter validation

Definition at line 42 of file GenericFragmentSimulator.hh.

### 6.55.2 Member Data Documentation

6.55.2.1 `fhicl::Atom<size_t> artdaq::GenericFragmentSimulator::Config::content_selection {fhicl::Name{"content_selection"}, fhicl::Comment{"What type of data to fill in generated Fragment payloads"}, 0}`

"content\_selection" (Default: 0) : What type of data to fill in generated Fragment payloads

- 0 : Use uninitialized memory
- 1 : Use the Fragment ID
- 2 : Use random data
- 3 : Use the word 0xDEADBEEFDEADBEEF

Definition at line 49 of file GenericFragmentSimulator.hh.

6.55.2.2 `fhicl::Atom<Fragment::fragment_id_t> artdaq::GenericFragmentSimulator::Config::starting_fragment_id {fhicl::Name{"starting_fragment_id"}, fhicl::Comment{"The first Fragment ID handled by this GenericFragmentSimulator."}, 0}`

"starting\_fragment\_id" (Default: 0) : The first Fragment ID handled by this [GenericFragmentSimulator](#).

- Fragment IDs will be starting\_fragment\_id to starting\_fragment\_id + fragments\_per\_event.

Definition at line 60 of file GenericFragmentSimulator.hh.

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

- artdaq/artdaq/DAQdata/GenericFragmentSimulator.hh

## 6.56 artdaq::HostMap::Config Struct Reference

Template for the host\_map configuration parameter.

```
#include <artdaq/DAQdata/HostMap.hh>
```

### Public Attributes

- `fhicl::Sequence< fhicl::Table < HostConfig > > host_map {fhicl::Name("host_map"), fhicl::Comment("List of artdaq applications by rank and location")}`

*List of artdaq applications by rank and location. See [artdaq::HostMap::HostConfig](#)*

### 6.56.1 Detailed Description

Template for the host\_map configuration parameter.

Definition at line 33 of file HostMap.hh.

### 6.56.2 Member Data Documentation

6.56.2.1 fhicl::Sequence<fhicl::Table<HostConfig> > artdaq::HostMap::Config::host\_map {fhicl::Name("host\_map"), fhicl::Comment("List of artdaq applications by rank and location")}

List of artdaq applications by rank and location. See [artdaq::HostMap::HostConfig](#)

Definition at line 38 of file HostMap.hh.

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

- artdaq/artdaq/DAQdata/HostMap.hh

## 6.57 artdaq::PortManager::Config Struct Reference

Configuration of [PortManager](#). May be used for parameter validation

```
#include <artdaq/DAQdata/PortManager.hh>
```

### Public Attributes

- fhicl::Atom< int > [artdaq\\_base\\_port](#) {fhicl::Name("artdaq\_base\_port"), fhicl::Comment{"Base port for all artdaq partitions. Should be the same across all running systems. Overridden by environment variable ARTDAQ\_BASE\_PORT."}, 10000 }  
*"artdaq\_base\_port" (Default: 10000): Base port for all artdaq partitions. Should be the same across all running systems. Overridden by environment variable ARTDAQ\_BASE\_PORT.*
- fhicl::Atom< int > [ports\\_per\\_partition](#) {fhicl::Name("ports\_per\_partition"), fhicl::Comment{"Number of ports to reserve for each partition. Should be the same across all running systems. Overridden by environment variable ARTDAQ\_PORTS\_PER\_PARTITION."}, 1000 }  
*"ports\_per\_partition" (Default: 1000): Number of ports to reserve for each partition. Should be the same across all running systems. Overridden by environment variable ARTDAQ\_PORTS\_PER\_PARTITION.*
- fhicl::Atom< std::string > [multicast\\_output\\_interface](#) {fhicl::Name{"multicast\_output\_interface"}, fhicl::Comment{"Name of the interface to be used for all multicasts. Has precedence over \"multicast\_output\_network\". OPTIONAL"}, ""}  
*"multicast\_output\_interface" (Default: ""): Name of the interface to be used for all multicasts. Has precedence over \"multicast\_output\_network\". OPTIONAL*
- fhicl::Atom< std::string > [multicast\\_output\\_network](#) {fhicl::Name{"multicast\_output\_network"}, fhicl::Comment{"Address in network to be used for all multicasts. OPTIONAL"}, "0.0.0.0"}  
*"multicast\_output\_network" (Default: "0.0.0.0"): Address in network to be used for all multicasts. OPTIONAL*
- fhicl::Atom< int > [multicast\\_group\\_offset](#) {fhicl::Name{"multicast\_group\_offset"}, fhicl::Comment{"Number to add to last byte of multicast groups, to avoid problematic 0s."}, 128 }  
*"multicast\_group\_offset" (Default: 128): Number to add to last byte of multicast groups, to avoid problematic 0s.*
- fhicl::Atom< int > [routing\\_token\\_port\\_offset](#) {fhicl::Name{"routing\_token\_port\_offset"}, fhicl::Comment{"Offset from partition base port for routing token ports"}, 10 }  
*"routing\_token\_port\_offset" (Default: 10): Offset from partition base port for routing token ports*

- fhicl::Atom< int > [routing\\_table\\_ack\\_port\\_offset](#) {fhicl::Name{"routing\_table\_ack\_port\_offset"}, fhicl::Comment{"Offset from partition base port for routing table ack ports"}, 30 }  
*"routing\_table\_ack\_port\_offset" (Default: 30): Offset from partition base port for routing table ack ports*
- fhicl::Atom< int > [xmlrpc\\_port\\_offset](#) {fhicl::Name{"xmlrpc\_port\_offset"}, fhicl::Comment{"Offset from partition base port for XMLRPC ports"}, 100 }  
*"xmlrpc\_port\_offset" (Default: 100): Offset from partition base port for XMLRPC ports*
- fhicl::Atom< int > [tcp\\_socket\\_port\\_offset](#) {fhicl::Name{"tcp\_socket\_port\_offset"}, fhicl::Comment{"Offset from partition base port for TCP Socket ports"}, 500 }  
*"tcp\_socket\_port\_offset" (Default: 500): Offset from partition base port for TCP Socket ports*
- fhicl::Atom< int > [request\\_port](#) {fhicl::Name{"request\_port"}, fhicl::Comment{"Port to use for request messages (multicast)"}, 3001 }  
*"request\_port" (Default: 3001): Port to use for request messages (multicast)*
- fhicl::Atom< std::string > [request\\_pattern](#) {fhicl::Name{"request\_pattern"}, fhicl::Comment{"Pattern to use to generate request multicast group. PPP => Partition number, SSS => Subsystem ID (default 0)"}, "227.128.PPP.SSS"}  
*"request\_pattern" (Default: "227.128.PPP.SSS"): Pattern to use to generate request multicast group. PPP => Partition number, SSS => Subsystem ID (default 0)*
- fhicl::Atom< int > [routing\\_table\\_port](#) {fhicl::Name{"routing\_table\_port"}, fhicl::Comment{"Port to use for routing tables (multicast)"}, 3001 }  
*"routing\_table\_port" (Default: 3001): Port to use for routing tables (multicast)*
- fhicl::Atom< std::string > [routing\\_table\\_pattern](#) {fhicl::Name{"routing\_table\_pattern"}, fhicl::Comment{"Pattern to use to generate routing table multicast group. PPP => Partition number, SSS => Subsystem ID (default 0)."}, "227.129.PPP.SSS"}  
*"routing\_table\_pattern" (Default: "227.129.PPP.SSS"): Pattern to use to generate routing table multicast group. PPP => Partition number, SSS => Subsystem ID (default 0).*
- fhicl::Atom< int > [multicast\\_transfer\\_port\\_offset](#) {fhicl::Name{"multicast\_transfer\_port\_offset"}, fhicl::Comment{"Offset to use for [MulticastTransfer](#) ports (port = offset + rank)"}, 1024 }  
*"multicast\_transfer\_port\_offset" (Default: 1024): Offset to use for [MulticastTransfer](#) ports (port = offset + rank)*
- fhicl::Atom< std::string > [multicast\\_transfer\\_pattern](#) {fhicl::Name{"multicast\_transfer\_pattern"}, fhicl::Comment{"Pattern to use to generate Multicast Transfer group address. PPP => Partition Number, SSS => Subsystem ID (default 0), RRR => Rank"}, "227.130.14.PPP"}  
*"multicast\_transfer\_pattern" (Default: "227.130.14.PPP"): Pattern to use to generate Multicast Transfer group address. PPP => Partition Number, SSS => Subsystem ID (default 0), RRR => Rank*

### 6.57.1 Detailed Description

Configuration of [PortManager](#). May be used for parameter validation

Definition at line 36 of file PortManager.hh.

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

- artdaq/artdaq/DAQdata/PortManager.hh

## 6.58 artdaqtest::BrokenTransferTest::Config Struct Reference

Configuration parameters for [BrokenTransferTest](#)

```
#include <test/TransferPlugins/BrokenTransferTest.hh>
```

## Public Attributes

- fhicl::Atom< size\_t > [fragment\\_rate\\_hz](#) {fhicl::Name{"fragment\_rate\_hz"}, fhicl::Comment{"The rate at which to generate Fragments, in Hz"}, 10}  
*"fragment\_rate\_hz" (Default: 10): The rate at which to generate Fragments, in Hz*
- fhicl::Atom< bool > [reliable\\_mode](#) {fhicl::Name{"reliable\_mode"}, fhicl::Comment{"Whether to use reliable-mode transfers (true) or min-blocking (false)"}, true}  
*"reliable\_mode" (Default: true): Whether to use reliable-mode transfers (true) or min-blocking (false)*
- fhicl::Atom< size\_t > [fragment\\_size](#) {fhicl::Name{"fragment\_size"}, fhicl::Comment{"The size of generated Fragments, in Fragment words"}, 0x10000}  
*"fragment\_size" (Default: 0x10000): The size of generated Fragments, in Fragment words*
- fhicl::Atom< size\_t > [send\\_timeout\\_us](#) {fhicl::Name{"send\_timeout\_us"}, fhicl::Comment{"The timeout for min-blocking mode sends"}, 100000}  
*"send\_timeout\_us" (Default: 100000): The timeout for min-blocking mode sends*
- fhicl::Atom< size\_t > [transfer\\_buffer\\_count](#) {fhicl::Name{"transfer\_buffer\_count"}, fhicl::Comment{"The number of buffers in the Transfer Plugins"}, 10}  
*"transfer\_buffer\_count" (Default: 10): The number of buffers in the Transfer Plugins*
- fhicl::Atom< size\_t > [event\\_buffer\\_count](#) {fhicl::Name{"event\_buffer\_count"}, fhicl::Comment{"The number of \"EventBuilder\" buffers on the receiver end"}, 20}  
*"event\_buffer\_count" (Default: 20): The number of \"EventBuilder\" buffers on the receiver end*
- fhicl::Atom< size\_t > [event\\_buffer\\_timeout\\_us](#) {fhicl::Name{"event\_buffer\_timeout\_us"}, fhicl::Comment{"The timeout for \"EventBuilder\" buffers to be marked incomplete and abandoned"}, 1000000}  
*"event\_buffer\_timeout\_us" (Default: 1000000): The timeout for \"EventBuilder\" buffers to be marked incomplete and abandoned*
- fhicl::Table  
 < artdaq::TransferInterface::Config > [default\\_transfer\\_ps](#) {fhicl::Name{"default\_transfer\_ps"}, fhicl::Comment{"The default ParameterSet to use for transfers. Will have transferPluginType, destination\_rank, buffer\_count and source\_rank overridden. If max\_fragment\_size\_words is unspecified, will be set using [fragment\\_size](#)"}}  
*"default\_transfer\_ps" (Default: {}): The default ParameterSet to use for transfers. Will have transferPluginType, destination\_rank, buffer\_count and source\_rank overridden. If max\_fragment\_size\_words is unspecified, will be set using fragment\_size*
- fhicl::Atom< std::string > [transfer\\_to\\_use](#) {fhicl::Name{"transfer\_to\_use"}, fhicl::Comment{"The name of the Transfer Plugin to use"}, "Shmem"}  
*"transfer\_to\_use" (Default: "Shmem"): The name of the Transfer Plugin to use*

## 6.58.1 Detailed Description

Configuration parameters for [BrokenTransferTest](#)

Definition at line 33 of file BrokenTransferTest.hh.

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

- artdaq/test/TransferPlugins/BrokenTransferTest.hh

## 6.59 artdaq::DataSenderManager::Config Struct Reference

Configuration of [DataSenderManager](#). May be used for parameter validation

```
#include <artdaq/DAQrate/DataSenderManager.hh>
```

## Public Attributes

- fhicl::Atom< bool > [broadcast\\_sends](#) {fhicl::Name{"broadcast\_sends"}, fhicl::Comment{"Send all Fragments to all [destinations](#)"}, false}  
*"broadcast\_sends" (Default: false): Send all Fragments to all destinations*
- fhicl::Atom< bool > [nonblocking\\_sends](#) {fhicl::Name{"nonblocking\_sends"}, fhicl::Comment{"Whether sends should block. Used for DL->DISP connection."}, false}  
*"nonblocking\_sends" (Default: false): If true, will use non-reliable mode of [TransferInterface](#) plugins*
- fhicl::Atom< size\_t > [send\\_timeout\\_us](#) {fhicl::Name{"send\_timeout\_usec"}, fhicl::Comment{"Timeout for sends in non-reliable modes (broadcast and nonblocking)"}, 5000000}  
*"send\_timeout\_usec" (Default: 5000000 (5 seconds): [Timeout](#) for sends in non-reliable modes (broadcast and nonblocking)*
- fhicl::Atom< size\_t > [send\\_retry\\_count](#) {fhicl::Name{"send\_retry\_count"}, fhicl::Comment{"Number of times to retry a send in non-reliable mode"}, 2}  
*"send\_retry\_count" (Default: 2): Number of times to retry a send in non-reliable mode*
- fhicl::OptionalTable  
 < [artdaq::TableReceiver::Config](#) > [routing\\_table\\_config](#) {fhicl::Name{"routing\_table\_config"}}
- fhicl::OptionalTable  
 < [DestinationsConfig](#) > [destinations](#) {fhicl::Name{"destinations"}}
- fhicl::TableFragment  
 < [artdaq::HostMap::Config](#) > [host\\_map](#)  
*Optional host\_map configuration (Can also be specified in each [DestinationsConfig](#) entry. See [artdaq::HostMap::Config](#).*
- fhicl::Sequence< size\_t > [enabled\\_destinations](#) {fhicl::Name{"enabled\_destinations"}, fhicl::Comment{"List of destination ranks to activate (must be defined in [destinations](#) block)"}, std::vector<size\_t>()}  
*enabled\_destinations" (OPTIONAL): If specified, only the destination ranks listed will be enabled. If not specified, all destinations will be enabled.*

### 6.59.1 Detailed Description

Configuration of [DataSenderManager](#). May be used for parameter validation

Definition at line 50 of file DataSenderManager.hh.

### 6.59.2 Member Data Documentation

6.59.2.1 fhicl::OptionalTable<[DestinationsConfig](#)> [artdaq::DataSenderManager::Config::destinations](#)  
 {fhicl::Name{"destinations"}}

"destinations" (Default: Empty ParameterSet): FHiCL table for [TransferInterface](#) configurations for each destination. See [artdaq::DataSenderManager::DestinationsConfig](#) NOTE: "destination\_rank" MUST be specified (and unique) for each destination!

Definition at line 63 of file DataSenderManager.hh.

6.59.2.2 fhicl::OptionalTable<[artdaq::TableReceiver::Config](#)> [artdaq::DataSenderManager::Config::routing\\_table\\_config](#)  
 {fhicl::Name{"routing\_table\_config"}}

Configuration for Routing Table reception. See [artdaq::DataSenderManager::RoutingTableConfig](#)

Definition at line 60 of file DataSenderManager.hh.

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



- artdaq/artdaq/DAQrate/DataSenderManager.hh

## 6.60 artdaq::RequestReceiver::Config Struct Reference

Configuration of the [RequestReceiver](#). May be used for parameter validation

```
#include <artdaq/DAQrate/detail/RequestReceiver.hh>
```

### Public Attributes

- fhicl::Atom< bool > [receive\\_requests](#) {fhicl::Name{"receive\_requests"}, fhicl::Comment{"Whether this [RequestReceiver](#) will listen for requests"}, false}  
*"receive\_requests" (Default: false): Whether this [RequestReceiver](#) will listen for requests*
- fhicl::Atom< int > [request\\_port](#) {fhicl::Name{"request\_port"}, fhicl::Comment{"Port to listen for request messages on"}, 3001}  
*"request\_port" (Default: 3001) : Port on which data requests will be received*
- fhicl::Atom< std::string > [request\\_addr](#) {fhicl::Name{"request\_address"}, fhicl::Comment{"Multicast address to listen for request messages on"}, "227.128.12.26"}  
*"request\_address" (Default: "227.128.12.26") : Address which [CommandableFragmentGenerator](#) will listen for requests on*
- fhicl::Atom< std::string > [output\\_address](#) {fhicl::Name{"multicast\_interface\_ip"}, fhicl::Comment{"Use this host-name for multicast (to assign to the proper NIC)"}, "0.0.0.0"}  
*"multicast\_interface\_ip" (Default: "0.0.0.0") : Use this hostname for multicast(to assign to the proper NIC)*
- fhicl::Atom< size\_t > [end\\_of\\_run\\_timeout\\_ms](#) {fhicl::Name{"end\_of\_run\_quiet\_timeout\_ms"}, fhicl::Comment{"Amount of time (in ms) to wait for no new requests when a Stop transition is pending"}, 1000}  
*"end\_of\_run\_quiet\_timeout\_ms" (Default: 1000) : Time, in milliseconds, that the entire system must be quiet for check-stop to return true in request mode. **DO NOT EDIT UNLESS YOU KNOW WHAT YOU ARE DOING!***

### 6.60.1 Detailed Description

Configuration of the [RequestReceiver](#). May be used for parameter validation

Definition at line 31 of file RequestReceiver.hh.

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

- artdaq/artdaq/DAQrate/detail/RequestReceiver.hh

## 6.61 artdaq::RequestSender::Config Struct Reference

Configuration of the [RequestSender](#). May be used for parameter validation

```
#include <artdaq/DAQrate/detail/RequestSender.hh>
```

### Public Attributes

- fhicl::Atom< bool > [send\\_requests](#) {fhicl::Name{"send\_requests"}, fhicl::Comment{"Enable sending Data Request messages"}, false}  
*"send\_requests" (Default: false): Whether to send DataRequests when new sequence IDs are seen*

- `fhicl::Atom< int > request_port` {fhicl::Name{"request\_port"}, fhicl::Comment{"Port to send DataRequests on"}, 3001}  
*"request\_port" (Default: 3001): Port to send DataRequests on*
- `fhicl::Atom< size_t > request_delay_ms` {fhicl::Name{"request\_delay\_ms"}, fhicl::Comment{"How long to wait before sending new DataRequests"}, 10}  
*"request\_delay\_ms" (Default: 10): How long to wait before sending new DataRequests*
- `fhicl::Atom< size_t > request_shutdown_timeout_us` {fhicl::Name{"request\_shutdown\_timeout\_us"}, fhicl::Comment{"How long to wait for pending requests to be sent at shutdown"}, 100000}  
*"request\_shutdown\_timeout\_us" (Default: 100000 us): How long to wait for pending requests to be sent at shutdown*
- `fhicl::Atom< std::string > output_address` {fhicl::Name{"multicast\_interface\_ip"}, fhicl::Comment{"Use this host-name for multicast output(to assign to the proper NIC)"}, "0.0.0.0"}  
*"multicast\_interface\_ip" (Default: "0.0.0.0"): Use this hostname for multicast output (to assign to the proper NIC)*
- `fhicl::Atom< std::string > request_address` {fhicl::Name{"request\_address"}, fhicl::Comment{"Multicast address to send DataRequests to"}, "227.128.12.26"}  
*"request\_address" (Default: "227.128.12.26"): Multicast address to send DataRequests to*
- `fhicl::Atom< size_t > min_request_interval_ms` {fhicl::Name{"min\_request\_interval\_ms"}, fhicl::Comment{"Minimum time between automatic sends (ignored in EndOfRun RequetsMode)"}, 100}  
*"min\_request\_interval\_ms" (Default: 500): Minimum time between automatic sends (ignored in EndOfRun RequetsMode)*

### 6.61.1 Detailed Description

Configuration of the [RequestSender](#). May be used for parameter validation

Definition at line 32 of file `RequestSender.hh`.

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

- `artdaq/artdaq/DAQrate/detail/RequestSender.hh`

## 6.62 artdaq::TableReceiver::Config Struct Reference

Configuration for Routing table reception

```
#include <artdaq/DAQrate/detail/TableReceiver.hh>
```

### Public Attributes

- `fhicl::Atom< bool > use_routing_manager` {fhicl::Name{"use\_routing\_manager"}, fhicl::Comment{"True if using the Routing Manager"}, false}  
*"use\_routing\_manager" (Default: false): True if using the Routing Manager*
- `fhicl::Atom< bool > route_on_request_mode` {fhicl::Name{"route\_on\_request\_mode"}, fhicl::Comment{"True if a request for routing information should be sent to the RoutingManager (versus RoutingManager pushing table updates)."}, false}  
*"route\_on\_request\_mode" (Default: false): True if a request for routing information should be sent to the RoutingManager (versus RoutingManager pushing table updates).*
- `fhicl::Atom< bool > use_routing_table_thread` {fhicl::Name{"use\_routing\_table\_thread"}, fhicl::Comment{"True if a thread should be run to receive routing updates. Required if route\_on\_request\_mode is false."}, true}  
*"use\_routing\_table\_thread" (Default: true): True if a thread should be run to receive routing updates. Required if route\_on\_request\_mode is false.*

- fhicl::Atom< int > [table\\_port](#) {fhicl::Name{"table\_update\_port"}, fhicl::Comment{"Port to connect to for receiving table updates"}, 35556}  
*"table\_update\_port" (Default: 35556): Port to connect to for receiving table updates*
- fhicl::Atom< std::string > [routing\\_manager\\_hostname](#) {fhicl::Name{"routing\_manager\_hostname"}, fhicl::Comment{"outingManager hostname for Table connection"}, "localhost"}  
*"routing\_manager\_hostname" (Default: "localhost"): RoutingManager hostname for Table connection*
- fhicl::Atom< int > [routing\\_timeout\\_ms](#) {fhicl::Name{"routing\_timeout\_ms"}, fhicl::Comment{"Time to wait (in ms) for a routing table update"}, 1000}  
*"routing\_timeout\_ms" (Default: 1000): Time to wait for a routing table update*
- fhicl::Atom< size\_t > [routing\\_table\\_max\\_size](#) {fhicl::Name{"routing\_table\_max\_size"}, fhicl::Comment{"Maximum number of entries in the routing table"}, 1000}  
*"routing\_table\_max\_size" (Default: 1000): Maximum number of entries in the routing table*

### 6.62.1 Detailed Description

Configuration for Routing table reception

This configuration should be the same for all processes receiving routing tables from a given RoutingManager.

Definition at line 42 of file TableReceiver.hh.

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

- artdaq/artdaq/DAQrate/detail/TableReceiver.hh

## 6.63 artdaq::TokenReceiver::Config Struct Reference

Configuration of the [TokenReceiver](#). May be used for parameter validation.

```
#include <artdaq/DAQrate/detail/TokenReceiver.hh>
```

### Public Attributes

- fhicl::Atom< int > [routing\\_token\\_port](#) {fhicl::Name{"routing\_token\_port"}, fhicl::Comment{"Port to listen for routing tokens on"}, 35555}  
*"routing\_token\_port" (Default: 35555) : Port on which routing tokens will be received*

### 6.63.1 Detailed Description

Configuration of the [TokenReceiver](#). May be used for parameter validation.

Definition at line 31 of file TokenReceiver.hh.

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

- artdaq/artdaq/DAQrate/detail/TokenReceiver.hh

## 6.64 artdaq::FragmentBuffer::Config Struct Reference

Configuration of the [FragmentBuffer](#). May be used for parameter validation

```
#include <artdaq/DAQrate/FragmentBuffer.hh>
```

## Public Attributes

- fhicl::Atom< std::string > **generator\_type** {fhicl::Name{"generator"}, fhicl::Comment{"Name of the [FragmentBuffer](#) plugin to load"}}  
*"generator" (REQUIRED) Name of the [FragmentBuffer](#) plugin to load*
- fhicl::Atom  
 < Fragment::timestamp\_t > **request\_window\_offset** {fhicl::Name{"request\_window\_offset"}, fhicl::Comment{"Request messages contain a timestamp. For Window request mode, start the window this far before the timestamp in the request"}, 0}  
*"request\_window\_offset" (Default: 0) : Request messages contain a timestamp. For Window request mode, start the window this far before the timestamp in the request*
- fhicl::Atom  
 < Fragment::timestamp\_t > **request\_window\_width** {fhicl::Name{"request\_window\_width"}, fhicl::Comment{"For Window request mode, the window will be timestamp - offset to timestamp - offset + width"}, 0}  
*"request\_window\_width" (Default: 0) : For Window request mode, the window will be timestamp - offset to timestamp - offset + width*
- fhicl::Atom  
 < Fragment::timestamp\_t > **stale\_fragment\_timeout** {fhicl::Name{"stale\_fragment\_timeout"}, fhicl::Comment{"Fragments stored in the fragment generator which are older than the newest stored fragment by at least stale\_fragment\_timeout units of request timestamp ticks will get discarded"}, 0}  
*"stale\_fragment\_timeout" (Default: 0) : Fragments stored in the fragment generator which are older than the newest stored fragment by at least stale\_fragment\_timeout units of request timestamp ticks will get discarded (0 to disable)*
- fhicl::Atom< bool > **buffer\_mode\_keep\_latest** {fhicl::Name{"buffer\_mode\_keep\_latest"}, fhicl::Comment{"Keep the latest Fragment when running in Buffer mode, so that each response has at least one Fragment (Fragment will be discarded if new data arrives before next request)"}, false}  
*"buffer\_mode\_keep\_latest" (Default: false): Keep the latest Fragment when running in Buffer mode, so that each response has at least one Fragment (Fragment will be discarded if new data arrives before next request)*
- fhicl::Atom< Fragment::type\_t > **expected\_fragment\_type** {fhicl::Name{"expected\_fragment\_type"}, fhicl::Comment{"The type of Fragments this CFG will be generating. \"Empty\" will auto-detect type based on Fragments generated."}, Fragment::type\_t(Fragment::EmptyFragmentType)}  
*"expected\_fragment\_type" (Default: 231, EmptyFragmentType) : The type of Fragments this CFG will be generating. \"Empty\" will auto - detect type based on Fragments generated.*
- fhicl::Atom< bool > **request\_windows\_are\_unique** {fhicl::Name{"request\_windows\_are\_unique"}, fhicl::Comment{"Whether Fragments should be removed from the buffer when matched to a request window"}, true}  
*"request\_windows\_are\_unique" (Default: true) : Whether Fragments should be removed from the buffer when matched to a request window*
- fhicl::Atom< size\_t > **missing\_request\_window\_timeout\_us** {fhicl::Name{"missing\_request\_window\_timeout\_us"}, fhicl::Comment{"How long to track missing requests in the \"out - of - order Windows\" list"}, 5000000}  
*"missing\_request\_window\_timeout\_us" (Default: 5000000) : How long to track missing requests in the \"out-of-order Windows\" list*
- fhicl::Atom< size\_t > **window\_close\_timeout\_us** {fhicl::Name{"window\_close\_timeout\_us"}, fhicl::Comment{"How long to wait for the end of the data buffer to pass the end of a request window (measured from the time the request was received)"}, 2000000}  
*"window\_close\_timeout\_us" (Default: 2000000) : How long to wait for the end of the data buffer to pass the end of a request window(measured from the time the request was received)*
- fhicl::Atom< bool > **separate\_data\_thread** {fhicl::Name{"separate\_data\_thread"}, fhicl::Comment{"Whether data collection should proceed on its own thread. Required for all data request processing"}, false}  
*"separate\_data\_thread" (Default: false) : Whether data collection should proceed on its own thread.Required for all data request processing*
- fhicl::Atom< bool > **circular\_buffer\_mode** {fhicl::Name{"circular\_buffer\_mode"}, fhicl::Comment{"Whether the data buffer should be treated as a circular buffer on the input side (i.e. old fragments are automatically discarded when the buffer is full to always call getNext\_)."}, false}

*"circular\_buffer\_mode" (Default: false) : Whether the data buffer should be treated as a circular buffer on the input side (i.e. old fragments are automatically discarded when the buffer is full to always call getNext\_).*

- fhicl::Atom< size\_t > [sleep\\_on\\_no\\_data\\_us](#) {fhicl::Name{"sleep\_on\_no\_data\_us"}, fhicl::Comment{"How long to sleep after calling getNext\_ if no data is returned"}, 0}

*"sleep\_on\_no\_data\_us" (Default: 0 (no sleep)) : How long to sleep after calling getNext\_ if no data is returned*

- fhicl::Atom< int > [data\\_buffer\\_depth\\_fragments](#) {fhicl::Name{"data\_buffer\_depth\_fragments"}, fhicl::Comment{"The max fragments which can be stored before dropping occurs"}, 1000}

*"data\_buffer\_depth\_fragments" (Default: 1000) : The max fragments which can be stored before dropping occurs*

- fhicl::Atom< size\_t > [data\\_buffer\\_depth\\_mb](#) {fhicl::Name{"data\_buffer\_depth\_mb"}, fhicl::Comment{"The max cumulative size in megabytes of the fragments which can be stored before dropping occurs"}, 1000}

*"data\_buffer\_depth\_mb" (Default: 1000) : The max cumulative size in megabytes of the fragments which can be stored before dropping occurs*

- fhicl::Atom< bool > [separate\\_monitoring\\_thread](#) {fhicl::Name{"separate\_monitoring\_thread"}, fhicl::Comment{"Whether a thread that calls the checkHWStatus\_ method should be created"}, false}

*"separate\_monitoring\_thread" (Default: false) : Whether a thread that calls the checkHWStatus\_ method should be created*

- fhicl::Atom< int64\_t > [hardware\\_poll\\_interval\\_us](#) {fhicl::Name{"hardware\_poll\_interval\_us"}, fhicl::Comment{"If a separate monitoring thread is used, how often should it call checkHWStatus\_"}, 0}

*"hardware\_poll\_interval\_us" (Default: 0) : If a separate monitoring thread is used, how often should it call checkHWStatus\_*

- fhicl::Atom< int > [board\\_id](#) {fhicl::Name{"board\_id"}, fhicl::Comment{"The identification number for this [FragmentBuffer](#)"}}

*"board\_id" (REQUIRED) : The identification number for this [FragmentBuffer](#)*

- fhicl::Sequence  
< Fragment::fragment\_id\_t > [fragment\\_ids](#) {fhicl::Name{"fragment\_ids"}, fhicl::Comment{"A list of Fragment IDs created by this [FragmentBuffer](#)"}}

- fhicl::Atom< int > [fragment\\_id](#) {fhicl::Name{"fragment\_id"}, fhicl::Comment{"The Fragment ID created by this [FragmentBuffer](#)"}, -99}

- fhicl::Atom< int > [sleep\\_on\\_stop\\_us](#) {fhicl::Name{"sleep\_on\_stop\_us"}, fhicl::Comment{"How long to sleep before returning when stop transition is called"}, 0}

*"sleep\_on\_stop\_us" (Default: 0) : How long to sleep before returning when stop transition is called*

- fhicl::Atom< std::string > [request\\_mode](#) {fhicl::Name{"request\_mode"}, fhicl::Comment{"The mode by which the [FragmentBuffer](#) will process requests"}, "ignored"}

*"request\_mode" (Default: Ignored) : The mode by which the [FragmentBuffer](#) will process requests Ignored : Request messages are ignored. This is a "push" [FragmentBuffer](#) Single : The [FragmentBuffer](#) responds to each request with the latest Fragment it has received Buffer : The [FragmentBuffer](#) responds to each request with all Fragments it has received since the last request Window : The [FragmentBuffer](#) searches its data buffer for all Fragments whose timestamp falls within the request window SequenceID: The [FragmentBuffer](#) responds to each request with all Fragments that match the sequence ID in the request*

### 6.64.1 Detailed Description

Configuration of the [FragmentBuffer](#). May be used for parameter validation

Definition at line 95 of file [FragmentBuffer.hh](#).

### 6.64.2 Member Data Documentation

6.64.2.1 `fhicl::Atom<int> artdaq::FragmentBuffer::Config::fragment_id {fhicl::Name{"fragment_id"}, fhicl::Comment{"The Fragment ID created by this FragmentBuffer"}, -99}`

"fragment\_id" (Default: -99) : The Fragment ID created by this [FragmentBuffer](#) Note that only one of fragment\_ids and fragment\_id should be specified in the configuration

Definition at line 136 of file FragmentBuffer.hh.

6.64.2.2 `fhicl::Sequence<Fragment::fragment_id_t> artdaq::FragmentBuffer::Config::fragment_ids {fhicl::Name{"fragment_ids"}, fhicl::Comment{"A list of Fragment IDs created by this FragmentBuffer"}}`

"fragment\_ids" (Default: empty vector) : A list of Fragment IDs created by this [FragmentBuffer](#) Note that only one of fragment\_ids and fragment\_id should be specified in the configuration

Definition at line 133 of file FragmentBuffer.hh.

6.64.2.3 `fhicl::Atom<std::string> artdaq::FragmentBuffer::Config::request_mode {fhicl::Name{"request_mode"}, fhicl::Comment{"The mode by which the FragmentBuffer will process requests"}, "ignored"}`

"request\_mode" (Default: Ignored) : The mode by which the [FragmentBuffer](#) will process requests Ignored : Request messages are ignored. This is a "push" [FragmentBuffer](#) Single : The [FragmentBuffer](#) responds to each request with the latest Fragment it has received Buffer : The [FragmentBuffer](#) responds to each request with all Fragments it has received since the last request Window : The [FragmentBuffer](#) searches its data buffer for all Fragments whose timestamp falls within the request window SequenceID: The [FragmentBuffer](#) responds to each request with all Fragments that match the sequence ID in the request

Definition at line 147 of file FragmentBuffer.hh.

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

- `artdaq/artdaq/DAQrate/FragmentBuffer.hh`

## 6.65 artdaq::TokenSender::Config Struct Reference

Configuration for Routing token sending

```
#include <artdaq/DAQrate/detail/TokenSender.hh>
```

### Public Attributes

- `fhicl::Atom< bool > use_routing_manager {fhicl::Name{"use_routing_manager"}, fhicl::Comment{"True if using the Routing Manager"}, false}`  
*"use\_routing\_manager" (Default: false) : Whether to send tokens to a RoutingManager*
- `fhicl::Atom< int > routing_token_port {fhicl::Name{"routing_token_port"}, fhicl::Comment{"Port to send tokens on"}, 35555}`  
*"routing\_token\_port" (Default: 35555) : Port to send tokens on*
- `fhicl::Atom< std::string > routing_token_host {fhicl::Name{"routing_manager_hostname"}, fhicl::Comment{"Hostname or IP of RoutingManager"}, "localhost"}`  
*"routing\_manager\_hostname" (Default: "localhost") : Hostname or IP of RoutingManager*

### 6.65.1 Detailed Description

Configuration for Routing token sending

This configuration should be the same for all processes sending routing tokens to a given RoutingManager.

Definition at line 36 of file TokenSender.hh.

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

- artdaq/artdaq/DAQrate/detail/TokenSender.hh

## 6.66 artdaq::SharedMemoryEventManager::Config Struct Reference

Configuration of the [SharedMemoryEventManager](#). May be used for parameter validation

```
#include <artdaq/DAQrate/SharedMemoryEventManager.hh>
```

### Public Attributes

- fhicl::Atom< size\_t > [max\\_event\\_size\\_bytes](#) {fhicl::Name{"max\_event\_size\_bytes"}, fhicl::Comment{"Maximum event size (all Fragments), in bytes"}}
- fhicl::Atom< size\_t > [stale\\_buffer\\_timeout\\_usec](#) {fhicl::Name{"stale\_buffer\_timeout\_usec"}, fhicl::Comment{"Maximum amount of time elapsed before a buffer is marked as abandoned. Time is reset each time an operation is performed on the buffer."}, 5000000}
 

*"stale\_buffer\_timeout\_usec" (Default: event\_queue\_wait\_time \* 1, 000, 000) : Maximum amount of time elapsed before a buffer is marked as abandoned. Time is reset each time an operation is performed on the buffer.*
- fhicl::Atom< bool > [overwrite\\_mode](#) {fhicl::Name{"overwrite\_mode"}, fhicl::Comment{"Whether buffers are allowed to be overwritten when safe (state == Full or Reading)"}, false}
 

*"overwrite\_mode" (Default: false): Whether new data is allowed to overwrite buffers in the "Full" state*
- fhicl::Atom< bool > [restart\\_crashed\\_art\\_processes](#) {fhicl::Name{"restart\_crashed\_art\_processes"}, fhicl::Comment{"Whether to automatically restart art processes that fail for any reason"}, true}
 

*"restart\_crashed\_art\_processes" (Default: true) : Whether to automatically restart art processes that fail for any reason*
- fhicl::Atom< uint32\_t > [shared\\_memory\\_key](#) {fhicl::Name{"shared\_memory\_key"}, fhicl::Comment{"Key to use for shared memory access"}, 0xBEE70000 + getpid()}
 

*"shared\_memory\_key" (Default 0xBEE70000 + PID) : Key to use for shared memory access*
- fhicl::Atom< size\_t > [buffer\\_count](#) {fhicl::Name{"buffer\_count"}, fhicl::Comment{"Number of events in the Shared Memory (incomplete + pending art)"}}
 

*"buffer\_count" REQUIRED: Number of events in the Shared Memory(incomplete + pending art)*
- fhicl::Atom< size\_t > [max\\_fragment\\_size\\_bytes](#) {fhicl::Name{"max\_fragment\_size\_bytes"}, fhicl::Comment{"Maximum Fragment size, in bytes"}}
- fhicl::Atom< size\_t > [event\\_queue\\_wait\\_time](#) {fhicl::Name{"event\_queue\_wait\_time"}, fhicl::Comment{"Amount of time (in seconds) an event can exist in shared memory before being released to art. Used as input to default parameter of \"stale\_buffer\_timeout\_usec\"."}, 5}
 

*"event\_queue\_wait\_time" (Default: 5) : Amount of time(in seconds) an event can exist in shared memory before being released to art.Used as input to default parameter of "stale\_buffer\_timeout\_usec".*
- fhicl::Atom< bool > [broadcast\\_mode](#) {fhicl::Name{"broadcast\_mode"}, fhicl::Comment{"When true, buffers are not marked Empty when read, but return to Full state. Buffers are overwritten in order received."}, false}
 

*"broadcast\_mode" (Default: false) : When true, buffers are not marked Empty when read, but return to Full state. Buffers are overwritten in order received.*
- fhicl::Atom< size\_t > [art\\_analyzer\\_count](#) {fhicl::Name{"art\_analyzer\_count"}, fhicl::Comment{"Number of art processes to start"}, 1}



- "art\_analyzer\_count" (Default: 1) : Number of art processes to start*
- fhicl::Atom< size\_t > [expected\\_fragments\\_per\\_event](#) {fhicl::Name{"expected\_fragments\_per\_event"}, fhicl::Comment{"Number of Fragments to expect per event"}}
- "expected\_fragments\_per\_event" (REQUIRED) : Number of Fragments to expect per event*
- fhicl::Atom< int > [maximum\\_oversize\\_fragment\\_count](#) {fhicl::Name{"maximum\_oversize\_fragment\_count"}, fhicl::Comment{"Maximum number of over-size Fragments to drop before throwing an exception. Default is 1, which means to throw an exception if any over-size Fragments are dropped. Set to 0 to disable."}, 1}
- "maximum\_oversize\_fragment\_count" (Default: 1): Maximum number of over-size Fragments to drop before throwing an exception. Default is 1, which means to throw an exception if any over-size Fragments are dropped. Set to 0 to disable.*
- fhicl::Atom< bool > [update\\_run\\_ids\\_on\\_new\\_fragment](#) {fhicl::Name{"update\_run\_ids\_on\_new\_fragment"}, fhicl::Comment{"Whether the run and subrun ID of an event should be updated whenever a Fragment is added."}, true}
- "update\_run\_ids\_on\_new\_fragment" (Default: true) : Whether the run and subrun ID of an event should be updated whenever a Fragment is added.*
- fhicl::Atom< bool > [use\\_sequence\\_id\\_for\\_event\\_number](#) {fhicl::Name{"use\_sequence\_id\_for\_event\_number"}, fhicl::Comment{"Whether to use the artdaq Sequence ID (true) or the Timestamp (false) for art Event numbers"}, true}
- "use\_sequence\_id\_for\_event\_number" (Default: true): Whether to use the artdaq Sequence ID (true) or the Timestamp (false) for art Event numbers*
- fhicl::Atom< size\_t > [max\\_subrun\\_lookup\\_table\\_size](#) {fhicl::Name{"max\_subrun\_lookup\_table\_size"}, fhicl::Comment{"The maximum number of entries to store in the sequence ID-SubRun ID lookup table"}, 100}
- "max\_subrun\_lookup\_table\_size" (Default: 100): The maximum number of entries to store in the sequence ID-SubRun ID lookup table*
- fhicl::Atom< size\_t > [max\\_event\\_list\\_length](#) {fhicl::Name{"max\_event\_list\_length"}, fhicl::Comment{"The maximum number of entries to store in the released events list"}, 100}
- "max\_event\_list\_length" (Default: 100): The maximum number of entries to store in the released events list*
- fhicl::Atom< bool > [send\\_init\\_fragments](#) {fhicl::Name{"send\_init\_fragments"}, fhicl::Comment{"Whether Init Fragments are expected to be sent to art. If true, a Warning message is printed when an Init Fragment is requested but none are available."}, true}
- "send\_init\_fragments" (Default: true): Whether Init Fragments are expected to be sent to art. If true, a Warning message is printed when an Init Fragment is requested but none are available.*
- fhicl::Atom< int > [open\\_event\\_report\\_interval\\_ms](#) {fhicl::Name{"open\_event\_report\_interval\_ms"}, fhicl::Comment{"Interval at which an open event report should be written"}, -1}
- "open\_event\_report\_interval\_ms" (Default: -1): Interval at which an open event report should be written*
- fhicl::Atom< int > [fragment\\_broadcast\\_timeout\\_ms](#) {fhicl::Name{"fragment\_broadcast\_timeout\_ms"}, fhicl::Comment{"Amount of time broadcast fragments should live in the broadcast shared memory segment"}, 3000}
  - fhicl::Atom< std::string > [art\\_command\\_line](#) {fhicl::Name{"art\_command\_line"}, fhicl::Comment{"Command line used to start analysis processes. Supports two special sequences: #CONFIG\_FILE# will be replaced with the fhicl config file. #PROCESS\_INDEX# will be replaced by the index of the art process."}, "art -c #CONFIG\_FILE#"}
- "art\_command\_line" (Default: "art -c \#CONFIG\_FILE#\"): Command line used to start analysis processes. Supports two special sequences: #CONFIG\_FILE# will be replaced with the fhicl config file. #PROCESS\_INDEX# will be replaced by the index of the art process.*
- fhicl::Atom< size\_t > [art\\_index\\_offset](#) {fhicl::Name{"art\_index\_offset"}, fhicl::Comment{"Offset to add to art process index when replacing #PROCESS\_INDEX#"}, 0}
- "art\_index\_offset" (Default: 0): Offset to add to art process index when replacing #PROCESS\_INDEX#*
- fhicl::Atom< double > [minimum\\_art\\_lifetime\\_s](#) {fhicl::Name{"minimum\_art\_lifetime\_s"}, fhicl::Comment{"Amount of time that an art process should run to not be considered \"DOA\""}, 2.0}
- "minimum\_art\_lifetime\_s" (Default: 2 seconds): Amount of time that an art process should run to not be considered \"DOA\"*
- fhicl::Atom< size\_t > [expected\\_art\\_event\\_processing\\_time\\_us](#) {fhicl::Name{"expected\_art\_event\_processing\_time\_us"}, fhicl::Comment{"During shutdown, SMEM will wait for this amount of time while it is checking that the art threads are done reading buffers."}, 100000}



- fhicl::Atom< uint32\_t > [broadcast\\_shared\\_memory\\_key](#) {fhicl::Name{"broadcast\_shared\_memory\_key"}, fhicl::Comment{""}, 0xCCE70000 + getpid()}  
*"broadcast\_shared\_memory\_key" (Default: 0xCCE70000 + PID): Key to use for broadcast shared memory access*
- fhicl::Atom< size\_t > [broadcast\\_buffer\\_count](#) {fhicl::Name{"broadcast\_buffer\_count"}, fhicl::Comment{"Buffers in the broadcast shared memory segment"}, 10}  
*"broadcast\_buffer\_count" (Default: 10): Buffers in the broadcast shared memory segment*
- fhicl::Atom< size\_t > [broadcast\\_buffer\\_size](#) {fhicl::Name{"broadcast\_buffer\_size"}, fhicl::Comment{"Size of the buffers in the broadcast shared memory segment"}, 0x100000}  
*"broadcast\_buffer\_size" (Default: 0x100000): Size of the buffers in the broadcast shared memory segment*
- fhicl::Atom< bool > [use\\_art](#) {fhicl::Name{"use\_art"}, fhicl::Comment{"Whether to start and manage art threads (Sets art\_analyzer count to 0 and [overwrite\\_mode](#) to true when false)"}, true}  
*"use\_art" (Default: true): Whether to start and manage art threads (Sets art\_analyzer count to 0 and overwrite\_mode to true when false)*
- fhicl::Atom< bool > [manual\\_art](#) {fhicl::Name{"manual\_art"}, fhicl::Comment{"Prints the startup command line for the art process so that the user may (for example) run it in GDB or valgrind"}, false}  
*"manual\_art" (Default: false): Prints the startup command line for the art process so that the user may (for example) run it in GDB or valgrind*
- fhicl::TableFragment  
< [artdaq::RequestSender::Config](#) > [requestSenderConfig](#)  
*Configuration of the [RequestSender](#). See [artdaq::RequestSender::Config](#).*
- fhicl::OptionalTable  
< [artdaq::TokenSender::Config](#) > [tokenSenderConfig](#) {fhicl::Name{"routing\_token\_config"}, fhicl::Comment{"Configuration for the Routing [TokenSender](#)"}}  
*Configuration of the [TokenSender](#). See [artdaq::TokenSender::Config](#).*

## 6.66.1 Detailed Description

Configuration of the [SharedMemoryEventManager](#). May be used for parameter validation

Definition at line 115 of file SharedMemoryEventManager.hh.

## 6.66.2 Member Data Documentation

- 6.66.2.1 fhicl::Atom<size\_t> artdaq::SharedMemoryEventManager::Config::expected\_art\_event\_processing\_time\_us  
{fhicl::Name{"expected\_art\_event\_processing\_time\_us"}, fhicl::Comment{"During shutdown, SMEM will wait for this amount of time while it is checking that the art threads are done reading buffers."}, 100000}

"expected\_art\_event\_processing\_time\_us" (Default: 100000 us): During shutdown, SMEM will wait for this amount of time while it is checking that the art threads are done reading buffers. (TUNING: Should be slightly longer than the mean art processing time, but not so long that the Stop transition times out)

Definition at line 166 of file SharedMemoryEventManager.hh.

- 6.66.2.2 fhicl::Atom<int> artdaq::SharedMemoryEventManager::Config::fragment\_broadcast\_timeout\_ms  
{fhicl::Name{"fragment\_broadcast\_timeout\_ms"}, fhicl::Comment{"Amount of time broadcast fragments should live in the broadcast shared memory segment"}, 3000}

"fragment\_broadcast\_timeout\_ms" (Default: 3000): Amount of time broadcast fragments should live in the broadcast shared memory segment A "Broadcast shared memory segment" is used for all system-level fragments, such as Init, Start/End Run, Start/End Subrun and EndOfData

Definition at line 157 of file SharedMemoryEventManager.hh.

6.66.2.3 `fhicl::Atom<size_t> artdaq::SharedMemoryEventManager::Config::max_event_size_bytes`  
`{fhicl::Name{"max_event_size_bytes"}, fhicl::Comment{"Maximum event size (all Fragments), in bytes"}}`

"max\_event\_size\_bytes" REQUIRED: Maximum event size(all Fragments), in bytes Either max\_fragment\_size\_bytes or max\_event\_size\_bytes must be specified

Definition at line 119 of file SharedMemoryEventManager.hh.

6.66.2.4 `fhicl::Atom<size_t> artdaq::SharedMemoryEventManager::Config::max_fragment_size_bytes`  
`{fhicl::Name{"max_fragment_size_bytes"}, fhicl::Comment{" Maximum Fragment size, in bytes"}}`

"max\_fragment\_size\_bytes" REQUIRED: Maximum Fragment size, in bytes Either max\_fragment\_size\_bytes or max\_event\_size\_bytes must be specified

Definition at line 132 of file SharedMemoryEventManager.hh.

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

- artdaq/artdaq/DAQrate/SharedMemoryEventManager.hh

## 6.67 artdaq::CommanderInterface::Config Struct Reference

Configuration of the [CommanderInterface](#). May be used for parameter validation

```
#include <artdaq/ExternalComms/CommanderInterface.hh>
```

### Public Attributes

- `fhicl::Atom< int > id` `{fhicl::Name{"id"}, fhicl::Comment{"The unique ID associated with this Commander plugin. (ex. XMLRPC Port number)"}, 0}`  
*"id" (Default: 0): Integer ID number of this Commandable. May be constrained by plugin types(i.e.XMLRPC port number).*
- `fhicl::Atom< std::string > commanderPluginType` `{fhicl::Name{"commanderPluginType"}, fhicl::Comment{"String identifying the name of the CommanderInterface plugin to load"}}`  
*"commanderPluginType" (REQUIRED): The type of Commander plugin to load*

### 6.67.1 Detailed Description

Configuration of the [CommanderInterface](#). May be used for parameter validation

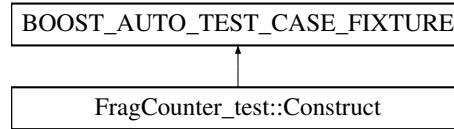
Definition at line 28 of file CommanderInterface.hh.

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

- artdaq/artdaq/ExternalComms/CommanderInterface.hh

## 6.68 FragCounter\_test::Construct Struct Reference

Inheritance diagram for FragCounter\_test::Construct:



## Public Member Functions

- void **test\_method** ()

### 6.68.1 Detailed Description

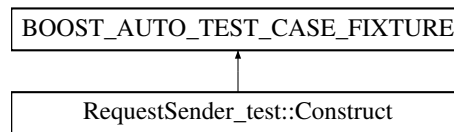
Definition at line 11 of file FragCounter\_t.cc.

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

- artdaq/test/DAQrate/FragCounter\_t.cc

## 6.69 RequestSender\_test::Construct Struct Reference

Inheritance diagram for RequestSender\_test::Construct:



## Public Member Functions

- void **test\_method** ()

### 6.69.1 Detailed Description

Definition at line 31 of file RequestSender\_t.cc.

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

- artdaq/test/DAQrate/RequestSender\_t.cc

## 6.70 FragCounter\_test::Construct\_id Struct Reference

### 6.70.1 Detailed Description

Definition at line 11 of file FragCounter\_t.cc.

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

- `artdaq/test/DAQrate/FragCounter_t.cc`

## 6.71 RequestSender\_test::Construct\_id Struct Reference

### 6.71.1 Detailed Description

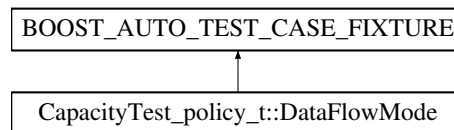
Definition at line 31 of file `RequestSender_t.cc`.

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

- `artdaq/test/DAQrate/RequestSender_t.cc`

## 6.72 CapacityTest\_policy\_t::DataFlowMode Struct Reference

Inheritance diagram for `CapacityTest_policy_t::DataFlowMode`:



### Public Member Functions

- void **test\_method** ()

### 6.72.1 Detailed Description

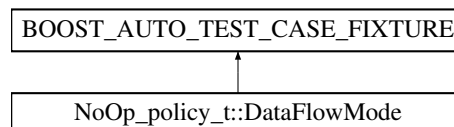
Definition at line 130 of file `CapacityTest_policy_t.cc`.

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

- `artdaq/test/RoutingPolicies/CapacityTest_policy_t.cc`

## 6.73 NoOp\_policy\_t::DataFlowMode Struct Reference

Inheritance diagram for `NoOp_policy_t::DataFlowMode`:



### Public Member Functions

- void **test\_method** ()

### 6.73.1 Detailed Description

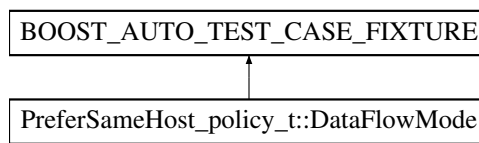
Definition at line 46 of file NoOp\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/NoOp\_policy\_t.cc

## 6.74 PreferSameHost\_policy\_t::DataFlowMode Struct Reference

Inheritance diagram for PreferSameHost\_policy\_t::DataFlowMode:



### Public Member Functions

- void **test\_method** ()

### 6.74.1 Detailed Description

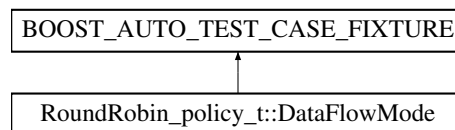
Definition at line 164 of file PreferSameHost\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/PreferSameHost\_policy\_t.cc

## 6.75 RoundRobin\_policy\_t::DataFlowMode Struct Reference

Inheritance diagram for RoundRobin\_policy\_t::DataFlowMode:



### Public Member Functions

- void **test\_method** ()

### 6.75.1 Detailed Description

Definition at line 192 of file RoundRobin\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/RoundRobin\_policy\_t.cc

## 6.76 CapacityTest\_policy\_t::DataFlowMode\_id Struct Reference

### 6.76.1 Detailed Description

Definition at line 130 of file CapacityTest\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/CapacityTest\_policy\_t.cc

## 6.77 NoOp\_policy\_t::DataFlowMode\_id Struct Reference

### 6.77.1 Detailed Description

Definition at line 46 of file NoOp\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/NoOp\_policy\_t.cc

## 6.78 PreferSameHost\_policy\_t::DataFlowMode\_id Struct Reference

### 6.78.1 Detailed Description

Definition at line 164 of file PreferSameHost\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/PreferSameHost\_policy\_t.cc

## 6.79 RoundRobin\_policy\_t::DataFlowMode\_id Struct Reference

### 6.79.1 Detailed Description

Definition at line 192 of file RoundRobin\_policy\_t.cc.

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

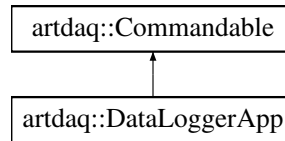
- artdaq/test/RoutingPolicies/RoundRobin\_policy\_t.cc

## 6.80 artdaq::DataLoggerApp Class Reference

[DataLoggerApp](#) is an [artdaq::Commandable](#) derived class which controls the [DataLoggerCore](#).

```
#include <artdaq/Application/DataLoggerApp.hh>
```

Inheritance diagram for artdaq::DataLoggerApp:



### Public Member Functions

- [DataLoggerApp](#) ()  
*DataLoggerApp Constructor.*
- [DataLoggerApp](#) ([DataLoggerApp](#) const &)=delete  
*Copy Constructor is Deleted.*
- virtual [~DataLoggerApp](#) ()=default  
*Default virtual destructor.*
- [DataLoggerApp](#) & [operator=](#) ([DataLoggerApp](#) const &)=delete  
*Copy Assignment operator is Deleted.*
- [DataLoggerApp](#) ([DataLoggerApp](#) &&)=delete  
*Move Constructor is deleted.*
- [DataLoggerApp](#) & [operator=](#) ([DataLoggerApp](#) &&)=delete  
*Move Assignment Operator is deleted.*
- bool [do\\_initialize](#) (fhicl::ParameterSet const &pset, uint64\_t, uint64\_t) override  
*Initialize the DataLoggerCore.*
- bool [do\\_start](#) (art::RunID id, uint64\_t, uint64\_t) override  
*Start the DataLoggerCore.*
- bool [do\\_stop](#) (uint64\_t, uint64\_t) override  
*Stop the DataLoggerCore.*
- bool [do\\_pause](#) (uint64\_t, uint64\_t) override  
*Pause the DataLoggerCore.*
- bool [do\\_resume](#) (uint64\_t, uint64\_t) override  
*Resume the DataLoggerCore.*
- bool [do\\_shutdown](#) (uint64\_t) override  
*Shutdown the DataLoggerCore.*
- bool [do\\_soft\\_initialize](#) (fhicl::ParameterSet const &, uint64\_t, uint64\_t) override  
*Soft-initialize the DataLoggerCore. No-Op.*
- bool [do\\_reinitialize](#) (fhicl::ParameterSet const &, uint64\_t, uint64\_t) override  
*Reinitialize the DataLoggerCore. No-Op.*
- std::string [report](#) (std::string const &which) const override  
*If which is "transition\_status", report the status of the last transition. Otherwise pass through to DataLoggerCore.*
- bool [do\\_add\\_config\\_archive\\_entry](#) (std::string const &, std::string const &) override  
*Add the specified configuration archive entry to the DataLoggerCore.*
- bool [do\\_clear\\_config\\_archive](#) () override  
*Clear the configuration archive list in the DataLoggerCore.*

## Additional Inherited Members

### 6.80.1 Detailed Description

[DataLoggerApp](#) is an [artdaq::Commandable](#) derived class which controls the [DataLoggerCore](#).

Definition at line 16 of file [DataLoggerApp.hh](#).

### 6.80.2 Member Function Documentation

**6.80.2.1** `bool artdaq::DataLoggerApp::do_add_config_archive_entry ( std::string const & key, std::string const & value )`  
`[override], [virtual]`

Add the specified configuration archive entry to the [DataLoggerCore](#).

#### Returns

Whether the command succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 151 of file [DataLoggerApp.cc](#).

**6.80.2.2** `bool artdaq::DataLoggerApp::do_clear_config_archive ( )` `[override], [virtual]`

Clear the configuration archive list in the [DataLoggerCore](#).

#### Returns

Whether the command succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 166 of file [DataLoggerApp.cc](#).

**6.80.2.3** `bool artdaq::DataLoggerApp::do_initialize ( fhicl::ParameterSet const & pset, uint64_t, uint64_t )` `[override], [virtual]`

Initialize the [DataLoggerCore](#).

#### Parameters

<i>pset</i>	ParameterSet used to initialize the <a href="#">DataLoggerCore</a>
-------------	--

#### Returns

Whether the initialize transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 19 of file [DataLoggerApp.cc](#).



6.80.2.4 `bool artdaq::DataLoggerApp::do_pause ( uint64_t, uint64_t ) [override],[virtual]`

Pause the [DataLoggerCore](#).

#### Returns

Whether the pause transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 68 of file DataLoggerApp.cc.

6.80.2.5 `bool artdaq::DataLoggerApp::do_reinitialize ( fhicl::ParameterSet const &, uint64_t, uint64_t ) [override],[virtual]`

Reinitialize the [DataLoggerCore](#). No-Op.

#### Returns

This function always returns true

Reimplemented from [artdaq::Commandable](#).

Definition at line 112 of file DataLoggerApp.cc.

6.80.2.6 `bool artdaq::DataLoggerApp::do_resume ( uint64_t, uint64_t ) [override],[virtual]`

Resume the [DataLoggerCore](#).

#### Returns

Whether the resume transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 81 of file DataLoggerApp.cc.

6.80.2.7 `bool artdaq::DataLoggerApp::do_shutdown ( uint64_t ) [override],[virtual]`

Shutdown the [DataLoggerCore](#).

#### Returns

Whether the shutdown transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 94 of file DataLoggerApp.cc.

6.80.2.8 `bool artdaq::DataLoggerApp::do_soft_initialize ( fhicl::ParameterSet const &, uint64_t, uint64_t ) [override],[virtual]`

Soft-initialize the [DataLoggerCore](#). No-Op.

**Returns**

This function always returns true

Reimplemented from [artdaq::Commandable](#).

Definition at line 107 of file DataLoggerApp.cc.

**6.80.2.9** `bool artdaq::DataLoggerApp::do_start ( art::RunID id, uint64_t, uint64_t ) [override],[virtual]`

Start the [DataLoggerCore](#).

**Parameters**

<i>id</i>	Run number of the new run
-----------	---------------------------

**Returns**

Whether the start transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 39 of file DataLoggerApp.cc.

**6.80.2.10** `bool artdaq::DataLoggerApp::do_stop ( uint64_t, uint64_t ) [override],[virtual]`

Stop the [DataLoggerCore](#).

**Returns**

Whether the stop transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 55 of file DataLoggerApp.cc.

**6.80.2.11** `DataLoggerApp& artdaq::DataLoggerApp::operator= ( DataLoggerApp const & ) [delete]`

Copy Assignment operator is Deleted.

**Returns**

[DataLoggerApp](#) copy

**6.80.2.12** `std::string artdaq::DataLoggerApp::report ( std::string const & which ) const [override],[virtual]`

If *which* is "transition\_status", report the status of the last transition. Otherwise pass through to [DataLoggerCore](#).

**Parameters**

<i>which</i>	What to report on
--------------	-------------------

**Returns**

Report string. Empty for unknown "which" parameter

Reimplemented from [artdaq::Commandable](#).

Definition at line 117 of file DataLoggerApp.cc.

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

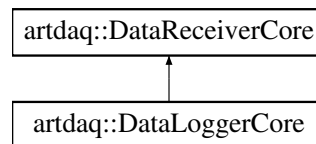
- artdaq/artdaq/Application/DataLoggerApp.hh
- artdaq/artdaq/Application/DataLoggerApp.cc

## 6.81 artdaq::DataLoggerCore Class Reference

[DataLoggerCore](#) implements the state machine for the DataLogger artdaq application. [DataLoggerCore](#) processes incoming events in one of three roles: Data Logger, Online Monitor, or Dispatcher.

```
#include <artdaq/Application/DataLoggerCore.hh>
```

Inheritance diagram for artdaq::DataLoggerCore:

**Public Member Functions**

- [DataLoggerCore](#) ()=default  
*DataLoggerCore Constructor.*
- [DataLoggerCore](#) ([DataLoggerCore](#) const &)=delete  
*Copy Constructor is deleted.*
- [~DataLoggerCore](#) ()
- [DataLoggerCore](#) & [operator=](#) ([DataLoggerCore](#) const &)=delete  
*Copy Assignment operator is deleted.*
- [DataLoggerCore](#) ([DataLoggerCore](#) &&)=delete  
*Move Constructor is deleted.*
- [DataLoggerCore](#) & [operator=](#) ([DataLoggerCore](#) &&)=delete  
*Move Assignment Operator is deleted.*
- bool [initialize](#) (fhicl::ParameterSet const &pset) override  
*Processes the initialize request.*

## Additional Inherited Members

### 6.81.1 Detailed Description

[DataLoggerCore](#) implements the state machine for the DataLogger artdaq application. [DataLoggerCore](#) processes incoming events in one of three roles: Data Logger, Online Monitor, or Dispatcher.

Definition at line 14 of file DataLoggerCore.hh.

### 6.81.2 Constructor & Destructor Documentation

#### 6.81.2.1 `artdaq::DataLoggerCore::~DataLoggerCore ( )` `[inline]`

Destructor.

Definition at line 30 of file DataLoggerCore.hh.

### 6.81.3 Member Function Documentation

#### 6.81.3.1 `bool artdaq::DataLoggerCore::initialize ( fhicl::ParameterSet const & pset )` `[override]`, `[virtual]`

Processes the initialize request.

Parameters

<i>pset</i>	ParameterSet used to configure the <a href="#">DataLoggerCore</a>
-------------	---

Returns

Whether the initialize attempt succeeded

```
* DataLoggerCore accepts the following Parameters:
* "daq" (REQUIRED): FHiCL table containing DAQ configuration
*   "DataLogger" (REQUIRED): FHiCL table containing DataLogger paramters
*     "expected_events_per_bunch" (REQUIRED): Number of events to collect before sending them to art
*     "enq_timeout" (Default: 5.0): Maximum amount of time to wait while enqueueing events to the ConcurrentQueue
*     "is_data_logger": True if the DataLogger is a Data Logger
*     "is_online_monitor": True if the DataLogger is an Online Monitor. is_data_logger takes precedence
*     "is_dispatcher": True if the DataLogger is a Dispatcher. is_data_logger and is_online_monitor take precedence
*     NOTE: At least ONE of these three parameters must be specified.
*     "inrun_rcv_timeout_usec" (Default: 100000): Amount of time to wait for new events while running
*     "endrun_rcv_timeout_usec" (Default: 20000000): Amount of time to wait for additional events at EndOfRun
*     "pause_rcv_timeout_usec" (Default: 3000000): Amount of time to wait for additional events at PauseRun
*     "onmon_event_prescale" (Default: 1): Only send 1/N events to art for online monitoring (requires is_data_logger)
*     "verbose" (Default: true): Whether to print transition messages
*     "metrics": FHiCL table containing configuration for MetricManager
*     "outputs" (REQUIRED): FHiCL table containing output parameters
*       "normalOutput" (REQUIRED): FHiCL table containing default output parameters
*       "fileName" (Default: ""): Name template of the output file. Used to determine output directory
*
```

Note that the "DataLogger" ParameterSet is also used to configure the EventStore. See that class' documentation for more information.

Implements [artdaq::DataReceiverCore](#).

Definition at line 9 of file DataLoggerCore.cc.

### 6.81.3.2 DataLoggerCore& artdaq::DataLoggerCore::operator= ( DataLoggerCore const & ) [delete]

Copy Assignment operator is deleted.

Returns

[DataLoggerCore](#) copy

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

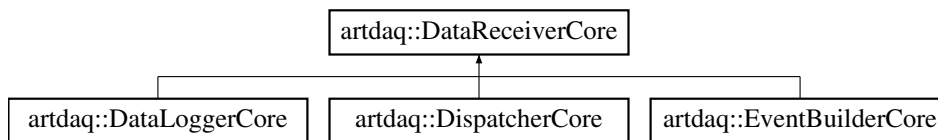
- artdaq/artdaq/Application/DataLoggerCore.hh
- artdaq/artdaq/Application/DataLoggerCore.cc

## 6.82 artdaq::DataReceiverCore Class Reference

[DataReceiverCore](#) implements the state machine for the DataReceiver artdaq application. [DataReceiverCore](#) receives Fragment objects from the [DataReceiverManager](#), and sends them to the EventStore.

```
#include <artdaq/Application/DataReceiverCore.hh>
```

Inheritance diagram for artdaq::DataReceiverCore:



### Public Member Functions

- [DataReceiverCore](#) ()  
*DataReceiverCore* Constructor.
- [DataReceiverCore](#) ([DataReceiverCore](#) const &)=delete  
*Copy Constructor is deleted.*
- virtual [~DataReceiverCore](#) ()
- [DataReceiverCore](#) & operator= ([DataReceiverCore](#) const &)=delete  
*Copy Assignment operator is deleted.*
- [DataReceiverCore](#) ([DataReceiverCore](#) &&)=delete  
*Move Constructor is deleted.*
- [DataReceiverCore](#) & operator= ([DataReceiverCore](#) &&)=delete  
*Move Assignment Operator is deleted.*
- virtual bool [initialize](#) (fhicl::ParameterSet const &pset)=0  
*Processes the initialize request.*
- bool [start](#) (art::RunID id)  
*Start the DataReceiverCore.*
- bool [stop](#) ()  
*Stops the DataReceiverCore.*
- bool [pause](#) ()  
*Pauses the DataReceiverCore.*

- bool [resume](#) ()  
*Resumes the [DataReceiverCore](#).*
- bool [shutdown](#) ()  
*Shuts Down the [DataReceiverCore](#).*
- bool [soft\\_initialize](#) (fhicl::ParameterSet const &pset)  
*Soft-Initializes the [DataReceiverCore](#). No-Op.*
- bool [reinitialize](#) (fhicl::ParameterSet const &pset)  
*Reinitializes the [DataReceiverCore](#).*
- bool [rollover\\_subrun](#) (uint64\_t boundary, uint32\_t subrun)  
*Rollover the subrun after the given event.*
- std::string [report](#) (std::string const &which) const  
*Send a report on a given run-time quantity.*
- bool [add\\_config\\_archive\\_entry](#) (std::string const &key, std::string const &value)  
*Add the specified key and value to the configuration archive list.*
- bool [clear\\_config\\_archive](#) ()  
*Clear the configuration archive list.*

## Protected Member Functions

- bool [initializeDataReceiver](#) (fhicl::ParameterSet const &pset, fhicl::ParameterSet const &data\_pset, fhicl::ParameterSet const &metric\_pset)  
*Initialize the [DataReceiverCore](#) (should be called from [initialize\(\)](#) overrides.*

## Protected Attributes

- std::unique\_ptr  
< [DataReceiverManager](#) > [receiver\\_ptr\\_](#)  
*Pointer to the [DataReceiverManager](#).*
- std::shared\_ptr  
< [SharedMemoryEventManager](#) > [event\\_store\\_ptr\\_](#)  
*Pointer to the [SharedMemoryEventManager](#).*
- std::atomic< bool > [stop\\_requested\\_](#)  
*Stop has been requested?*
- std::atomic< bool > [pause\\_requested\\_](#)  
*Pause has been requested?*
- std::atomic< bool > [run\\_is\\_paused\\_](#)  
*Pause has been successfully completed?*
- bool [verbose\\_](#)  
*Whether to log transition messages.*
- fhicl::ParameterSet [art\\_pset\\_](#)  
*ParameterSet sent to art process.*
- std::map< std::string,  
std::string > [config\\_archive\\_entries\\_](#)  
*Additional strings to archive as part of the art configuration.*

### 6.82.1 Detailed Description

[DataReceiverCore](#) implements the state machine for the DataReceiver artdaq application. [DataReceiverCore](#) receives Fragment objects from the [DataReceiverManager](#), and sends them to the EventStore.

Definition at line 23 of file DataReceiverCore.hh.

### 6.82.2 Constructor & Destructor Documentation

6.82.2.1 `artdaq::DataReceiverCore::~~DataReceiverCore ( ) [virtual]`

Destructor.

Definition at line 21 of file DataReceiverCore.cc.

### 6.82.3 Member Function Documentation

6.82.3.1 `bool artdaq::DataReceiverCore::add_config_archive_entry ( std::string const & key, std::string const & value ) [inline]`

Add the specified key and value to the configuration archive list.

Parameters

<i>key</i>	String key to be used
<i>value</i>	String value to be stored

Returns

This function will always return true

Definition at line 142 of file DataReceiverCore.hh.

6.82.3.2 `bool artdaq::DataReceiverCore::clear_config_archive ( ) [inline]`

Clear the configuration archive list.

Returns

True if archive is empty after clear operation

Definition at line 152 of file DataReceiverCore.hh.

6.82.3.3 `virtual bool artdaq::DataReceiverCore::initialize ( fhicl::ParameterSet const & pset ) [pure virtual]`

Processes the initialize request.

Parameters

<i>pset</i>	ParameterSet used to configure the <a href="#">DataReceiverCore</a>
-------------	---

### Returns

Whether the initialize attempt succeeded

```
* DataReceiverCore accepts the following Parameters:
* "daq" (REQUIRED): FHiCL table containing DAQ configuration
* "event_builder" (REQUIRED): FHiCL table containing Aggregator paramters
*   "fragment_count" (REQUIRED): Number of Fragment objects to collect before sending them to art
*   "inrun_rcv_timeout_usec" (Default: 100000): Amount of time to wait for new Fragment objects while running
*   "endrun_rcv_timeout_usec" (Default: 20000000): Amount of time to wait for additional Fragment objects at E
*   "pause_rcv_timeout_usec" (Default: 3000000): Amount of time to wait for additional Fragment objects at Pau
*   "verbose" (Default: true): Whether to print transition messages
* "metrics": FHiCL table containing configuration for MetricManager
*
```

Note that the "event\_builder" ParameterSet is also used to configure the [SharedMemoryEventManager](#). See that class' documentation for more information.

Implemented in [artdaq::DispatcherCore](#), [artdaq::DataLoggerCore](#), and [artdaq::EventBuilderCore](#).

**6.82.3.4** `bool artdaq::DataReceiverCore::initializeDataReceiver ( fhicl::ParameterSet const & pset, fhicl::ParameterSet const & data_pset, fhicl::ParameterSet const & metric_pset )` [protected]

Initialize the [DataReceiverCore](#) (should be called from [initialize\(\)](#) overrides).

### Parameters

<i>pset</i>	ParameterSet for art configuration
<i>data_pset</i>	ParameterSet for <a href="#">DataReceiverManager</a> and <a href="#">SharedMemoryEventManager</a> configuration
<i>metric_pset</i>	ParameterSet for MetricManager

### Returns

Whether the initialize succeeded

Definition at line 26 of file DataReceiverCore.cc.

**6.82.3.5** `DataReceiverCore& artdaq::DataReceiverCore::operator= ( DataReceiverCore const & )` [delete]

Copy Assignment operator is deleted.

### Returns

AggregatorCore copy

**6.82.3.6** `bool artdaq::DataReceiverCore::pause ( )`

Pauses the [DataReceiverCore](#).

### Returns

True if no exception

Definition at line 174 of file DataReceiverCore.cc.



6.82.3.7 `bool artdaq::DataReceiverCore::reinitialize ( fhicl::ParameterSet const & pset )`

Reinitializes the [DataReceiverCore](#).

## Parameters

<i>pset</i>	ParameterSet for configuring <a href="#">DataReceiverCore</a>
-------------	---

## Returns

True if no exception

Definition at line 236 of file DataReceiverCore.cc.

#### 6.82.3.8 `std::string artdaq::DataReceiverCore::report ( std::string const & which ) const`

Send a report on a given run-time quantity.

## Parameters

<i>which</i>	Which quantity to report
--------------	--------------------------

## Returns

A string containing the requested quantity.

report accepts the following values of "which": "event\_count": The number of events received, or -1 if not initialized  
"incomplete\_event\_count": The number of incomplete event bunches in the EventStore, or -1 if not initialized

Anything else will return the run number and an error message.

Definition at line 255 of file DataReceiverCore.cc.

#### 6.82.3.9 `bool artdaq::DataReceiverCore::resume ( )`

Resumes the [DataReceiverCore](#).

## Returns

True if no exception

Definition at line 183 of file DataReceiverCore.cc.

#### 6.82.3.10 `bool artdaq::DataReceiverCore::rollover_subrun ( uint64_t boundary, uint32_t subrun )`

Rollover the subrun after the given event.

## Parameters

<i>boundary</i>	Sequence ID of boundary
<i>subrun</i>	Subrun number of new subrun

## Returns

True event\_store\_ptr is valid

Definition at line 245 of file DataReceiverCore.cc.

### 6.82.3.11 bool artdaq::DataReceiverCore::shutdown ( )

Shuts Down the [DataReceiverCore](#).

#### Returns

If the shutdown was successful

Definition at line 194 of file DataReceiverCore.cc.

### 6.82.3.12 bool artdaq::DataReceiverCore::soft\_initialize ( fhicl::ParameterSet const & pset )

Soft-Initializes the [DataReceiverCore](#). No-Op.

#### Parameters

<i>pset</i>	ParameterSet for configuring <a href="#">DataReceiverCore</a>
-------------	---

#### Returns

Always returns true

Definition at line 228 of file DataReceiverCore.cc.

### 6.82.3.13 bool artdaq::DataReceiverCore::start ( art::RunID id )

Start the [DataReceiverCore](#).

#### Parameters

<i>id</i>	Run ID of the current run
-----------	---------------------------

#### Returns

True if no exception

Definition at line 96 of file DataReceiverCore.cc.

### 6.82.3.14 bool artdaq::DataReceiverCore::stop ( )

Stops the [DataReceiverCore](#).

#### Returns

True if no exception

Definition at line 130 of file DataReceiverCore.cc.

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

- artdaq/artdaq/Application/DataReceiverCore.hh
- artdaq/artdaq/Application/DataReceiverCore.cc

## 6.83 artdaq::DataReceiverManager Class Reference

Receives Fragment objects from one or more [DataSenderManager](#) instances using [TransferInterface](#) plugins DataReceiverManager runs a reception thread for each source, and can automatically suppress reception from sources which are going faster than the others.

```
#include <artdaq/DAQrate/DataReceiverManager.hh>
```

### Public Member Functions

- [DataReceiverManager](#) (const fhicl::ParameterSet &ps, std::shared\_ptr< [SharedMemoryEventManager](#) > shm)  
*DataReceiverManager Constructor.*
- virtual [~DataReceiverManager](#) ()  
*DataReceiverManager Destructor.*
- size\_t [count](#) () const  
*Return the count of Fragment objects received by this DataReceiverManager.*
- size\_t [slotCount](#) (size\_t rank) const  
*Get the count of Fragment objects received by this DataReceiverManager from a given source.*
- size\_t [byteCount](#) () const  
*Get the total size of all data recieved by this DataReceiverManager.*
- void [start\\_threads](#) ()  
*Start receiver threads for all enabled sources.*
- void [stop\\_threads](#) ()  
*Stop receiver threads.*
- std::set< int > [enabled\\_sources](#) () const  
*Get the list of enabled sources.*
- std::set< int > [running\\_sources](#) () const  
*Get the list of sources which are still receiving data.*
- std::shared\_ptr< [SharedMemoryEventManager](#) > [getSharedMemoryEventManager](#) () const  
*Get a handle to the SharedMemoryEventManager connected to this DataReceiverManager.*

### 6.83.1 Detailed Description

Receives Fragment objects from one or more [DataSenderManager](#) instances using [TransferInterface](#) plugins DataReceiverManager runs a reception thread for each source, and can automatically suppress reception from sources which are going faster than the others.

Definition at line 29 of file DataReceiverManager.hh.

### 6.83.2 Constructor & Destructor Documentation

- 6.83.2.1 [artdaq::DataReceiverManager::DataReceiverManager](#) ( const fhicl::ParameterSet & ps, std::shared\_ptr< [SharedMemoryEventManager](#) > shm ) [explicit]

[DataReceiverManager](#) Constructor.

## Parameters

<i>ps</i>	ParameterSet used to configure the <a href="#">DataReceiverManager</a>
<i>shm</i>	Pointer to <a href="#">SharedMemoryEventManager</a> instance (destination for received data)

\* DataReceiverManager accepts the following Parameters:  
 \* "auto\_suppression\_enabled" (Default: true): Whether to suppress a source that gets too far ahead  
 \* "max\_receive\_difference" (Default: 50): Threshold (in sequence ID) for suppressing a source  
 \* "receive\_timeout\_usec" (Default: 100000): The timeout for receive operations  
 \* "enabled\_sources" (OPTIONAL): List of sources which are enabled. If not specified, all sources are assumed enabled  
 \* "sources" (Default: blank table): FHiCL table containing TransferInterface configurations for each source.  
 \* NOTE: "source\_rank" MUST be specified (and unique) for each source!  
 \*

Definition at line 20 of file DataReceiverManager.cc.

## 6.83.3 Member Function Documentation

6.83.3.1 `size_t artdaq::DataReceiverManager::byteCount ( ) const [inline]`

Get the total size of all data recieved by this [DataReceiverManager](#).

## Returns

The total size of all data received by this [DataReceiverManager](#)

Definition at line 146 of file DataReceiverManager.hh.

6.83.3.2 `size_t artdaq::DataReceiverManager::count ( ) const [inline]`

Return the count of Fragment objects received by this [DataReceiverManager](#).

## Returns

The count of Fragment objects received by this [DataReceiverManager](#)

Definition at line 132 of file DataReceiverManager.hh.

6.83.3.3 `std::set< int > artdaq::DataReceiverManager::enabled_sources ( ) const`

Get the list of enabled sources.

## Returns

The list of enabled sources

Definition at line 219 of file DataReceiverManager.cc.

6.83.3.4 `std::shared_ptr<SharedMemoryEventManager> artdaq::DataReceiverManager::getSharedMemoryEventManager ( ) const [inline]`

Get a handle to the [SharedMemoryEventManager](#) connected to this [DataReceiverManager](#).

**Returns**

shared\_ptr to [SharedMemoryEventManager](#) instance

Definition at line 99 of file DataReceiverManager.hh.

### 6.83.3.5 std::set< int > artdaq::DataReceiverManager::running\_sources ( ) const

Get the list of sources which are still receiving data.

**Returns**

std::set containing ranks of sources which are still receiving data

Definition at line 232 of file DataReceiverManager.cc.

### 6.83.3.6 size\_t artdaq::DataReceiverManager::slotCount ( size\_t rank ) const [inline]

Get the count of Fragment objects received by this [DataReceiverManager](#) from a given source.

**Parameters**

<i>rank</i>	Source rank to get count for
-------------	------------------------------

**Returns**

The count of Fragment objects received by this [DataReceiverManager](#) from the source

Definition at line 139 of file DataReceiverManager.hh.

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

- artdaq/artdaq/DAQrate/DataReceiverManager.hh
- artdaq/artdaq/DAQrate/DataReceiverManager.cc

## 6.84 artdaq::DataSenderManager Class Reference

Sends Fragment objects using [TransferInterface](#) plugins. Uses Routing Tables if configured, otherwise will Round-Robin Fragments to the destinations.

```
#include <artdaq/DAQrate/DataSenderManager.hh>
```

**Classes**

- struct [Config](#)  
*Configuration of [DataSenderManager](#). May be used for parameter validation*
- struct [DestinationsConfig](#)  
*Configuration for transfers to destinations*

**Public Types**

- using [Parameters](#) = fhicl::WrappedTable< [Config](#) >  
*Used for ParameterSet validation (if desired)*

## Public Member Functions

- [DataSenderManager](#) (const fhicl::ParameterSet &ps)  
*DataSenderManager Constructor.*
- virtual [~DataSenderManager](#) ()  
*DataSenderManager Destructor.*
- std::pair< int,  
[TransferInterface::CopyStatus](#) > [sendFragment](#) (Fragment &&frag)  
*Send the given Fragment. Return the rank of the destination to which the Fragment was sent.*
- size\_t [count](#) () const  
*Return the count of Fragment objects sent by this DataSenderManager.*
- size\_t [slotCount](#) (size\_t rank) const  
*Get the count of Fragment objects sent by this DataSenderManager to a given destination.*
- size\_t [destinationCount](#) () const  
*Get the number of configured destinations.*
- std::set< int > [enabled\\_destinations](#) () const  
*Get the list of enabled destinations.*
- size\_t [GetRoutingTableEntryCount](#) () const  
*Gets the current size of the Routing Table, in case other parts of the system want to use this information.*
- size\_t [GetRemainingRoutingTableEntries](#) () const  
*Gets the number of sends remaining in the routing table, in case other parts of the system want to use this information.*
- void [StopSender](#) ()  
*Stop the DataSenderManager, aborting any sends in progress.*
- void [RemoveRoutingTableEntry](#) (Fragment::sequence\_id\_t seq)  
*Remove the given sequence ID from the routing table and sent\_count lists.*
- size\_t [GetSentSequenceIDCount](#) (Fragment::sequence\_id\_t seq)  
*Get the number of Fragments sent with a given Sequence ID.*

### 6.84.1 Detailed Description

Sends Fragment objects using [TransferInterface](#) plugins. Uses Routing Tables if configured, otherwise will Round-Robin Fragments to the destinations.

Definition at line 35 of file DataSenderManager.hh.

### 6.84.2 Constructor & Destructor Documentation

#### 6.84.2.1 artdaq::DataSenderManager::DataSenderManager ( const fhicl::ParameterSet & ps ) [explicit]

[DataSenderManager](#) Constructor.

Parameters

<i>ps</i>	ParameterSet used to configure the <a href="#">DataSenderManager</a> . See <a href="#">artdaq::DataSenderManager::Config</a>
-----------	--

Definition at line 16 of file DataSenderManager.cc.

### 6.84.3 Member Function Documentation

#### 6.84.3.1 `size_t artdaq::DataSenderManager::count ( ) const [inline]`

Return the count of Fragment objects sent by this DataSenderManager.

##### Returns

The count of Fragment objects sent by this [DataSenderManager](#)

Definition at line 175 of file DataSenderManager.hh.

#### 6.84.3.2 `size_t artdaq::DataSenderManager::destinationCount ( ) const [inline]`

Get the number of configured destinations.

##### Returns

The number of configured destinations

Definition at line 107 of file DataSenderManager.hh.

#### 6.84.3.3 `std::set<int> artdaq::DataSenderManager::enabled_destinations ( ) const [inline]`

Get the list of enabled destinations.

##### Returns

The list of enabled destination ranks

Definition at line 113 of file DataSenderManager.hh.

#### 6.84.3.4 `size_t artdaq::DataSenderManager::GetRemainingRoutingTableEntries ( ) const`

Gets the number of sends remaining in the routing table, in case other parts of the system want to use this information.

##### Returns

The number of sends remaining in the routing table

Definition at line 136 of file DataSenderManager.cc.

#### 6.84.3.5 `size_t artdaq::DataSenderManager::GetRoutingTableEntryCount ( ) const`

Gets the current size of the Routing Table, in case other parts of the system want to use this information.

##### Returns

The current size of the Routing Table.

Definition at line 131 of file DataSenderManager.cc.

#### 6.84.3.6 `size_t artdaq::DataSenderManager::GetSentSequenceIDCount ( Fragment::sequence_id_t seq )`

Get the number of Fragments sent with a given Sequence ID.



## Parameters

<i>seq</i>	Sequence ID to query
------------	----------------------

## Returns

The number of Fragments sent with a given Sequence ID

Definition at line 182 of file DataSenderManager.cc.

#### 6.84.3.7 void artdaq::DataSenderManager::RemoveRoutingTableEntry ( Fragment::sequence\_id\_t seq )

Remove the given sequence ID from the routing table and sent\_count lists.

## Parameters

<i>seq</i>	Sequence ID to remove
------------	-----------------------

Definition at line 170 of file DataSenderManager.cc.

#### 6.84.3.8 std::pair< int, artdaq::TransferInterface::CopyStatus > artdaq::DataSenderManager::sendFragment ( Fragment && frag )

Send the given Fragment. Return the rank of the destination to which the Fragment was sent.

## Parameters

<i>frag</i>	Fragment to sent
-------------	------------------

## Returns

Pair containing Rank of destination for Fragment and the CopyStatus from the send call

Definition at line 192 of file DataSenderManager.cc.

#### 6.84.3.9 size\_t artdaq::DataSenderManager::slotCount ( size\_t rank ) const [inline]

Get the count of Fragment objects sent by this [DataSenderManager](#) to a given destination.

## Parameters

<i>rank</i>	Destination rank to get count for
-------------	-----------------------------------

## Returns

The count of Fragment objects sent by this [DataSenderManager](#) to the destination

Definition at line 182 of file DataSenderManager.hh.

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

- artdaq/artdaq/DAQrate/DataSenderManager.hh
- artdaq/artdaq/DAQrate/DataSenderManager.cc

## 6.85 artdaq::DataSenderManager::DestinationsConfig Struct Reference

Configuration for transfers to destinations

```
#include <artdaq/DAQrate/DataSenderManager.hh>
```

### Public Attributes

- fhicl::OptionalTable  
 < [artdaq::TransferInterface::Config](#) > dest {fhicl::Name{"d1"}, fhicl::Comment{"Configuration for transfer to destination"}}

*Example Configuration for transfer to destination. See [artdaq::TransferInterface::Config](#).*

### 6.85.1 Detailed Description

Configuration for transfers to destinations

Definition at line 41 of file DataSenderManager.hh.

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

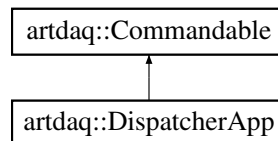
- artdaq/artdaq/DAQrate/DataSenderManager.hh

## 6.86 artdaq::DispatcherApp Class Reference

[DispatcherApp](#) is an [artdaq::Commandable](#) derived class which controls the [DispatcherCore](#).

```
#include <artdaq/Application/DispatcherApp.hh>
```

Inheritance diagram for [artdaq::DispatcherApp](#):



### Public Member Functions

- [DispatcherApp](#) ()  
*DispatcherApp Constructor.*
- [DispatcherApp](#) ([DispatcherApp](#) const &)=delete  
*Copy Constructor is Deleted.*
- virtual ~[DispatcherApp](#) ()=default  
*Default virtual destructor.*
- [DispatcherApp](#) & operator= ([DispatcherApp](#) const &)=delete  
*Copy Assignment operator is Deleted.*
- [DispatcherApp](#) ([DispatcherApp](#) &&)=delete  
*Move Constructor is deleted.*

- [DispatcherApp](#) & [operator=](#) ([DispatcherApp](#) &&)=delete  
*Move Assignment Operator is deleted.*
- bool [do\\_initialize](#) (fhicl::ParameterSet const &pset, uint64\_t, uint64\_t) override  
*Initialize the [DispatcherCore](#).*
- bool [do\\_start](#) (art::RunID id, uint64\_t, uint64\_t) override  
*Start the [DispatcherCore](#).*
- bool [do\\_stop](#) (uint64\_t, uint64\_t) override  
*Stop the [DispatcherCore](#).*
- bool [do\\_pause](#) (uint64\_t, uint64\_t) override  
*Pause the [DispatcherCore](#).*
- bool [do\\_resume](#) (uint64\_t, uint64\_t) override  
*Resume the [DispatcherCore](#).*
- bool [do\\_shutdown](#) (uint64\_t) override  
*Shutdown the [DispatcherCore](#).*
- bool [do\\_soft\\_initialize](#) (fhicl::ParameterSet const &, uint64\_t, uint64\_t) override  
*Soft-initialize the [DispatcherCore](#). No-Op.*
- bool [do\\_reinitialize](#) (fhicl::ParameterSet const &, uint64\_t, uint64\_t) override  
*Reinitialize the [DispatcherCore](#). No-Op.*
- std::string [report](#) (std::string const &which) const override  
*If which is "transition\_status", report the status of the last transition. Otherwise pass through to [DispatcherCore](#).*
- std::string [register\\_monitor](#) (fhicl::ParameterSet const &info) override  
*Register an art Online Monitor to the [DispatcherCore](#).*
- std::string [unregister\\_monitor](#) (std::string const &label) override  
*Remove an art Online Monitor from the [DispatcherCore](#).*

## Additional Inherited Members

### 6.86.1 Detailed Description

[DispatcherApp](#) is an [artdaq::Commandable](#) derived class which controls the [DispatcherCore](#).

Definition at line 18 of file DispatcherApp.hh.

### 6.86.2 Member Function Documentation

6.86.2.1 bool artdaq::DispatcherApp::do\_initialize ( fhicl::ParameterSet const & pset, uint64\_t, uint64\_t ) [override],  
[virtual]

Initialize the [DispatcherCore](#).

Parameters

<i>pset</i>	ParameterSet used to initialize the <a href="#">DispatcherCore</a>
-------------	--

Returns

Whether the initialize transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 19 of file DispatcherApp.cc.

**6.86.2.2** `bool artdaq::DispatcherApp::do_pause ( uint64_t, uint64_t ) [override],[virtual]`

Pause the [DispatcherCore](#).

**Returns**

Whether the pause transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 68 of file DispatcherApp.cc.

**6.86.2.3** `bool artdaq::DispatcherApp::do_reinitialize ( fhicl::ParameterSet const &, uint64_t, uint64_t ) [override],[virtual]`

Reinitialize the [DispatcherCore](#). No-Op.

**Returns**

This function always returns true

Reimplemented from [artdaq::Commandable](#).

Definition at line 111 of file DispatcherApp.cc.

**6.86.2.4** `bool artdaq::DispatcherApp::do_resume ( uint64_t, uint64_t ) [override],[virtual]`

Resume the [DispatcherCore](#).

**Returns**

Whether the resume transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 80 of file DispatcherApp.cc.

**6.86.2.5** `bool artdaq::DispatcherApp::do_shutdown ( uint64_t ) [override],[virtual]`

Shutdown the [DispatcherCore](#).

**Returns**

Whether the shutdown transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 93 of file DispatcherApp.cc.

**6.86.2.6** `bool artdaq::DispatcherApp::do_soft_initialize ( fhicl::ParameterSet const &, uint64_t, uint64_t ) [override],[virtual]`

Soft-initialize the [DispatcherCore](#). No-Op.

#### Returns

This function always returns true

Reimplemented from [artdaq::Commandable](#).

Definition at line 106 of file DispatcherApp.cc.

**6.86.2.7** `bool artdaq::DispatcherApp::do_start ( art::RunID id, uint64_t, uint64_t )` `[override]`, `[virtual]`

Start the [DispatcherCore](#).

#### Parameters

<i>id</i>	Run number of the new run
-----------	---------------------------

#### Returns

Whether the start transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 39 of file DispatcherApp.cc.

**6.86.2.8** `bool artdaq::DispatcherApp::do_stop ( uint64_t, uint64_t )` `[override]`, `[virtual]`

Stop the [DispatcherCore](#).

#### Returns

Whether the stop transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 55 of file DispatcherApp.cc.

**6.86.2.9** `DispatcherApp& artdaq::DispatcherApp::operator= ( DispatcherApp const & )` `[delete]`

Copy Assignment operator is Deleted.

#### Returns

[DispatcherApp](#) copy

**6.86.2.10** `std::string artdaq::DispatcherApp::register_monitor ( fhicl::ParameterSet const & info )` `[override]`, `[virtual]`

Register an art Online Monitor to the [DispatcherCore](#).

## Parameters

<i>info</i>	ParameterSet containing information about the monitor
-------------	---

## Returns

String detailing result status

Reimplemented from [artdaq::Commandable](#).

Definition at line 150 of file DispatcherApp.cc.

**6.86.2.11** `std::string artdaq::DispatcherApp::report ( std::string const & which ) const` `[override],[virtual]`

If *which* is "transition\_status", report the status of the last transition. Otherwise pass through to [DispatcherCore](#).

## Parameters

<i>which</i>	What to report on
--------------	-------------------

## Returns

Report string. Empty for unknown "which" parameter

Reimplemented from [artdaq::Commandable](#).

Definition at line 116 of file DispatcherApp.cc.

**6.86.2.12** `std::string artdaq::DispatcherApp::unregister_monitor ( std::string const & label )` `[override],[virtual]`

Remove an art Online Monitor from the [DispatcherCore](#).

## Parameters

<i>label</i>	Name of the monitor to remove
--------------	-------------------------------

## Returns

String detailing result status

Reimplemented from [artdaq::Commandable](#).

Definition at line 174 of file DispatcherApp.cc.

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

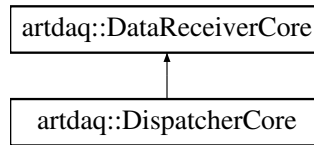
- artdaq/artdaq/Application/DispatcherApp.hh
- artdaq/artdaq/Application/DispatcherApp.cc

## 6.87 artdaq::DispatcherCore Class Reference

[DispatcherCore](#) implements the state machine for the Dispatcher artdaq application. [DispatcherCore](#) processes incoming events in one of three roles: Data Logger, Online Monitor, or Dispatcher.

```
#include <artdaq/Application/DispatcherCore.hh>
```

Inheritance diagram for artdaq::DispatcherCore:



## Public Member Functions

- [DispatcherCore](#) ()=default  
*DispatcherCore Constructor.*
- [DispatcherCore](#) ([DispatcherCore](#) const &)=delete  
*Copy Constructor is deleted.*
- [~DispatcherCore](#) ()
- [DispatcherCore](#) & [operator=](#) ([DispatcherCore](#) const &)=delete  
*Copy Assignment operator is deleted.*
- [DispatcherCore](#) ([DispatcherCore](#) &&)=delete  
*Move Constructor is deleted.*
- [DispatcherCore](#) & [operator=](#) ([DispatcherCore](#) &&)=delete  
*Move Assignment Operator is deleted.*
- bool [initialize](#) (fhicl::ParameterSet const &pset) override  
*Processes the initialize request.*
- std::string [register\\_monitor](#) (fhicl::ParameterSet const &pset)  
*Create a new [TransferInterface](#) instance using the given configuration.*
- std::string [unregister\\_monitor](#) (std::string const &label)  
*Delete the [TransferInterface](#) having the given unique label.*

## Additional Inherited Members

### 6.87.1 Detailed Description

[DispatcherCore](#) implements the state machine for the Dispatcher artdaq application. [DispatcherCore](#) processes incoming events in one of three roles: Data Logger, Online Monitor, or Dispatcher.

Definition at line 20 of file DispatcherCore.hh.

### 6.87.2 Constructor & Destructor Documentation

#### 6.87.2.1 artdaq::DispatcherCore::~~DispatcherCore ( ) [inline]

Destructor.

Definition at line 36 of file DispatcherCore.hh.

### 6.87.3 Member Function Documentation

6.87.3.1 `bool artdaq::DispatcherCore::initialize ( fhicl::ParameterSet const & pset )` `[override],[virtual]`

Processes the initialize request.



## Parameters

<i>pset</i>	ParameterSet used to configure the <a href="#">DispatcherCore</a>
-------------	---

## Returns

Whether the initialize attempt succeeded

```
* DispatcherCore accepts the following Parameters:
* "daq" (REQUIRED): FHiCL table containing DAQ configuration
* "Dispatcher" (REQUIRED): FHiCL table containing Dispatcher parameters
*   "expected_events_per_bunch" (REQUIRED): Number of events to collect before sending them to art
*   "enq_timeout" (Default: 5.0): Maximum amount of time to wait while enqueueing events to the ConcurrentQueue
*   "is_data_logger": True if the Dispatcher is a Data Logger
*   "is_online_monitor": True if the Dispatcher is an Online Monitor. is_data_logger takes precedence
*   "is_dispatcher": True if the Dispatcher is a Dispatcher. is_data_logger and is_online_monitor take precedence
*   NOTE: At least ONE of these three parameters must be specified.
*   "inrun_rcv_timeout_usec" (Default: 100000): Amount of time to wait for new events while running
*   "endrun_rcv_timeout_usec" (Default: 20000000): Amount of time to wait for additional events at EndOfRun
*   "pause_rcv_timeout_usec" (Default: 3000000): Amount of time to wait for additional events at PauseRun
*   "onmon_event_prescale" (Default: 1): Only send 1/N events to art for online monitoring (requires is_data_logger)
*   "verbose" (Default: true): Whether to print transition messages
* "metrics": FHiCL table containing configuration for MetricManager
* "outputs" (REQUIRED): FHiCL table containing output parameters
* "normalOutput" (REQUIRED): FHiCL table containing default output parameters
*   "fileName" (Default: ""): Name template of the output file. Used to determine output directory
*
```

Note that the "Dispatcher" ParameterSet is also used to configure the EventStore. See that class' documentation for more information.

Implements [artdaq::DataReceiverCore](#).

Definition at line 25 of file DispatcherCore.cc.

### 6.87.3.2 DispatcherCore& artdaq::DispatcherCore::operator=( DispatcherCore const & ) [delete]

Copy Assignment operator is deleted.

## Returns

[DispatcherCore](#) copy

### 6.87.3.3 std::string artdaq::DispatcherCore::register\_monitor ( fhicl::ParameterSet const & pset )

Create a new [TransferInterface](#) instance using the given configuration.

## Parameters

<i>pset</i>	ParameterSet used to configure the <a href="#">TransferInterface</a>
-------------	--

## Returns

String detailing any errors encountered or "Success"

See [TransferInterface](#) for details on the expected configuration

Definition at line 94 of file DispatcherCore.cc.

6.87.3.4 `std::string artdaq::DispatcherCore::unregister_monitor ( std::string const & label )`

Delete the [TransferInterface](#) having the given unique label.

## Parameters

<i>label</i>	Label of the <a href="#">TransferInterface</a> to delete
--------------	--

## Returns

String detailing any errors encountered or "Success"

Definition at line 160 of file DispatcherCore.cc.

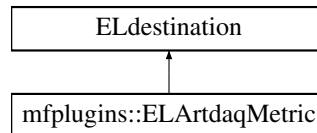
The documentation for this class was generated from the following files:

- artdaq/artdaq/Application/DispatcherCore.hh
- artdaq/artdaq/Application/DispatcherCore.cc

## 6.88 mfplugins::ELArtdaqMetric Class Reference

Message Facility destination which logs messages to a TRACE buffer

Inheritance diagram for mfplugins::ELArtdaqMetric:



## Classes

- struct [Config](#)  
Configuration Parameters for [ELArtdaqMetric](#).

## Public Types

- using [Parameters](#) = fhicl::WrappedTable< [Config](#) >  
Used for *ParameterSet* validation.

## Public Member Functions

- [ELArtdaqMetric](#) ([Parameters](#) const &pset)  
*ELArtdaqMetric* Constructor
- void [fillPrefix](#) (std::ostream &o, const ErrorObj &msg) override  
Fill the "Prefix" portion of the message.
- void [fillUsrMsg](#) (std::ostream &o, const ErrorObj &msg) override  
Fill the "User Message" portion of the message.
- void [fillSuffix](#) (std::ostream &, const ErrorObj &) override  
Fill the "Suffix" portion of the message (*Unused*)
- void [routePayload](#) (const std::ostream &o, const ErrorObj &msg) override  
Serialize a *MessageFacility* message to the output.

### 6.88.1 Detailed Description

Message Facility destination which logs messages to a TRACE buffer

Definition at line 23 of file ArtdaqMetric\_mfPlugin.cc.

### 6.88.2 Constructor & Destructor Documentation

#### 6.88.2.1 mfplugins::ELArtdaqMetric::ELArtdaqMetric ( Parameters const & *pset* )

[ELArtdaqMetric](#) Constructor

Parameters

<i>pset</i>	ParameterSet used to configure <a href="#">ELArtdaqMetric</a>
-------------	---

Definition at line 103 of file ArtdaqMetric\_mfPlugin.cc.

### 6.88.3 Member Function Documentation

#### 6.88.3.1 void mfplugins::ELArtdaqMetric::fillPrefix ( std::ostream & *o*, const ErrorObj & *msg* ) [override]

Fill the "Prefix" portion of the message.

Parameters

<i>o</i>	Output stringstream
<i>msg</i>	MessageFacility object containing header information

Definition at line 118 of file ArtdaqMetric\_mfPlugin.cc.

#### 6.88.3.2 void mfplugins::ELArtdaqMetric::fillUsrMsg ( std::ostream & *o*, const ErrorObj & *msg* ) [override]

Fill the "User Message" portion of the message.

Parameters

<i>o</i>	Output stringstream
<i>msg</i>	MessageFacility object containing header information

Definition at line 125 of file ArtdaqMetric\_mfPlugin.cc.

#### 6.88.3.3 void mfplugins::ELArtdaqMetric::routePayload ( const std::ostream & *o*, const ErrorObj & *msg* ) [override]

Serialize a MessageFacility message to the output.

Parameters

<i>o</i>	Stringstream object containing message data
<i>msg</i>	MessageFacility object containing header information

Definition at line 154 of file ArtdaqMetric\_mfPlugin.cc.

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

- artdaq/artdaq/ArtModules/ArtdaqMetric\_mfPlugin.cc

## 6.89 anonymous\_namespace{xmlrpc\_commander.cc}::env\_wrap Class Reference

Wrapper for XMLRPC environment construction/destruction

### Public Attributes

- xmlrpc\_env [env\\_c](#)  
*XMLRPC Environment.*

### 6.89.1 Detailed Description

Wrapper for XMLRPC environment construction/destruction

Definition at line 271 of file xmlrpc\_commander.cc.

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

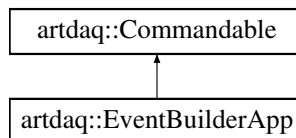
- artdaq/artdaq/ExternalComms/xmlrpc\_commander.cc

## 6.90 artdaq::EventBuilderApp Class Reference

[EventBuilderApp](#) is an [artdaq::Commandable](#) derived class which controls the [EventBuilderCore](#).

```
#include <artdaq/Application/EventBuilderApp.hh>
```

Inheritance diagram for artdaq::EventBuilderApp:



### Public Member Functions

- [EventBuilderApp](#) ()  
*EventBuilderApp Constructor.*
- [EventBuilderApp](#) ([EventBuilderApp](#) const &)=delete  
*Copy Constructor is deleted.*
- virtual [~EventBuilderApp](#) ()=default  
*Default Destructor.*
- [EventBuilderApp](#) & operator= ([EventBuilderApp](#) const &)=delete  
*Copy Assignment Operator is deleted.*
- [EventBuilderApp](#) ([EventBuilderApp](#) &&)=delete  
*Move Constructor is deleted.*
- [EventBuilderApp](#) & operator= ([EventBuilderApp](#) &&)=delete  
*Move Assignment Operator is deleted.*
- bool [do\\_initialize](#) (fhicl::ParameterSet const &pset, uint64\_t, uint64\_t) override

Initialize the [EventBuilderCore](#).

- bool [do\\_start](#) (art::RunID id, uint64\_t, uint64\_t) override

Start the [EventBuilderCore](#).

- bool [do\\_stop](#) (uint64\_t, uint64\_t) override

Stop the [EventBuilderCore](#).

- bool [do\\_pause](#) (uint64\_t, uint64\_t) override

Pause the [EventBuilderCore](#).

- bool [do\\_resume](#) (uint64\_t, uint64\_t) override

Resume the [EventBuilderCore](#).

- bool [do\\_shutdown](#) (uint64\_t) override

Shutdown the [EventBuilderCore](#).

- bool [do\\_soft\\_initialize](#) (fhicl::ParameterSet const &pset, uint64\_t, uint64\_t) override

Soft-Initialize the [EventBuilderCore](#).

- bool [do\\_reinitialize](#) (fhicl::ParameterSet const &pset, uint64\_t, uint64\_t) override

Reinitialize the [EventBuilderCore](#).

- bool [do\\_rollover\\_subrun](#) (uint64\_t boundary, uint32\_t subrunNum) override

Rollover the subrun after the given event.

- void [BootedEnter](#) () override

Action taken upon entering the "Booted" state.

- std::string [report](#) (std::string const &which) const override

If which is "transition\_status", report the status of the last transition. Otherwise pass through to [EventBuilderCore](#).

- bool [do\\_add\\_config\\_archive\\_entry](#) (std::string const &, std::string const &) override

Add the specified configuration archive entry to the [EventBuilderCore](#).

- bool [do\\_clear\\_config\\_archive](#) () override

Clear the configuration archive list in the [EventBuilderCore](#).

## Additional Inherited Members

### 6.90.1 Detailed Description

[EventBuilderApp](#) is an [artdaq::Commandable](#) derived class which controls the [EventBuilderCore](#).

Definition at line 16 of file EventBuilderApp.hh.

### 6.90.2 Member Function Documentation

#### 6.90.2.1 void artdaq::EventBuilderApp::BootedEnter ( ) [override],[virtual]

Action taken upon entering the "Booted" state.

This is a No-Op

Reimplemented from [artdaq::Commandable](#).

Definition at line 151 of file EventBuilderApp.cc.

6.90.2.2 `bool artdaq::EventBuilderApp::do_add_config_archive_entry ( std::string const & key, std::string const & value )`  
`[override], [virtual]`

Add the specified configuration archive entry to the [EventBuilderCore](#).

#### Returns

Whether the command succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 196 of file EventBuilderApp.cc.

6.90.2.3 `bool artdaq::EventBuilderApp::do_clear_config_archive ( )` `[override], [virtual]`

Clear the configuration archive list in the [EventBuilderCore](#).

#### Returns

Whether the command succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 211 of file EventBuilderApp.cc.

6.90.2.4 `bool artdaq::EventBuilderApp::do_initialize ( fhicl::ParameterSet const & pset, uint64_t, uint64_t )` `[override], [virtual]`

Initialize the [EventBuilderCore](#).

#### Parameters

<i>pset</i>	ParameterSet used to configure the <a href="#">EventBuilderCore</a>
-------------	---

#### Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 20 of file EventBuilderApp.cc.

6.90.2.5 `bool artdaq::EventBuilderApp::do_pause ( uint64_t, uint64_t )` `[override], [virtual]`

Pause the [EventBuilderCore](#).

#### Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 73 of file EventBuilderApp.cc.

6.90.2.6 `bool artdaq::EventBuilderApp::do_reinitialize ( fhicl::ParameterSet const & pset, uint64_t, uint64_t )` `[override], [virtual]`

Reinitialize the [EventBuilderCore](#).

## Parameters

<i>pset</i>	ParameterSet used to configure the <a href="#">EventBuilderCore</a>
-------------	---

## Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 124 of file EventBuilderApp.cc.

6.90.2.7 `bool artdaq::EventBuilderApp::do_resume ( uint64_t, uint64_t )` `[override],[virtual]`

Resume the [EventBuilderCore](#).

## Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 86 of file EventBuilderApp.cc.

6.90.2.8 `bool artdaq::EventBuilderApp::do_rollover_subrun ( uint64_t boundary, uint32_t subrunNum )` `[override],[virtual]`

Rollover the subrun after the given event.

## Parameters

<i>boundary</i>	Sequence ID of boundary
<i>subrunNum</i>	Number of new subrun

## Returns

True event\_store\_ptr is valid

Reimplemented from [artdaq::Commandable](#).

Definition at line 137 of file EventBuilderApp.cc.

6.90.2.9 `bool artdaq::EventBuilderApp::do_shutdown ( uint64_t )` `[override],[virtual]`

Shutdown the [EventBuilderCore](#).

## Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 99 of file EventBuilderApp.cc.

6.90.2.10 `bool artdaq::EventBuilderApp::do_soft_initialize ( fhicl::ParameterSet const & pset, uint64_t, uint64_t )` `[override],[virtual]`

Soft-Initialize the [EventBuilderCore](#).



## Parameters

<i>pset</i>	ParameterSet used to configure the <a href="#">EventBuilderCore</a>
-------------	---

## Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 111 of file EventBuilderApp.cc.

6.90.2.11 `bool artdaq::EventBuilderApp::do_start ( art::RunID id, uint64_t, uint64_t ) [override],[virtual]`

Start the [EventBuilderCore](#).

## Parameters

<i>id</i>	Run ID of new run
-----------	-------------------

## Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 45 of file EventBuilderApp.cc.

6.90.2.12 `bool artdaq::EventBuilderApp::do_stop ( uint64_t, uint64_t ) [override],[virtual]`

Stop the [EventBuilderCore](#).

## Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 61 of file EventBuilderApp.cc.

6.90.2.13 `EventBuilderApp& artdaq::EventBuilderApp::operator= ( EventBuilderApp const & ) [delete]`

Copy Assignment Operator is deleted.

## Returns

[EventBuilderApp](#) copy

6.90.2.14 `std::string artdaq::EventBuilderApp::report ( std::string const & which ) const [override],[virtual]`

If which is "transition\_status", report the status of the last transition. Otherwise pass through to [EventBuilderCore](#).

## Parameters

<i>which</i>	What to report on
--------------	-------------------

## Returns

Report string. Empty for unknown "which" parameter

Reimplemented from [artdaq::Commandable](#).

Definition at line 162 of file EventBuilderApp.cc.

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

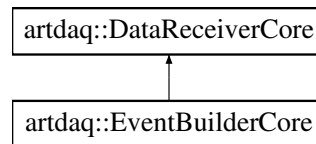
- `artdaq/artdaq/Application/EventBuilderApp.hh`
- `artdaq/artdaq/Application/EventBuilderApp.cc`

## 6.91 artdaq::EventBuilderCore Class Reference

[EventBuilderCore](#) implements the state machine for the EventBuilder artdaq application. [EventBuilderCore](#) receives Fragment objects from the [DataReceiverManager](#), and sends them to the EventStore.

```
#include <artdaq/Application/EventBuilderCore.hh>
```

Inheritance diagram for `artdaq::EventBuilderCore`:



### Public Member Functions

- [EventBuilderCore](#) ()=default  
*EventBuilderCore Constructor.*
- [EventBuilderCore](#) ([EventBuilderCore](#) const &)=delete  
*Copy Constructor is deleted.*
- [~EventBuilderCore](#) ()
- [EventBuilderCore](#) & operator= ([EventBuilderCore](#) const &)=delete  
*Copy Assignment operator is deleted.*
- [EventBuilderCore](#) ([EventBuilderCore](#) &&)=delete  
*Move Constructor is deleted.*
- [EventBuilderCore](#) & operator= ([EventBuilderCore](#) &&)=delete  
*Move Assignment Operator is deleted.*
- bool [initialize](#) (fhicl::ParameterSet const &pset) override  
*Processes the initialize request.*

## Additional Inherited Members

### 6.91.1 Detailed Description

[EventBuilderCore](#) implements the state machine for the EventBuilder artdaq application. [EventBuilderCore](#) receives Fragment objects from the [DataReceiverManager](#), and sends them to the EventStore.

Definition at line 18 of file EventBuilderCore.hh.

### 6.91.2 Constructor & Destructor Documentation

#### 6.91.2.1 `artdaq::EventBuilderCore::~EventBuilderCore ( )` `[inline]`

Destructor.

Definition at line 34 of file EventBuilderCore.hh.

### 6.91.3 Member Function Documentation

#### 6.91.3.1 `bool artdaq::EventBuilderCore::initialize ( fhicl::ParameterSet const & pset )` `[override]`, `[virtual]`

Processes the initialize request.

Parameters

<i>pset</i>	ParameterSet used to configure the <a href="#">EventBuilderCore</a>
-------------	---

Returns

Whether the initialize attempt succeeded

```
* EventBuilderCore accepts the following Parameters:
* "daq" (REQUIRED): FHiCL table containing DAQ configuration
* "event_builder" (REQUIRED): FHiCL table containing Aggregator parameters
*   "fragment_count" (REQUIRED): Number of Fragment objects to collect before sending them to art
*   "inrun_recv_timeout_usec" (Default: 100000): Amount of time to wait for new Fragment objects while running
*   "endrun_recv_timeout_usec" (Default: 20000000): Amount of time to wait for additional Fragment objects at E
*   "pause_recv_timeout_usec" (Default: 3000000): Amount of time to wait for additional Fragment objects at Pau
*   "verbose" (Default: true): Whether to print transition messages
* "metrics": FHiCL table containing configuration for MetricManager
*
```

Note that the "event\_builder" ParameterSet is also used to configure the [SharedMemoryEventManager](#). See that class' documentation for more information.

Implements [artdaq::DataReceiverCore](#).

Definition at line 11 of file EventBuilderCore.cc.

#### 6.91.3.2 `EventBuilderCore& artdaq::EventBuilderCore::operator= ( EventBuilderCore const & )` `[delete]`

Copy Assignment operator is deleted.

**Returns**

AggregatorCore copy

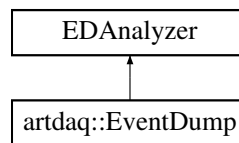
The documentation for this class was generated from the following files:

- artdaq/artdaq/Application/EventBuilderCore.hh
- artdaq/artdaq/Application/EventBuilderCore.cc

## 6.92 artdaq::EventDump Class Reference

Write Event information to the console.

Inheritance diagram for artdaq::EventDump:

**Public Member Functions**

- [EventDump](#) (fhicl::ParameterSet const &pset)  
*EventDump Constructor.*
- [~EventDump](#) () override=default  
*Default virtual Destructor.*
- void [analyze](#) (art::Event const &e) override  
*This method is called for each art::Event in a file or run.*

### 6.92.1 Detailed Description

Write Event information to the console.

Definition at line 33 of file EventDump\_module.cc.

### 6.92.2 Constructor & Destructor Documentation

#### 6.92.2.1 artdaq::EventDump::EventDump ( fhicl::ParameterSet const & pset ) [explicit]

[EventDump](#) Constructor.

**Parameters**

<i>pset</i>	ParameterSet used to configure <a href="#">EventDump</a>
-------------	--

```

* EventDump accepts the following Parameters:
* "raw_data_label" (Default: "daq"): The label used to store artdaq data
* "verbosity" (Default: 0): verboseness level
*

```

Definition at line 72 of file EventDump\_module.cc.

### 6.92.3 Member Function Documentation

#### 6.92.3.1 void artdaq::EventDump::analyze ( art::Event const & e ) [override]

This method is called for each art::Event in a file or run.

Parameters

<i>e</i>	The art::Event to analyze
----------	---------------------------

This module simply prints the event number, and art by default prints the products found in the event.

Definition at line 77 of file EventDump\_module.cc.

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

- artdaq/artdaq/ArtModules/EventDump\_module.cc

## 6.93 art::RootDAQOut::Config::FileNameSubstitution Struct Reference

### Public Attributes

- fhicl::Atom< string > **targetString** {fhicl::Name("targetString")}
- fhicl::Sequence< fhicl::Table  
< NewSubStringForApp > > **replacementList** {fhicl::Name("replacementList")}

#### 6.93.1 Detailed Description

Definition at line 140 of file RootDAQOut\_module.cc.

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

- artdaq/artdaq/ArtModules/RootDAQOutput-s124/RootDAQOut\_module.cc

## 6.94 art::FiltersConfig Struct Reference

```
#include <artdaq/ArtModules/detail/ArtConfig.hh>
```

#### 6.94.1 Detailed Description

**Todo** Fill in artdaq-provided Filter modules

Definition at line 63 of file ArtConfig.hh.

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

- artdaq/artdaq/ArtModules/detail/ArtConfig.hh

## 6.95 artdaq::detail::FragCounter Class Reference

Keep track of the count of Fragments received from a set of sources.

```
#include <artdaq/DAQrate/detail/FragCounter.hh>
```

## Public Member Functions

- [FragCounter](#) ()  
*Default Constructor.*
- void [incSlot](#) (size\_t slot)  
*Increment the given slot by one.*
- void [incSlot](#) (size\_t slot, size\_t inc)  
*Increment the given slot by the given amount.*
- void [setSlot](#) (size\_t slot, size\_t val)  
*Set the given slot to the given value.*
- size\_t [nSlots](#) () const  
*Get the number of slots this [FragCounter](#) instance is tracking.*
- size\_t [count](#) () const  
*Get the total number of Fragments received.*
- size\_t [slotCount](#) (size\_t slot) const  
*Get the current count for the requested slot.*
- size\_t [minCount](#) () const  
*Get the minimum slot count.*
- size\_t [operator\[\]](#) (size\_t slot) const  
*Get the current count for the requested slot.*

### 6.95.1 Detailed Description

Keep track of the count of Fragments received from a set of sources.

Definition at line 18 of file `FragCounter.hh`.

### 6.95.2 Member Function Documentation

6.95.2.1 `size_t artdaq::detail::FragCounter::count ( ) const` `[inline]`

Get the total number of Fragments received.

#### Returns

The total number of Fragments received

Definition at line 120 of file `FragCounter.hh`.

6.95.2.2 `void artdaq::detail::FragCounter::incSlot ( size_t slot )` `[inline]`

Increment the given slot by one.

#### Parameters

<i>slot</i>	Slot to increment
-------------	-------------------

Definition at line 89 of file `FragCounter.hh`.

6.95.2.3 `void artdaq::detail::FragCounter::incSlot ( size_t slot, size_t inc )` `[inline]`

Increment the given slot by the given amount.

## Parameters

<i>slot</i>	Slot to increment
<i>inc</i>	Amount to increment

Definition at line 96 of file FragCounter.hh.

#### 6.95.2.4 `size_t artdaq::detail::FragCounter::minCount ( ) const [inline]`

Get the minimum slot count.

## Returns

The minimum slot count

Definition at line 141 of file FragCounter.hh.

#### 6.95.2.5 `size_t artdaq::detail::FragCounter::nSlots ( ) const [inline]`

Get the number of slots this [FragCounter](#) instance is tracking.

## Returns

The number of slots in this [FragCounter](#) instance

Definition at line 112 of file FragCounter.hh.

#### 6.95.2.6 `size_t artdaq::detail::FragCounter::operator[] ( size_t slot ) const [inline]`

Get the current count for the requested slot.

## Parameters

<i>slot</i>	Slot to get count for
-------------	-----------------------

## Returns

The current count for the requested slot

Definition at line 76 of file FragCounter.hh.

#### 6.95.2.7 `void artdaq::detail::FragCounter::setSlot ( size_t slot, size_t val ) [inline]`

Set the given slot to the given value.

## Parameters

<i>slot</i>	Slot to set
<i>val</i>	Value to set

Definition at line 104 of file FragCounter.hh.

#### 6.95.2.8 `size_t artdaq::detail::FragCounter::slotCount ( size_t slot ) const [inline]`

Get the current count for the requested slot.

## Parameters

<i>slot</i>	Slot to get count for
-------------	-----------------------

## Returns

The current count for the requested slot

Definition at line 133 of file FragCounter.hh.

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

- artdaq/artdaq/DAQrate/detail/FragCounter.hh

## 6.96 artdaq::FragmentBuffer Class Reference

[FragmentBuffer](#) is a FragmentGenerator-derived abstract class that defines the interface for a FragmentGenerator designed as a state machine with start, stop, etc., transition commands.

```
#include <artdaq/DAQrate/FragmentBuffer.hh>
```

## Classes

- struct [Config](#)  
*Configuration of the [FragmentBuffer](#). May be used for parameter validation*

## Public Types

- using [Parameters](#) = fhicl::WrappedTable< [Config](#) >  
*Used for ParameterSet validation (if desired)*

## Public Member Functions

- [FragmentBuffer](#) (const fhicl::ParameterSet &ps)  
*[FragmentBuffer](#) Constructor.*
- virtual [~FragmentBuffer](#) ()  
*[FragmentBuffer](#) Destructor.*
- void [AddFragmentsToBuffer](#) (FragmentPtrs frags)  
*Add Fragments to the [FragmentBuffer](#).*
- void [Stop](#) ()  
*Inform the [FragmentBuffer](#) that it should stop.*
- void [Reset](#) (bool stop)  
*Reset the [FragmentBuffer](#) (flushes all Fragments from buffers)*
- void [applyRequestsIgnoredMode](#) (artdaq::FragmentPtrs &frags)  
*Create fragments using data buffer for request mode Ignored. Precondition: dataBufferMutex\_ and request\_mutex\_ are locked*
- void [applyRequestsSingleMode](#) (artdaq::FragmentPtrs &frags)  
*Create fragments using data buffer for request mode Single. Precondition: dataBufferMutex\_ and request\_mutex\_ are locked*



- void [applyRequestsBufferMode](#) (artdaq::FragmentPtrs &frags)  
*Create fragments using data buffer for request mode Buffer. Precondition: dataBufferMutex\_ and request\_mutex\_ are locked*
- void [applyRequestsWindowMode](#) (artdaq::FragmentPtrs &frags)  
*Create fragments using data buffer for request mode Window. Precondition: dataBufferMutex\_ and request\_mutex\_ are locked*
- void [applyRequestsSequenceIDMode](#) (artdaq::FragmentPtrs &frags)  
*Create fragments using data buffer for request mode SequenceID. Precondition: dataBufferMutex\_ and request\_mutex\_ are locked*
- void [applyRequestsWindowMode\\_CheckAndFillDataBuffer](#) (artdaq::FragmentPtrs &frags, artdaq::Fragment::fragment\_id\_t id, artdaq::Fragment::sequence\_id\_t seq, artdaq::Fragment::timestamp\_t ts)
- bool [applyRequests](#) (FragmentPtrs &frags)  
*See if any requests have been received, and add the corresponding data Fragment objects to the output list.*
- bool [sendEmptyFragment](#) (FragmentPtrs &frags, size\_t sequenceId, Fragment::fragment\_id\_t fragmentId, std::string desc)  
*Send an EmptyFragmentType Fragment.*
- void [sendEmptyFragments](#) (FragmentPtrs &frags, std::map< Fragment::sequence\_id\_t, Fragment::timestamp\_t > &requests)  
*This function is for Buffered and Single request modes, as they can only respond to one data request at a time. If the request message seqID > ev\_counter, simply send empties until they're equal.*
- void [checkSentWindows](#) (Fragment::sequence\_id\_t seq)  
*Check the windows\_sent\_ooo\_map for sequence IDs that may be removed.*
- bool [waitForDataBufferReady](#) (Fragment::fragment\_id\_t id)  
*Wait for the data buffer to drain (dataBufferIsTooLarge returns false), periodically reporting status.*
- bool [dataBufferIsTooLarge](#) (Fragment::fragment\_id\_t id)  
*Test the configured constraints on the data buffer.*
- void [getDataBufferStats](#) (Fragment::fragment\_id\_t id)  
*Calculate the size of the dataBuffer and report appropriate metrics.*
- void [getDataBuffersStats](#) ()  
*Calculate the size of all dataBuffers and report appropriate metrics.*
- std::string [getStatReport](#) ()  
*report statistics as a string*
- void [checkDataBuffer](#) (Fragment::fragment\_id\_t id)  
*Perform data buffer pruning operations for the given buffer. If the RequestMode is Single, removes all but the latest Fragment from the data buffer.*
- void [checkDataBuffers](#) ()  
*Perform data buffer pruning operations for all buffers.*
- std::map  
  < Fragment::sequence\_id\_t,  
  std::chrono::steady\_clock::time\_point > [GetSentWindowList](#) (Fragment::fragment\_id\_t id)  
*Get the map of Window-mode requests fulfilled by this Fragment Generator for the given Fragment ID.*
- std::vector  
  < Fragment::fragment\_id\_t > [fragmentIDs](#) ()  
*Get the list of Fragment IDs handled by this [FragmentBuffer](#).*
- [RequestMode request\\_mode](#) () const  
*Get the current request mode of the [FragmentBuffer](#)*
- void [SetRequestBuffer](#) (std::shared\_ptr< [RequestBuffer](#) > buffer)  
*Set the pointer to the [RequestBuffer](#) used to retrieve requests.*
- artdaq::Fragment::sequence\_id\_t [GetNextSequenceID](#) () const  
*Get the next sequence ID expected by this [FragmentBuffer](#). This is used to track sent windows and missed requests.*

## Protected Member Functions

- `artdaq::Fragment::fragment_id_t fragment_id ()` const  
*Get the Fragment ID of this Fragment generator.*
- `bool check_stop ()`  
*Routine used by `applyRequests` to make sure that all outstanding requests have been fulfilled before returning.*
- `std::string printMode_ ()`  
*Return the string representation of the current `RequestMode`.*
- `size_t dataBufferFragmentCount_ ()`  
*Get the total number of Fragments in all data buffers.*

### 6.96.1 Detailed Description

[FragmentBuffer](#) is a `FragmentGenerator`-derived abstract class that defines the interface for a `FragmentGenerator` designed as a state machine with start, stop, etc., transition commands.

Users of classes derived from [FragmentBuffer](#) will call these transitions via the publically defined `StartCmd()`, `StopCmd()`, etc.; these public functions contain functionality considered properly universal to all `FragmentBuffer`-derived classes, including calls to private virtual functions meant to be overridden in derived classes. The same applies to this class's implementation of the `FragmentGenerator::getNext()` pure virtual function, which is declared final (i.e., non-overridable in derived classes) and which itself calls a pure virtual `getNext_()` function to be implemented in derived classes.

State-machine related interface functions will be called only from a single thread. `getNext()` will be called only from a single thread. The thread from which state-machine interfaces functions are called may be a different thread from the one that calls `getNext()`.

John F., 3/24/14

After some discussion with Kurt, [FragmentBuffer](#) has been updated such that it now contains a member vector `fragment_ids_`; if "fragment\_id" is set in the FHiCL document controlling a class derived from [FragmentBuffer](#), `fragment_ids_` will be booked as a length-1 vector, and the value in this vector will be returned by `fragment_id()`. `fragment_id()` will throw an exception if the length of the vector isn't 1. If "fragment\_ids" is set in the FHiCL document, then `fragment_ids_` is filled with the values in the list which "fragment\_ids" refers to, otherwise it is set to the empty vector (this is what should happen if the user sets the "fragment\_id" variable in the FHiCL document, otherwise exceptions will end up thrown due to the logical conflict). If neither "fragment\_id" nor "fragment\_ids" is set in the FHiCL document, writers of classes derived from this one will be expected to override the virtual `fragmentIDs()` function with their own code (the [CompositeDriver](#) class is an example of this)

Definition at line 89 of file `FragmentBuffer.hh`.

### 6.96.2 Constructor & Destructor Documentation

#### 6.96.2.1 `artdaq::FragmentBuffer::FragmentBuffer ( const fhicl::ParameterSet & ps )` `[explicit]`

[FragmentBuffer](#) Constructor.

Parameters

<code>ps</code>	ParameterSet used to configure <a href="#">FragmentBuffer</a> . See <a href="#">artdaq::FragmentBuffer::Config</a> .
-----------------	--

Definition at line 41 of file `FragmentBuffer.cc`.

### 6.96.2.2 artdaq::FragmentBuffer::~~FragmentBuffer ( ) [virtual]

[FragmentBuffer](#) Destructor.

Joins all threads before returning

Definition at line 117 of file FragmentBuffer.cc.

## 6.96.3 Member Function Documentation

### 6.96.3.1 void artdaq::FragmentBuffer::AddFragmentsToBuffer ( FragmentPtrs *frags* )

Add Fragments to the [FragmentBuffer](#).

Parameters

<i>frags</i>	Fragments to add
--------------	------------------

Definition at line 143 of file FragmentBuffer.cc.

### 6.96.3.2 bool artdaq::FragmentBuffer::applyRequests ( FragmentPtrs & *frags* )

See if any requests have been received, and add the corresponding data Fragment objects to the output list.

Parameters

out	<i>frags</i>	list of FragmentPtr objects ready for transmission
-----	--------------	--

Returns

True if not stopped

Definition at line 829 of file FragmentBuffer.cc.

### 6.96.3.3 void artdaq::FragmentBuffer::applyRequestsBufferMode ( artdaq::FragmentPtrs & *frags* )

Create fragments using data buffer for request mode Buffer. Precondition: dataBufferMutex\_ and request\_mutex\_ are locked

Parameters

<i>frags</i>	Ouput fragments
--------------	-----------------

Definition at line 523 of file FragmentBuffer.cc.

### 6.96.3.4 void artdaq::FragmentBuffer::applyRequestsIgnoredMode ( artdaq::FragmentPtrs & *frags* )

Create fragments using data buffer for request mode Ignored. Precondition: dataBufferMutex\_ and request\_mutex\_ are locked

Parameters

<i>frags</i>	Ouput fragments
--------------	-----------------

Definition at line 460 of file FragmentBuffer.cc.

#### 6.96.3.5 void artdaq::FragmentBuffer::applyRequestsSequenceIDMode ( artdaq::FragmentPtrs & frags )

Create fragments using data buffer for request mode SequenceID. Precondition: dataBufferMutex\_ and request\_mutex\_ are locked

Parameters

<i>frags</i>	Ouput fragments
--------------	-----------------

Definition at line 771 of file FragmentBuffer.cc.

#### 6.96.3.6 void artdaq::FragmentBuffer::applyRequestsSingleMode ( artdaq::FragmentPtrs & frags )

Create fragments using data buffer for request mode Single. Precondition: dataBufferMutex\_ and request\_mutex\_ are locked

Parameters

<i>frags</i>	Ouput fragments
--------------	-----------------

Definition at line 480 of file FragmentBuffer.cc.

#### 6.96.3.7 void artdaq::FragmentBuffer::applyRequestsWindowMode ( artdaq::FragmentPtrs & frags )

Create fragments using data buffer for request mode Window. Precondition: dataBufferMutex\_ and request\_mutex\_ are locked

Parameters

<i>frags</i>	Ouput fragments
--------------	-----------------

Definition at line 716 of file FragmentBuffer.cc.

#### 6.96.3.8 void artdaq::FragmentBuffer::applyRequestsWindowMode\_CheckAndFillDataBuffer ( artdaq::FragmentPtrs & frags, artdaq::Fragment::fragment\_id\_t id, artdaq::Fragment::sequence\_id\_t seq, artdaq::Fragment::timestamp\_t ts )

Copy data from the relevant data buffer that matches the given timestamp.

Parameters

<i>frags</i>	Output Fragments
<i>id</i>	Fragment ID of buffer to search
<i>seq</i>	Sequence ID of output Fragment
<i>ts</i>	Timestamp of output Fragment (used to determine window limits)

Definition at line 593 of file FragmentBuffer.cc.

#### 6.96.3.9 bool artdaq::FragmentBuffer::check\_stop ( ) [protected]

Routine used by applyRequests to make sure that all outstanding requests have been fulfilled before returning.

**Returns**

The logical AND of `should_stop`, `mode` is not Ignored, and requests list size equal to 0

Definition at line 212 of file `FragmentBuffer.cc`.

**6.96.3.10 void artdaq::FragmentBuffer::checkDataBuffer ( Fragment::fragment\_id\_t *id* )**

Perform data buffer pruning operations for the given buffer. If the RequestMode is Single, removes all but the latest Fragment from the data buffer.

**Parameters**

<i>id</i>	Fragment ID of buffer In Window and Buffer RequestModes, this function discards the oldest Fragment objects until the data buffer is below its size constraints, then also checks for stale Fragments, based on the timestamp of the most recent Fragment.
-----------	--

Definition at line 410 of file `FragmentBuffer.cc`.

**6.96.3.11 void artdaq::FragmentBuffer::checkSentWindows ( Fragment::sequence\_id\_t *seq* )**

Check the `windows_sent_ooo_map` for sequence IDs that may be removed.

**Parameters**

<i>seq</i>	Sequence ID of current window
------------	-------------------------------

Definition at line 942 of file `FragmentBuffer.cc`.

**6.96.3.12 size\_t artdaq::FragmentBuffer::dataBufferFragmentCount\_ ( ) [protected]**

Get the total number of Fragments in all data buffers.

**Returns**

Number of Fragments in all data buffers

Definition at line 253 of file `FragmentBuffer.cc`.

**6.96.3.13 bool artdaq::FragmentBuffer::dataBufferIsTooLarge ( Fragment::fragment\_id\_t *id* )**

Test the configured constraints on the data buffer.

**Parameters**

<i>id</i>	Fragment ID of data buffer
-----------	----------------------------

**Returns**

Whether the data buffer is full

Definition at line 339 of file `FragmentBuffer.cc`.

**6.96.3.14 artdaq::Fragment::fragment\_id\_t artdaq::FragmentBuffer::fragment\_id ( ) const [inline], [protected]**

Get the Fragment ID of this Fragment generator.

**Exceptions**

<code>cet::exception("FragmentID")</code>	if there is more than one Fragment ID configured for this Fragment Generator
---	--

**Returns**

Fragment ID for the Fragment Generator

Definition at line 386 of file FragmentBuffer.hh.

6.96.3.15 `std::vector<Fragment::fragment_id_t> artdaq::FragmentBuffer::fragmentIDs ( ) [inline]`

Get the list of Fragment IDs handled by this [FragmentBuffer](#).

**Returns**

A `std::vector<Fragment::fragment_id_t>` containing the Fragment IDs handled by this [FragmentBuffer](#)

Definition at line 326 of file FragmentBuffer.hh.

6.96.3.16 `void artdaq::FragmentBuffer::getDataBufferStats ( Fragment::fragment_id_t id )`

Calculate the size of the dataBuffer and report appropriate metrics.

**Parameters**

<i>id</i>	Fragment ID of buffer
-----------	-----------------------

Definition at line 352 of file FragmentBuffer.cc.

6.96.3.17 `artdaq::Fragment::sequence_id_t artdaq::FragmentBuffer::GetNextSequenceID ( ) const [inline]`

Get the next sequence ID expected by this [FragmentBuffer](#). This is used to track sent windows and missed requests.

**Returns**

The next sequence ID expected by this [FragmentBuffer](#)

Definition at line 370 of file FragmentBuffer.hh.

6.96.3.18 `std::map<Fragment::sequence_id_t, std::chrono::steady_clock::time_point> artdaq::FragmentBuffer::GetSentWindow-List ( Fragment::fragment_id_t id ) [inline]`

Get the map of Window-mode requests fulfilled by this Fragment Generator for the given Fragment ID.

**Parameters**

<i>id</i>	Fragment ID of buffer
-----------	-----------------------

**Returns**

Map of sequence\_id and time\_point for sent Window-mode requests

This function is used in `FragmentBuffer_t` to verify correct functioning of Window mode

Definition at line 312 of file FragmentBuffer.hh.

6.96.3.19 `std::string artdaq::FragmentBuffer::printMode_ ( ) [protected]`

Return the string representation of the current RequestMode.

Returns

The string representation of the current RequestMode

Definition at line 234 of file FragmentBuffer.cc.

6.96.3.20 `RequestMode artdaq::FragmentBuffer::request_mode ( ) const [inline]`

Get the current request mode of the [FragmentBuffer](#)

Returns

Current RequestMode of the CFG

Definition at line 342 of file FragmentBuffer.hh.

6.96.3.21 `void artdaq::FragmentBuffer::Reset ( bool stop )`

Reset the [FragmentBuffer](#) (flushes all Fragments from buffers)

Parameters

<i>stop</i>	Whether the <a href="#">FragmentBuffer</a> should be stopped during the Reset
-------------	---

Definition at line 123 of file FragmentBuffer.cc.

6.96.3.22 `bool artdaq::FragmentBuffer::sendEmptyFragment ( FragmentPtrs & frags, size_t sequenceId, Fragment::fragment_id_t fragmentId, std::string desc )`

Send an EmptyFragmentType Fragment.

Parameters

<i>out</i>	<i>frags</i>	Output list to append EmptyFragmentType to
	<i>sequenceId</i>	Sequence ID of Empty Fragment
	<i>fragmentId</i>	Fragment ID of Empty Fragment
	<i>desc</i>	Message to log with reasoning for sending Empty Fragment

Returns

True if no exceptions

Definition at line 912 of file FragmentBuffer.cc.

6.96.3.23 `void artdaq::FragmentBuffer::sendEmptyFragments ( FragmentPtrs & frags, std::map< Fragment::sequence_id_t, Fragment::timestamp_t > & requests )`

This function is for Buffered and Single request modes, as they can only respond to one data request at a time. If the request message seqID > ev\_counter, simply send empties until they're equal.

## Parameters

out	frags	Output list to append EmptyFragmentType to
	requests	List of requests to process

Definition at line 923 of file FragmentBuffer.cc.

6.96.3.24 void artdaq::FragmentBuffer::SetRequestBuffer ( std::shared\_ptr< RequestBuffer > buffer ) [inline]

Set the pointer to the [RequestBuffer](#) used to retrieve requests.

## Parameters

buffer	Pointer to the <a href="#">RequestBuffer</a>
--------	--

Definition at line 364 of file FragmentBuffer.hh.

6.96.3.25 bool artdaq::FragmentBuffer::waitForDataBufferReady ( Fragment::fragment\_id\_t id )

Wait for the data buffer to drain (dataBufferIsTooLarge returns false), periodically reporting status.

## Parameters

id	Fragment ID of data buffer
----	----------------------------

## Returns

True if wait ended without something else disrupting the run

Definition at line 261 of file FragmentBuffer.cc.

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

- artdaq/artdaq/DAQrate/FragmentBuffer.hh
- artdaq/artdaq/DAQrate/FragmentBuffer.cc

## 6.97 artdaqtest::FragmentBufferTestGenerator Class Reference

CommandableFragmentGenerator derived class for testing.

### Public Member Functions

- [FragmentBufferTestGenerator](#) (const fhicl::ParameterSet &ps)  
*FragmentBufferTestGenerator Constructor.*
- artdaq::FragmentPtrs [Generate](#) (size\_t n, std::vector< artdaq::Fragment::fragment\_id\_t > fragmentIds=std::vector< artdaq::Fragment::fragment\_id\_t >())  
*Generate Fragments.*
- void [setTimestamp](#) (artdaq::Fragment::timestamp\_t ts)  
*Set the timestamp to be used for the next Fragment.*
- artdaq::Fragment::timestamp\_t [getTimestamp](#) ()  
*Get the timestamp that will be used for the next Fragment.*



### 6.97.1 Detailed Description

CommandableFragmentGenerator derived class for testing.

Definition at line 38 of file FragmentBuffer\_t.cc.

### 6.97.2 Member Function Documentation

**6.97.2.1** `artdaq::FragmentPtrs artdaqtest::FragmentBufferTestGenerator::Generate ( size_t n, std::vector< artdaq::Fragment::fragment_id_t > fragmentIds = std::vector<artdaq::Fragment::fragment_id_t> () )`

Generate Fragments.

Parameters

<i>n</i>	Number of Fragments to generate
<i>fragmentIds</i>	List of Fragment IDs to generate Fragments for (if different than configured fragment IDs)

Returns

artdaq::FragmentPtrs containing generated Fragments

Definition at line 86 of file FragmentBuffer\_t.cc.

**6.97.2.2** `artdaq::Fragment::timestamp_t artdaqtest::FragmentBufferTestGenerator::getTimestamp ( ) [inline]`

Get the timestamp that will be used for the next Fragment.

Returns

The timestamp that will be used for the next Fragment

Definition at line 65 of file FragmentBuffer\_t.cc.

**6.97.2.3** `void artdaqtest::FragmentBufferTestGenerator::setTimestamp ( artdaq::Fragment::timestamp_t ts ) [inline]`

Set the timestamp to be used for the next Fragment.

Parameters

<i>ts</i>	Timestamp to be used for the next Fragment
-----------	--

Definition at line 59 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.98 artdaq::FragmentReceiverManager Class Reference

Receives Fragment objects from one or more [DataSenderManager](#) instances using [TransferInterface](#) plugins Data-ReceiverManager runs a reception thread for each source, and can automatically suppress reception from sources which are going faster than the others.

```
#include <artdaq/DAQrate/FragmentReceiverManager.hh>
```

## Public Member Functions

- [FragmentReceiverManager](#) (const fhicl::ParameterSet &ps)  
*FragmentReceiverManager Constructor.*
- virtual [~FragmentReceiverManager](#) ()  
*FragmentReceiverManager Destructor.*
- FragmentPtr [recvFragment](#) (int &rank, size\_t timeout\_usec=0)  
*Receive a Fragment.*
- size\_t [count](#) () const  
*Return the count of Fragment objects received by this [FragmentReceiverManager](#).*
- size\_t [slotCount](#) (size\_t rank) const  
*Get the count of Fragment objects received by this [FragmentReceiverManager](#) from a given source.*
- size\_t [byteCount](#) () const  
*Get the total size of all data recieved by this [FragmentReceiverManager](#).*
- void [start\\_threads](#) ()  
*Start receiver threads for all enabled sources.*
- std::set< int > [enabled\\_sources](#) () const  
*Get the list of enabled sources.*
- std::set< int > [running\\_sources](#) () const  
*Get the list of sources which are still receiving data.*

### 6.98.1 Detailed Description

Receives Fragment objects from one or more [DataSenderManager](#) instances using [TransferInterface](#) plugins. DataReceiverManager runs a reception thread for each source, and can automatically suppress reception from sources which are going faster than the others.

Definition at line 29 of file `FragmentReceiverManager.hh`.

### 6.98.2 Constructor & Destructor Documentation

6.98.2.1 `artdaq::FragmentReceiverManager::FragmentReceiverManager ( const fhicl::ParameterSet & ps ) [explicit]`

[FragmentReceiverManager](#) Constructor.

#### Parameters

<i>ps</i>	ParameterSet used to configure the <a href="#">FragmentReceiverManager</a>
-----------	--

```
* FragmentReceiverManager accepts the following Parameters:
* "auto_suppression_enabled" (Default: true): Whether to suppress a source that gets too far ahead
* "max_receive_difference" (Default: 50): Threshold (in sequence ID) for suppressing a source
* "receive_timeout_usec" (Default: 100000): The timeout for receive operations
* "enabled_sources" (OPTIONAL): List of sources which are enabled. If not specified, all sources are assumed enabled
* "sources" (Default: blank table): FHiCL table containing TransferInterface configurations for each source.
* NOTE: "source_rank" MUST be specified (and unique) for each source!
*
```

Definition at line 11 of file `FragmentReceiverManager.cc`.

### 6.98.3 Member Function Documentation

#### 6.98.3.1 `size_t artdaq::FragmentReceiverManager::byteCount ( ) const` `[inline]`

Get the total size of all data recieved by this [FragmentReceiverManager](#).

##### Returns

The total size of all data received by this [FragmentReceiverManager](#)

Definition at line 239 of file `FragmentReceiverManager.hh`.

#### 6.98.3.2 `size_t artdaq::FragmentReceiverManager::count ( ) const` `[inline]`

Return the count of Fragment objects received by this [FragmentReceiverManager](#).

##### Returns

The count of Fragment objects received by this [FragmentReceiverManager](#)

Definition at line 225 of file `FragmentReceiverManager.hh`.

#### 6.98.3.3 `std::set< int > artdaq::FragmentReceiverManager::enabled_sources ( ) const`

Get the list of enabled sources.

##### Returns

The list of enabled sources

Definition at line 239 of file `FragmentReceiverManager.cc`.

#### 6.98.3.4 `artdaq::FragmentPtr artdaq::FragmentReceiverManager::recvFragment ( int & rank, size_t timeout_usec = 0 )`

Receive a Fragment.

##### Parameters

<code>out</code>	<code>rank</code>	Rank of sender that sent the Fragment, or RECV_TIMEOUT
	<code>timeout_usec</code>	<a href="#">Timeout</a> to wait for a Fragment to become ready

##### Returns

Pointer to received Fragment. May be nullptr if no Fragments are ready

Definition at line 180 of file `FragmentReceiverManager.cc`.

#### 6.98.3.5 `std::set< int > artdaq::FragmentReceiverManager::running_sources ( ) const`

Get the list of sources which are still receiving data.

##### Returns

std::set containing ranks of sources which are still receiving data

Definition at line 226 of file `FragmentReceiverManager.cc`.

6.98.3.6 `size_t artdaq::FragmentReceiverManager::slotCount ( size_t rank ) const` `[inline]`

Get the count of Fragment objects received by this [FragmentReceiverManager](#) from a given source.

Parameters

<i>rank</i>	Source rank to get count for
-------------	------------------------------

Returns

The count of Fragment objects received by this [FragmentReceiverManager](#) from the source

Definition at line 232 of file `FragmentReceiverManager.hh`.

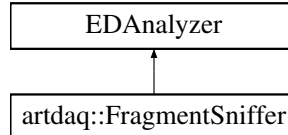
The documentation for this class was generated from the following files:

- `artdaq/artdaq/DAQrate/FragmentReceiverManager.hh`
- `artdaq/artdaq/DAQrate/FragmentReceiverManager.cc`

## 6.99 artdaq::FragmentSniffer Class Reference

This `art::EDAnalyzer` plugin tries to get Fragments from each event, asserting that the correct number of Fragments were present.

Inheritance diagram for `artdaq::FragmentSniffer`:



### Public Member Functions

- [FragmentSniffer](#) (`fhicl::ParameterSet const &p`)  
*FragmentSniffer Constructor.*
- [~FragmentSniffer](#) () `override=default`  
*Default destructor.*
- `void analyze` (`art::Event const &e`) `override`  
*Called for each event. Asserts that Fragment objects are present in the event and that the correct number of Fragments were found.*
- `void endJob` () `override`  
*Called at the end of the job. Asserts that the number of events processed was equal to the expected number.*

### 6.99.1 Detailed Description

This `art::EDAnalyzer` plugin tries to get Fragments from each event, asserting that the correct number of Fragments were present.

Definition at line 23 of file `FragmentSniffer_module.cc`.

## 6.99.2 Constructor & Destructor Documentation

### 6.99.2.1 artdaq::FragmentSniffer::FragmentSniffer ( fhicl::ParameterSet const & *p* ) [explicit]

[FragmentSniffer](#) Constructor.

#### Parameters

<i>p</i>	ParameterSet used to configure <a href="#">FragmentSniffer</a>
----------	--

```
* FragmentSniffer accepts the following Parameters:
* "raw_label" (Default: "daq"): Label under which Fragments are stored
* "product_instance_name" (REQUIRED): Instance name under which Fragments are stored (Should be Fragment type name)
* "num_frgs_per_event" (REQUIRED): Expected number of Fragments in each event
* "num_events_expected" (Default: 0): Expected number of events in the job. If 0, will not perform end-of-job test
*
```

Definition at line 69 of file `FragmentSniffer_module.cc`.

## 6.99.3 Member Function Documentation

### 6.99.3.1 void artdaq::FragmentSniffer::analyze ( art::Event const & *e* ) [override]

Called for each event. Asserts that Fragment objects are present in the event and that the correct number of Fragments were found.

#### Parameters

<i>e</i>	Event to analyze
----------	------------------

Definition at line 77 of file `FragmentSniffer_module.cc`.

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

- `artdaq/test/ArtModules/FragmentSniffer_module.cc`

## 6.100 artdaq::FragmentStoreElement Class Reference

This class contains tracking information for all Fragment objects which have been received from a specific source.

```
#include <artdaq/DAQrate/FragmentReceiverManager.hh>
```

### Public Member Functions

- [FragmentStoreElement](#) ()  
*FragmentStoreElement* Constructor.
- bool [empty](#) () const  
Are any Fragment objects contained in this [FragmentStoreElement](#)?
- void [emplace\\_front](#) (FragmentPtr &&frag)  
Add a Fragment to the front of the [FragmentStoreElement](#).
- void [emplace\\_back](#) (FragmentPtr &&frag)  
Add a Fragment to the end of the [FragmentStoreElement](#).
- FragmentPtr [front](#) ()

Remove the first Fragment from the [FragmentStoreElement](#) and return it.

- void [SetEndOfData](#) (size\_t eod)

Set the End-Of-Data marker value for this Receiver.

- size\_t [GetEndOfData](#) () const

Get the value of the End-Of-Data marker for this Receiver.

- size\_t [size](#) () const

Get the number of Fragments stored in this [FragmentStoreElement](#).

### 6.100.1 Detailed Description

This class contains tracking information for all Fragment objects which have been received from a specific source.

This class was designed so that there could be a mutex for each source, instead of locking all sources whenever a Fragment had to be retrieved. [FragmentStoreElement](#) is itself a container type, sorted by Fragment arrival time. It is a modified queue, with only the first element accessible, but it allows elements to be added to either end (for rejected Fragments).

Definition at line 142 of file [FragmentReceiverManager.hh](#).

### 6.100.2 Member Function Documentation

6.100.2.1 void [artdaq::FragmentStoreElement::emplace\\_back](#) ( [FragmentPtr](#) && *frag* ) [\[inline\]](#)

Add a Fragment to the end of the [FragmentStoreElement](#).

Parameters

<i>frag</i>	Fragment to add
-------------	-----------------

Definition at line 179 of file [FragmentReceiverManager.hh](#).

6.100.2.2 void [artdaq::FragmentStoreElement::emplace\\_front](#) ( [FragmentPtr](#) && *frag* ) [\[inline\]](#)

Add a Fragment to the front of the [FragmentStoreElement](#).

Parameters

<i>frag</i>	Fragment to add
-------------	-----------------

Definition at line 168 of file [FragmentReceiverManager.hh](#).

6.100.2.3 bool [artdaq::FragmentStoreElement::empty](#) ( ) const [\[inline\]](#)

Are any Fragment objects contained in this [FragmentStoreElement](#)?

Returns

Whether any Fragment objects are contained in this [FragmentStoreElement](#)

Definition at line 159 of file [FragmentReceiverManager.hh](#).

**6.100.2.4** `FragmentPtr artdaq::FragmentStoreElement::front ( ) [inline]`

Remove the first Fragment from the [FragmentStoreElement](#) and return it.

**Returns**

The first Fragment in the [FragmentStoreElement](#)

Definition at line 190 of file `FragmentReceiverManager.hh`.

**6.100.2.5** `size_t artdaq::FragmentStoreElement::GetEndOfData ( ) const [inline]`

Get the value of the End-Of-Data marker for this Receiver.

**Returns**

The value of the End-Of-Data marker. Returns -1 (0xFFFFFFFFFFFFFFFF) if no EndOfData Fragments received

Definition at line 208 of file `FragmentReceiverManager.hh`.

**6.100.2.6** `void artdaq::FragmentStoreElement::SetEndOfData ( size_t eod ) [inline]`

Set the End-Of-Data marker value for this Receiver.

**Parameters**

<i>eod</i>	Number of Receives expected for this receiver
------------	---

Definition at line 203 of file `FragmentReceiverManager.hh`.

**6.100.2.7** `size_t artdaq::FragmentStoreElement::size ( ) const [inline]`

Get the number of Fragments stored in this [FragmentStoreElement](#).

**Returns**

The number of Fragments stored in this [FragmentStoreElement](#)

Definition at line 214 of file `FragmentReceiverManager.hh`.

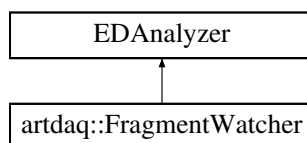
The documentation for this class was generated from the following file:

- `artdaq/artdaq/DAQrate/FragmentReceiverManager.hh`

**6.101 artdaq::FragmentWatcher Class Reference**

An `art::EDAnalyzer` module which checks events for certain error conditions (missing fragments, empty fragments, etc)

Inheritance diagram for `artdaq::FragmentWatcher`:



## Public Member Functions

- [FragmentWatcher](#) (fhicl::ParameterSet const &pset)  
*FragmentWatcher Constructor.*
- [~FragmentWatcher](#) () override  
*Virtual Destructor. Shuts down MetricManager if one is present.*
- void [analyze](#) (art::Event const &evt) override  
*Analyze each event, using the configured mode bitmask.*

### 6.101.1 Detailed Description

An art::EDAnalyzer module which checks events for certain error conditions (missing fragments, empty fragments, etc)  
Definition at line 47 of file FragmentWatcher\_module.cc.

### 6.101.2 Constructor & Destructor Documentation

6.101.2.1 `artdaq::FragmentWatcher::FragmentWatcher ( fhicl::ParameterSet const & pset )` `[explicit]`

[FragmentWatcher](#) Constructor.

Parameters

<i>pset</i>	ParameterSet used to configure <a href="#">FragmentWatcher</a>
-------------	--

[FragmentWatcher](#) accepts the following Parameters: mode\_bitmask (default: 0x1): Mask of modes to use. BASIC\_COUNTS\_MODE = 0, FRACTIONAL\_COUNTS\_MODE = 1, DETAILED\_COUNTS\_MODE = 2 metrics\_reporting\_level (default: 1): Level to use for metrics reporting metrics: A artdaq::MetricManager::Config ParameterSet used to configure MetricManager reporting for this module

Definition at line 99 of file FragmentWatcher\_module.cc.

### 6.101.3 Member Function Documentation

6.101.3.1 `void artdaq::FragmentWatcher::analyze ( art::Event const & evt )` `[override]`

Analyze each event, using the configured mode bitmask.

Parameters

<i>evt</i>	art::Event to analyze
------------	-----------------------

Definition at line 125 of file FragmentWatcher\_module.cc.

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

- artdaq/artdaq/ArtModules/FragmentWatcher\_module.cc

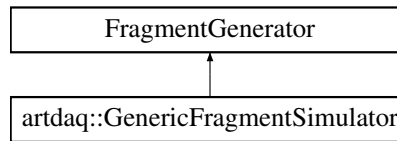
## 6.102 artdaq::GenericFragmentSimulator Class Reference

[GenericFragmentSimulator](#) creates simulated Generic events, with data distributed according to a "histogram" provided in the configuration data.

```
#include <artdaq/DAQdata/GenericFragmentSimulator.hh>
```



Inheritance diagram for artdaq::GenericFragmentSimulator:



## Classes

- struct [Config](#)

Configuration of the [GenericFragmentSimulator](#). May be used for parameter validation

## Public Types

- enum [content\\_selector\\_t](#) : uint8\_t { [content\\_selector\\_t::EMPTY](#) = 0, [content\\_selector\\_t::FRAG\\_ID](#) = 1, [content\\_selector\\_t::RANDOM](#) = 2, [content\\_selector\\_t::DEAD\\_BEEF](#) }

What type of content should the [GenericFragmentSimulator](#) put in Fragment objects?

- using [Parameters](#) = fhicl::WrappedTable< [Config](#) >

Used for [ParameterSet](#) validation (if desired)

## Public Member Functions

- [GenericFragmentSimulator](#) (fhicl::ParameterSet const &ps)

[GenericFragmentSimulator](#) Constructor.

- bool [getNext](#) (Fragment::sequence\_id\_t sequence\_id, Fragment::fragment\_id\_t fragment\_id, FragmentPtr &frag\_ptr)

Generate a Fragment according to the value of the [content\\_selector\\_t](#) enum.

- bool [getNext](#) (FragmentPtrs &output) override

Get the next Fragment from the generator.

- std::vector  
< Fragment::fragment\_id\_t > [fragmentIDs](#) () override

Get the Fragment IDs generated by this instance.

### 6.102.1 Detailed Description

[GenericFragmentSimulator](#) creates simulated Generic events, with data distributed according to a "histogram" provided in the configuration data.

With this implementation, a single call to getNext(frag) will return a complete event (event ids are incremented automatically); fragment ids are sequential. Event size and content are both configurable; see the implementation for details.

Definition at line 36 of file [GenericFragmentSimulator.hh](#).

## 6.102.2 Member Enumeration Documentation

6.102.2.1 `enum artdaq::GenericFragmentSimulator::content_selector_t : uint8_t [strong]`

What type of content should the [GenericFragmentSimulator](#) put in Fragment objects?

### Enumerator

- EMPTY** Nothing (Default-initialized Fragment)
- FRAG\_ID** Fill the payload with the Fragment ID.
- RANDOM** Use a random distribution to fill the payload.
- DEAD\_BEEF** Fill the payload with 0xDEADBEEFDEADBEEF.

Definition at line 74 of file GenericFragmentSimulator.hh.

## 6.102.3 Constructor & Destructor Documentation

6.102.3.1 `artdaq::GenericFragmentSimulator::GenericFragmentSimulator ( fhicl::ParameterSet const & ps ) [explicit]`

[GenericFragmentSimulator](#) Constructor.

### Parameters

<i>ps</i>	ParameterSet used to configure the <a href="#">GenericFragmentSimulator</a> . See <a href="#">artdaq::GenericFragmentSimulator::Config</a>
-----------	--

Definition at line 11 of file GenericFragmentSimulator.cc.

## 6.102.4 Member Function Documentation

6.102.4.1 `std::vector<Fragment::fragment_id_t> artdaq::GenericFragmentSimulator::fragmentIDs ( ) [inline], [override]`

Get the Fragment IDs generated by this instance.

### Returns

The Fragment IDs generated by this instance

Definition at line 110 of file GenericFragmentSimulator.hh.

6.102.4.2 `bool artdaq::GenericFragmentSimulator::getNext ( Fragment::sequence_id_t sequence_id, Fragment::fragment_id_t fragment_id, FragmentPtr & frag_ptr )`

Generate a Fragment according to the value of the content\_selector\_t enum.

### Parameters

	<i>sequence_id</i>	Sequence ID of generated Fragment
	<i>fragment_id</i>	Fragment ID of generated Fragment
out	<i>frag_ptr</i>	Generated Fragment

**Returns**

True if no exception or assertion failure

Definition at line 42 of file GenericFragmentSimulator.cc.

6.102.4.3 `bool artdaq::GenericFragmentSimulator::getNext ( FragmentPtrs & output ) [inline],[override]`

Get the next Fragment from the generator.

**Parameters**

<code>out</code>	<code>output</code>	List of FragmentPtr objects to add the new Fragment to
------------------	---------------------	--

**Returns**

Whether data taking should continue

Definition at line 101 of file GenericFragmentSimulator.hh.

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

- artdaq/artdaq/DAQdata/GenericFragmentSimulator.hh
- artdaq/artdaq/DAQdata/GenericFragmentSimulator.cc

## 6.103 artdaq::GenToBufferTest Class Reference

Test fixture for GenToBuffer\_t.

**Public Member Functions**

- [GenToBufferTest](#) (fhicl::ParameterSet const &ps)  
*GenToBufferTest constructor.*
- void [start](#) (int run\_number)  
*Start the test fixture.*
- void [stop](#) ()  
*Stop the test fixture.*
- std::shared\_ptr< [RequestBuffer](#) > [GetRequestBuffer](#) ()  
*Get a handle to the RequestBuffer.*

### 6.103.1 Detailed Description

Test fixture for GenToBuffer\_t.

Definition at line 29 of file GenToBuffer\_t.cc.

### 6.103.2 Constructor & Destructor Documentation

6.103.2.1 `artdaq::GenToBufferTest::GenToBufferTest ( fhicl::ParameterSet const & ps ) [inline],[explicit]`

[GenToBufferTest](#) constructor.

## Parameters

<i>ps</i>	ParameterSet for <a href="#">GenToBufferTest</a>
-----------	--

Definition at line 36 of file GenToBuffer\_t.cc.

### 6.103.3 Member Function Documentation

6.103.3.1 `std::shared_ptr<RequestBuffer> artdaq::GenToBufferTest::GetRequestBuffer ( ) [inline]`

Get a handle to the [RequestBuffer](#).

## Returns

[RequestBuffer](#) shared\_ptr handle

Definition at line 83 of file GenToBuffer\_t.cc.

6.103.3.2 `void artdaq::GenToBufferTest::start ( int run_number ) [inline]`

Start the test fixture.

## Parameters

<i>run_number</i>	Run Number to use in Start commands
-------------------	-------------------------------------

Definition at line 52 of file GenToBuffer\_t.cc.

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

- `artdaq/test/DAQrate/GenToBuffer_t.cc`

## 6.104 artdaq::GetPackageBuildInfo Struct Reference

Wrapper around the [artdaq::GetPackageBuildInfo::getPackageBuildInfo](#) function.

```
#include <artdaq/BuildInfo/GetPackageBuildInfo.hh>
```

### Static Public Member Functions

- static `artdaq::PackageBuildInfo` [getPackageBuildInfo](#) ()  
*Gets the version number and build timestmap for artdaq.*

#### 6.104.1 Detailed Description

Wrapper around the [artdaq::GetPackageBuildInfo::getPackageBuildInfo](#) function.

Definition at line 16 of file GetPackageBuildInfo.hh.

## 6.104.2 Member Function Documentation

### 6.104.2.1 static artdaq::PackageBuildInfo artdaq::GetPackageBuildInfo::getPackageBuildInfo ( ) [static]

Gets the version number and build timestamp for artdaq.

#### Returns

An artdaq::PackageBuildInfo object containing the version number and build timestamp for artdaq

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

- artdaq/artdaq/BuildInfo/GetPackageBuildInfo.hh

## 6.105 artdaq::Globals Class Reference

The [artdaq::Globals](#) class contains several variables which are useful across the entire artdaq system.

```
#include <artdaq/DAQdata/Globals.hh>
```

### Static Public Member Functions

- static uint32\_t [seedAndRandom\\_](#) ()  
*Seed the C random number generator with the current time (if that has not been done already) and generate a random value.*
- static int [getPartitionNumber\\_](#) ()  
*Get the current partition number, as defined by the ARTDAQ\_PARTITION\_NUMBER environment variable.*
- static std::string [GetMFIteration\\_](#) ()  
*Get the current iteration for MessageFacility messages.*
- static std::string [GetMFModuleName\\_](#) ()  
*Get the current module name for MessageFacility messages.*
- static void [SetMFIteration\\_](#) (std::string const &name)  
*Set the current iteration for MessageFacility messages.*
- static void [SetMFModuleName\\_](#) (std::string const &name)  
*Set the current module name for MessageFacility messages.*
- static void [CleanUpGlobals](#) ()  
*Clean up statically-allocated Manager class instances.*

### Static Public Attributes

- static int [my\\_rank\\_](#) = -1  
*The rank of the current application.*
- static std::unique\_ptr< MetricManager > [metricMan\\_](#) = std::make\_unique<artdaq::MetricManager>()  
*A handle to MetricManager.*
- static std::unique\_ptr< PortManager > [portMan\\_](#) = std::make\_unique<artdaq::PortManager>()  
*A handle to PortManager.*

- static std::string `app_name_`  
*The name of the current application, to be used in logging and metrics.*
- static int `partition_number_` = -1  
*The partition number of the current application.*
- static std::mutex `mfttrace_mutex_`  
*Mutex to protect mfttrace\_module\_ and mfttrace\_iteration\_.*
- static std::string `mfttrace_module_` = "DAQ"  
*MessageFacility's module and iteration are thread-local, but we want to use them to represent global state in artdaq.*
- static std::string `mfttrace_iteration_` = "Booted"  
*MessageFacility's module and iteration are thread-local, but we want to use them to represent global state in artdaq.*

### 6.105.1 Detailed Description

The `artdaq::Globals` class contains several variables which are useful across the entire artdaq system.

Definition at line 38 of file `Globals.hh`.

### 6.105.2 Member Function Documentation

#### 6.105.2.1 static std::string artdaq::Globals::GetMFIteration\_( ) [inline],[static]

Get the current iteration for MessageFacility messages.

Returns

The current iteration

Definition at line 117 of file `Globals.hh`.

#### 6.105.2.2 static std::string artdaq::Globals::GetMFModuleName\_( ) [inline],[static]

Get the current module name for MessageFacility messages.

Returns

The current module name

Definition at line 127 of file `Globals.hh`.

#### 6.105.2.3 static int artdaq::Globals::getPartitionNumber\_( ) [inline],[static]

Get the current partition number, as defined by the ARTDAQ\_PARTITION\_NUMBER environment variable.

Returns

The current partition number (defaults to 0 if unset, will be between 0 and 127)

Definition at line 81 of file `Globals.hh`.

6.105.2.4 `static uint32_t artdaq::Globals::seedAndRandom_ ( ) [inline],[static]`

Seed the C random number generator with the current time (if that has not been done already) and generate a random value.

#### Returns

A random number.

Definition at line 55 of file Globals.hh.

6.105.2.5 `static void artdaq::Globals::SetMFIteration_ ( std::string const & name ) [inline],[static]`

Set the current iteration for MessageFacility messages.

#### Parameters

<i>name</i>	The current iteration
-------------	-----------------------

Definition at line 137 of file Globals.hh.

6.105.2.6 `static void artdaq::Globals::SetMFModuleName_ ( std::string const & name ) [inline],[static]`

Set the current module name for MessageFacility messages.

#### Parameters

<i>name</i>	The current module name
-------------	-------------------------

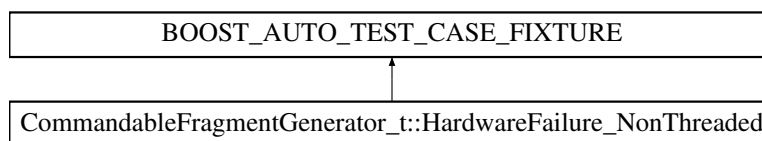
Definition at line 147 of file Globals.hh.

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

- artdaq/artdaq/DAQdata/Globals.hh
- artdaq/artdaq/DAQdata/Globals.cc

## 6.106 CommandableFragmentGenerator\_t::HardwareFailure\_NonThreaded Struct Reference

Inheritance diagram for CommandableFragmentGenerator\_t::HardwareFailure\_NonThreaded:



#### Public Member Functions

- void **test\_method** ()

### 6.106.1 Detailed Description

Definition at line 362 of file CommandableFragmentGenerator\_t.cc.

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

- artdaq/test/Generators/CommandableFragmentGenerator\_t.cc

## 6.107 CommandableFragmentGenerator\_t::HardwareFailure\_NonThreaded\_id Struct Reference

### 6.107.1 Detailed Description

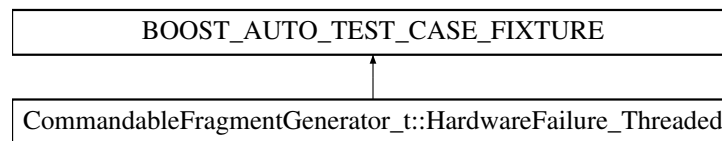
Definition at line 362 of file CommandableFragmentGenerator\_t.cc.

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

- artdaq/test/Generators/CommandableFragmentGenerator\_t.cc

## 6.108 CommandableFragmentGenerator\_t::HardwareFailure\_Threaded Struct Reference

Inheritance diagram for CommandableFragmentGenerator\_t::HardwareFailure\_Threaded:



### Public Member Functions

- void **test\_method** ()

### 6.108.1 Detailed Description

Definition at line 400 of file CommandableFragmentGenerator\_t.cc.

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

- artdaq/test/Generators/CommandableFragmentGenerator\_t.cc

## 6.109 CommandableFragmentGenerator\_t::HardwareFailure\_Threaded\_id Struct Reference

### 6.109.1 Detailed Description

Definition at line 400 of file CommandableFragmentGenerator\_t.cc.

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



- artdaq/test/Generators/CommandableFragmentGenerator\_t.cc

## 6.110 artdaq::HostMap::HostConfig Struct Reference

Entries in the host\_map should have these parameters. May be used for parameter validation

```
#include <artdaq/DAQdata/HostMap.hh>
```

### Public Attributes

- fhicl::Atom< int > [rank](#) {fhicl::Name{"rank"}, fhicl::Comment{"Rank index"}}  
*"rank": Rank index*
- fhicl::Atom< std::string > [host](#) {fhicl::Name{"host"}, fhicl::Comment{"Hostname for artdaq application with this rank"}}  
*"host": Hostname for artdaq application with this rank*

### 6.110.1 Detailed Description

Entries in the host\_map should have these parameters. May be used for parameter validation

Definition at line 23 of file HostMap.hh.

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

- artdaq/artdaq/DAQdata/HostMap.hh

## 6.111 artdaq::HostMap Struct Reference

### Classes

- struct [Config](#)  
*Template for the host\_map configuration parameter.*
- struct [HostConfig](#)  
*Entries in the host\_map should have these parameters. May be used for parameter validation*

### 6.111.1 Detailed Description

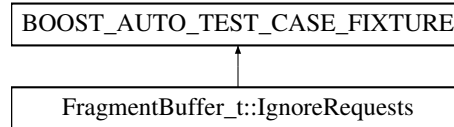
Definition at line 18 of file HostMap.hh.

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

- artdaq/artdaq/DAQdata/HostMap.hh

## 6.112 FragmentBuffer\_t::IgnoreRequests Struct Reference

Inheritance diagram for FragmentBuffer\_t::IgnoreRequests:



## Public Member Functions

- void **test\_method** ()

### 6.112.1 Detailed Description

Definition at line 131 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.113 FragmentBuffer\_t::IgnoreRequests\_id Struct Reference

### 6.113.1 Detailed Description

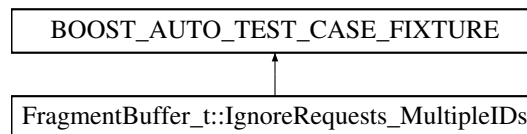
Definition at line 131 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.114 FragmentBuffer\_t::IgnoreRequests\_MultipleIDs Struct Reference

Inheritance diagram for FragmentBuffer\_t::IgnoreRequests\_MultipleIDs:



## Public Member Functions

- void **test\_method** ()

### 6.114.1 Detailed Description

Definition at line 1176 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.115 FragmentBuffer\_t::IgnoreRequests\_MultipleIDs\_id Struct Reference

### 6.115.1 Detailed Description

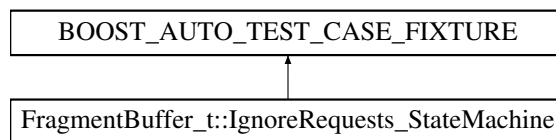
Definition at line 1176 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.116 FragmentBuffer\_t::IgnoreRequests\_StateMachine Struct Reference

Inheritance diagram for FragmentBuffer\_t::IgnoreRequests\_StateMachine:



### Public Member Functions

- void **test\_method** ()

### 6.116.1 Detailed Description

Definition at line 2035 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.117 FragmentBuffer\_t::IgnoreRequests\_StateMachine\_id Struct Reference

### 6.117.1 Detailed Description

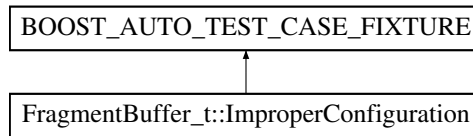
Definition at line 2035 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.118 FragmentBuffer\_t::ImproperConfiguration Struct Reference

Inheritance diagram for FragmentBuffer\_t::ImproperConfiguration:



### Public Member Functions

- void **test\_method** ()

#### 6.118.1 Detailed Description

Definition at line 108 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.119 FragmentBuffer\_t::ImproperConfiguration\_id Struct Reference

#### 6.119.1 Detailed Description

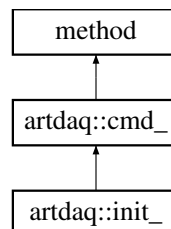
Definition at line 108 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.120 artdaq::init\_ Class Reference

Inheritance diagram for artdaq::init\_:



### Public Member Functions

- [init\\_](#) ([xmlrpc\\_commander](#) &c)

## Static Public Attributes

- static const uint64\_t [defaultTimeout](#) = 45
- static const uint64\_t [defaultTimestamp](#) = std::numeric\_limits<const uint64\_t>::max()

## Additional Inherited Members

### 6.120.1 Detailed Description

Command class representing an init transition

Definition at line 841 of file xmlrpc\_commander.cc.

### 6.120.2 Constructor & Destructor Documentation

6.120.2.1 `artdaq::init::init( xmlrpc_commander & c )` `[inline]`, `[explicit]`

Command class Constructor \ \*

Parameters

<code>c</code>	<code>xmlrpc_commander</code> to send parsed command to \
----------------	---

Definition at line 841 of file xmlrpc\_commander.cc.

### 6.120.3 Member Data Documentation

6.120.3.1 `const uint64_t artdaq::init::defaultTimeout = 45` `[static]`

Default timeout for command

Definition at line 841 of file xmlrpc\_commander.cc.

6.120.3.2 `const uint64_t artdaq::init::defaultTimestamp = std::numeric_limits<const uint64_t>::max()` `[static]`

Default timestamp for Command

Definition at line 841 of file xmlrpc\_commander.cc.

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

- artdaq/artdaq/ExternalComms/xmlrpc\_commander.cc

## 6.121 art::RootOutputConfig::KeysToIgnore Struct Reference

These keys should be ignored by the configuration validation processor

```
#include <artdaq/ArtModules/detail/ArtConfig.hh>
```

## Public Member Functions

- `std::set< std::string > operator() () const`

*Get the keys to ignore*

### 6.121.1 Detailed Description

These keys should be ignored by the configuration validation processor

Definition at line 111 of file ArtConfig.hh.

### 6.121.2 Member Function Documentation

6.121.2.1 `std::set<std::string> art::RootOutputConfig::KeysToIgnore::operator() ( ) const` `[inline]`

Get the keys to ignore

Returns

Set of keys to ignore

Definition at line 117 of file ArtConfig.hh.

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

- `artdaq/artdaq/ArtModules/detail/ArtConfig.hh`

## 6.122 art::RootDAQOut::Config::KeysToIgnore Struct Reference

### Public Member Functions

- `set< string > operator() () const`

### 6.122.1 Detailed Description

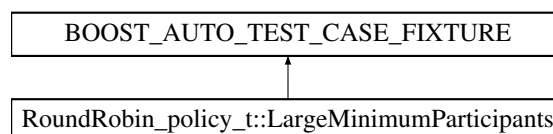
Definition at line 160 of file RootDAQOut\_module.cc.

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

- `artdaq/artdaq/ArtModules/RootDAQOutput-s124/RootDAQOut_module.cc`

## 6.123 RoundRobin\_policy\_t::LargeMinimumParticipants Struct Reference

Inheritance diagram for RoundRobin\_policy\_t::LargeMinimumParticipants:



## Public Member Functions

- void **test\_method** ()

### 6.123.1 Detailed Description

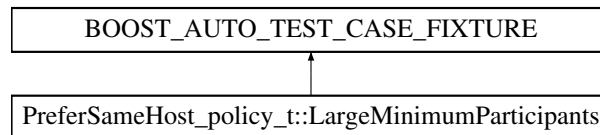
Definition at line 132 of file RoundRobin\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/RoundRobin\_policy\_t.cc

## 6.124 PreferSameHost\_policy\_t::LargeMinimumParticipants Struct Reference

Inheritance diagram for PreferSameHost\_policy\_t::LargeMinimumParticipants:



## Public Member Functions

- void **test\_method** ()

### 6.124.1 Detailed Description

Definition at line 107 of file PreferSameHost\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/PreferSameHost\_policy\_t.cc

## 6.125 RoundRobin\_policy\_t::LargeMinimumParticipants\_id Struct Reference

### 6.125.1 Detailed Description

Definition at line 132 of file RoundRobin\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/RoundRobin\_policy\_t.cc

## 6.126 PreferSameHost\_policy\_t::LargeMinimumParticipants\_id Struct Reference

### 6.126.1 Detailed Description

Definition at line 107 of file PreferSameHost\_policy\_t.cc.

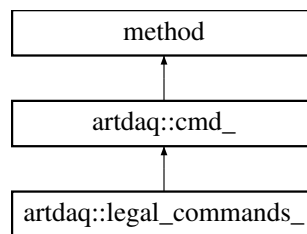
The documentation for this struct was generated from the following file:

- `artdaq/test/RoutingPolicies/PreferSameHost_policy_t.cc`

## 6.127 artdaq::legal\_commands\_ Class Reference

[legal\\_commands\\_](#) Command class

Inheritance diagram for `artdaq::legal_commands_`:



### Public Member Functions

- [legal\\_commands\\_](#) ([xmlrpc\\_commander](#) &c)  
*[legal\\_commands\\_](#) Constructor*

### Additional Inherited Members

### 6.127.1 Detailed Description

[legal\\_commands\\_](#) Command class

Definition at line 1018 of file `xmlrpc_commander.cc`.

### 6.127.2 Constructor & Destructor Documentation

6.127.2.1 `artdaq::legal_commands_::legal_commands_ ( xmlrpc_commander & c )` `[inline]`

[legal\\_commands\\_](#) Constructor

Parameters

<code>c</code>	<a href="#">xmlrpc_commander</a> to send transition commands to
----------------	---

Definition at line 1025 of file `xmlrpc_commander.cc`.

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

- `artdaq/artdaq/ExternalComms/xmlrpc_commander.cc`



## 6.128 artdaq::ListenTransferWrapper Class Reference

[ListenTransferWrapper](#) wraps a [TransferInterface](#) so that it can be used in the ArtdaqInput class to make an art::Source.

```
#include <artdaq/ArtModules/detail/ListenTransferWrapper.hh>
```

### Public Member Functions

- [ListenTransferWrapper](#) (const fhicl::ParameterSet &pset)  
*[ListenTransferWrapper](#) Constructor.*
- virtual [~ListenTransferWrapper](#) ()  
*[ListenTransferWrapper](#) Destructor.*
- artdaq::FragmentPtrs [receiveMessage](#) ()  
*Receive a Fragment from the [TransferInterface](#), and send it to art.*
- std::unordered\_map  
< artdaq::Fragment::type\_t,  
std::unique\_ptr  
< artdaq::Fragments > > [receiveMessages](#) ()  
*Receive all messages for an event from [ArtdaqSharedMemoryService](#).*
- artdaq::FragmentPtrs [receiveInitMessage](#) ()  
*Receive the Init message from the [TransferInterface](#), and send it to art.*
- std::shared\_ptr  
< artdaq::detail::RawEventHeader > [getEventHeader](#) ()  
*Get a pointer to the last received RawEventHeader.*

### 6.128.1 Detailed Description

[ListenTransferWrapper](#) wraps a [TransferInterface](#) so that it can be used in the ArtdaqInput class to make an art::Source.

JCF, May-27-2016

This is the class through which code that wants to access a transfer plugin (e.g., input sources, AggregatorCore, etc.) can do so. Its functionality is such that it satisfies the requirements needed to be a template in the ArtdaqInput class

Definition at line 29 of file ListenTransferWrapper.hh.

### 6.128.2 Constructor & Destructor Documentation

6.128.2.1 artdaq::ListenTransferWrapper::ListenTransferWrapper ( const fhicl::ParameterSet & pset ) [\[explicit\]](#)

[ListenTransferWrapper](#) Constructor.

Parameters

<i>pset</i>	ParameterSet used to configure the <a href="#">ListenTransferWrapper</a>
-------------	--

```
* ListenTransferWrapper accepts the following Parameters:
* "timeoutInUsecs" (Default: 100000): The receive timeout
* "maxEventsBeforeInit" (Default: 5): How many non-Init events to receive before raising an error
* "allowedFragmentTypes" (Default: [226,227,229]): The Fragment type codes for expected Fragments
* "dispatcherConnectTimeout" (Default: 0): Maximum amount of time (in seconds) to wait for the Dispatcher to reach
* "dispatcherConnectRetryInterval_us" (Default 1,000,000): Amount of time to wait between polls of the Dispatcher
* "quitOnFragmentIntegrityProblem" (Default: true): If there is an inconsistency in the received Fragment, throw
```

```

* "allowMultipleRuns" (Default: true): If true, will ignore EndOfData message and reconnect to the Dispatcher once
* "debugLevel" (Default: 0): Enables some additional messages
* "transfer_plugin" (REQUIRED): Name of the TransferInterface plugin to load
*
* This parameter set is also passed to TransferInterface, so any necessary Parameters for TransferInterface or the
* should be included here.
*

```

Definition at line 35 of file ListenTransferWrapper.cc.

### 6.128.3 Member Function Documentation

6.128.3.1 `std::shared_ptr<artdaq::detail::RawEventHeader> artdaq::ListenTransferWrapper::getEventHeader ( )` [inline]

Get a pointer to the last received RawEventHeader.

Returns

a shared\_ptr to the last received RawEventHeader

Definition at line 83 of file ListenTransferWrapper.hh.

6.128.3.2 `artdaq::FragmentPtrs artdaq::ListenTransferWrapper::receiveInitMessage ( )` [inline]

Receive the Init message from the [TransferInterface](#), and send it to art.

Returns

Received InitFragment

Definition at line 74 of file ListenTransferWrapper.hh.

6.128.3.3 `artdaq::FragmentPtrs artdaq::ListenTransferWrapper::receiveMessage ( )`

Receive a Fragment from the [TransferInterface](#), and send it to art.

Returns

Received Fragment

Definition at line 87 of file ListenTransferWrapper.cc.

6.128.3.4 `std::unordered_map< artdaq::Fragment::type_t, std::unique_ptr< artdaq::Fragments > > artdaq::ListenTransferWrapper::receiveMessages ( )`

Receive all messages for an event from [ArtdaqSharedMemoryService](#).

Returns

A map of Fragment::type\_t to a unique\_ptr to Fragments containing all Fragments in an event

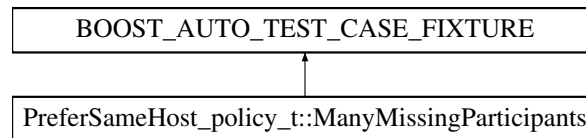
Definition at line 217 of file ListenTransferWrapper.cc.

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

- `artdaq/artdaq/ArtModules/detail/ListenTransferWrapper.hh`
- `artdaq/artdaq/ArtModules/detail/ListenTransferWrapper.cc`

## 6.129 PreferSameHost\_policy\_t::ManyMissingParticipants Struct Reference

Inheritance diagram for PreferSameHost\_policy\_t::ManyMissingParticipants:



### Public Member Functions

- void **test\_method** ()

#### 6.129.1 Detailed Description

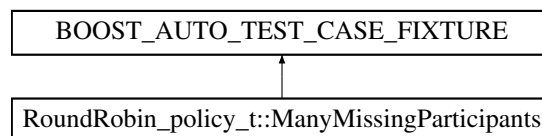
Definition at line 142 of file PreferSameHost\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/PreferSameHost\_policy\_t.cc

## 6.130 RoundRobin\_policy\_t::ManyMissingParticipants Struct Reference

Inheritance diagram for RoundRobin\_policy\_t::ManyMissingParticipants:



### Public Member Functions

- void **test\_method** ()

#### 6.130.1 Detailed Description

Definition at line 167 of file RoundRobin\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/RoundRobin\_policy\_t.cc

## 6.131 PreferSameHost\_policy\_t::ManyMissingParticipants\_id Struct Reference

### 6.131.1 Detailed Description

Definition at line 142 of file PreferSameHost\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/PreferSameHost\_policy\_t.cc

## 6.132 RoundRobin\_policy\_t::ManyMissingParticipants\_id Struct Reference

### 6.132.1 Detailed Description

Definition at line 167 of file RoundRobin\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/RoundRobin\_policy\_t.cc

## 6.133 MessHead Struct Reference

This header is sent by the TCPSocket\_transfer to allow for more efficient writev calls.

```
#include <artdaq/TransferPlugins/detail/SRSockets.hh>
```

### Public Types

- enum [MessType](#) {  
**connect\_v0** = 0, **data\_v0**, **data\_more\_v0**, **stop\_v0**,  
**routing\_v0**, **ack\_v0**, **header\_v0** }

*The Message Type.*

### Public Attributes

- uint8\_t [endian](#)  
*0=little(intel), 1=big*
- [MessType](#) [message\\_type](#)  
*Message Type.*
- int64\_t [source\\_id](#)  
*Rank of the source.*
- union {  
uint32\_t [conn\\_magic](#)  
*unsigned first is better for [MessHead](#) initializer: {0,0,my\_node\_idx\_,CONN\_MAGIC}*  
int32\_t [byte\\_count](#)  
*use CONN\_MAGIC for connect\_v0, data that follow for data\_v0 (and 0 lenght data)*  
};

### 6.133.1 Detailed Description

This header is sent by the TCPSocket\_transfer to allow for more efficient writev calls.

Definition at line 15 of file SRockets.hh.

### 6.133.2 Member Enumeration Documentation

#### 6.133.2.1 enum MessHead::MessType

The Message Type.

Only add to the end!

Definition at line 24 of file SRockets.hh.

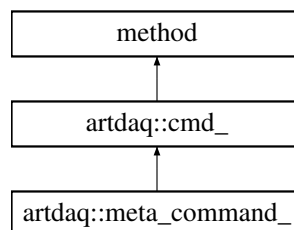
The documentation for this struct was generated from the following file:

- artdaq/artdaq/TransferPlugins/detail/SRockets.hh

## 6.134 artdaq::meta\_command\_ Class Reference

[meta\\_command\\_](#) Command class

Inheritance diagram for artdaq::meta\_command\_:



### Public Member Functions

- [meta\\_command\\_](#)(xmlrpc\_commander &c)  
*meta\_command\_ Constructor*

### Additional Inherited Members

#### 6.134.1 Detailed Description

[meta\\_command\\_](#) Command class

Definition at line 1181 of file xmlrpc\_commander.cc.

#### 6.134.2 Constructor & Destructor Documentation

6.134.2.1 `artdaq::meta_command::meta_command_( xmlrpc_commander & c )` `[inline]`

[meta\\_command\\_](#) Constructor

## Parameters

<code>c</code>	<code>xmlrpc_commander</code> to send transition commands to
----------------	--

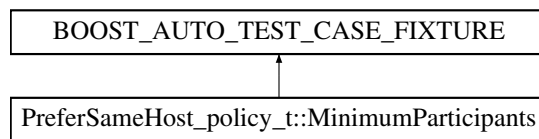
Definition at line 1188 of file `xmlrpc_commander.cc`.

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

- `artdaq/artdaq/ExternalComms/xmlrpc_commander.cc`

## 6.135 PreferSameHost\_policy\_t::MinimumParticipants Struct Reference

Inheritance diagram for `PreferSameHost_policy_t::MinimumParticipants`:



## Public Member Functions

- void **test\_method** ()

### 6.135.1 Detailed Description

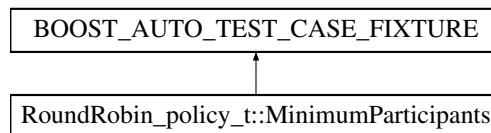
Definition at line 55 of file `PreferSameHost_policy_t.cc`.

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

- `artdaq/test/RoutingPolicies/PreferSameHost_policy_t.cc`

## 6.136 RoundRobin\_policy\_t::MinimumParticipants Struct Reference

Inheritance diagram for `RoundRobin_policy_t::MinimumParticipants`:



## Public Member Functions

- void **test\_method** ()

### 6.136.1 Detailed Description

Definition at line 77 of file RoundRobin\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/RoundRobin\_policy\_t.cc

## 6.137 PreferSameHost\_policy\_t::MinimumParticipants\_id Struct Reference

### 6.137.1 Detailed Description

Definition at line 55 of file PreferSameHost\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/PreferSameHost\_policy\_t.cc

## 6.138 RoundRobin\_policy\_t::MinimumParticipants\_id Struct Reference

### 6.138.1 Detailed Description

Definition at line 77 of file RoundRobin\_policy\_t.cc.

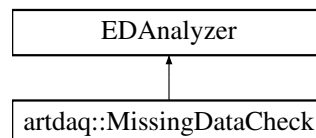
The documentation for this struct was generated from the following file:

- artdaq/test/RoutingPolicies/RoundRobin\_policy\_t.cc

## 6.139 artdaq::MissingDataCheck Class Reference

Check art::Event for potentially missing data.

Inheritance diagram for artdaq::MissingDataCheck:



### Public Member Functions

- [MissingDataCheck](#) (fhicl::ParameterSet const &pset)  
*MissingDataCheck Constructor.*
- [~MissingDataCheck](#) () override=default  
*Default virtual Destructor.*
- void [analyze](#) (art::Event const &e) override  
*This method is called for each art::Event in a file or run.*



- void [endJob](#) () override

*This method is called at the end of the job, used to print a summary.*

### 6.139.1 Detailed Description

Check art::Event for potentially missing data.

Definition at line 37 of file MissingDataCheck\_module.cc.

### 6.139.2 Constructor & Destructor Documentation

#### 6.139.2.1 artdaq::MissingDataCheck::MissingDataCheck ( fhicl::ParameterSet const & *pset* ) [explicit]

[MissingDataCheck](#) Constructor.

Parameters

<i>pset</i>	ParameterSet used to configure <a href="#">MissingDataCheck</a>
-------------	---

```
* MissingDataCheck accepts the following Parameters:
* "raw_data_label" (Default: "daq"): The label used to store artdaq data
* "expected_n_fragments" (Default: -1): number of expected fragments. Uses n_frags in first event if -1
* "verbosity" (Default: 0): verbosity level
*
```

Definition at line 104 of file MissingDataCheck\_module.cc.

### 6.139.3 Member Function Documentation

#### 6.139.3.1 void artdaq::MissingDataCheck::analyze ( art::Event const & *e* ) [override]

This method is called for each art::Event in a file or run.

Parameters

<i>e</i>	The art::Event to analyze
----------	---------------------------

Definition at line 145 of file MissingDataCheck\_module.cc.

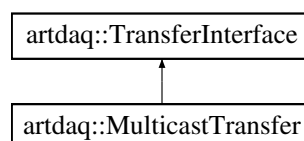
The documentation for this class was generated from the following file:

- artdaq/artdaq/ArtModules/MissingDataCheck\_module.cc

## 6.140 artdaq::MulticastTransfer Class Reference

[MulticastTransfer](#) is a [TransferInterface](#) implementation plugin that transfers data using Multicast.

Inheritance diagram for artdaq::MulticastTransfer:



## Public Types

- using `byte_t` = `artdaq::Fragment::byte_t`  
*Copy `Fragment::byte_t` into local scope.*

## Public Member Functions

- `~MulticastTransfer` () override=default  
*Default destructor.*
- `MulticastTransfer` (`fhicl::ParameterSet` const &`ps`, `Role` `role`)  
*`MulticastTransfer` Constructor.*
- `int receiveFragment` (`artdaq::Fragment` &`fragment`, `size_t` `receiveTimeout`) override  
*Receive a Fragment using Multicast.*
- `int receiveFragmentHeader` (`detail::RawFragmentHeader` &`header`, `size_t` `receiveTimeout`) override  
*Receive a Fragment Header from the transport mechanism.*
- `int receiveFragmentData` (`RawDataType` \*`destination`, `size_t` `wordCount`) override  
*Receive the body of a Fragment to the given destination pointer.*
- `CopyStatus transfer_fragment_min_blocking_mode` (`artdaq::Fragment` const &`fragment`, `size_t` `send_timeout_` - `usec`) override  
*Copy a Fragment to the destination. Multicast is always unreliable.*
- `CopyStatus transfer_fragment_reliable_mode` (`artdaq::Fragment` &&`fragment`) override  
*Move a Fragment to the destination. Multicast is always unreliable.*
- `bool isRunning` () override  
*Determine whether the `TransferInterface` plugin is able to send/receive data.*
- `void flush_buffers` () override  
*Flush any in-flight data. This should be used by the receiver after the receive loop has ended.*

## Additional Inherited Members

### 6.140.1 Detailed Description

`MulticastTransfer` is a `TransferInterface` implementation plugin that transfers data using Multicast.

Definition at line 28 of file `Multicast_transfer.cc`.

### 6.140.2 Constructor & Destructor Documentation

#### 6.140.2.1 `artdaq::MulticastTransfer::MulticastTransfer` ( `fhicl::ParameterSet` const & `ps`, `Role` `role` )

`MulticastTransfer` Constructor.

Parameters

<code>ps</code>	ParameterSet used to configure <code>MulticastTransfer</code>
<code>role</code>	Role of this <code>MulticastTransfer</code> instance (kSend or kReceive)

```
* MulticastTransfer accepts the following Parameters:
* "subfragment_size" (REQUIRED): Size of the sub-Fragments
* "subfragments_per_send" (REQUIRED): How many sub-Fragments to send in each batch
* "pause_on_copy_usecs" (Default: 0): Pause after sending a batch of sub-Fragments for this many microseconds
```

```
* "multicast_port" (REQUIRED): Port number to connect to
* "multicast_address" (REQUIRED): Multicast address to send to/receive from
* "local_address" (REQUIRED): Local origination address for multicast
* "receive_buffer_size" (Default: 0): The UDP receive buffer size. 0 uses automatic size.
*
```

[MulticastTransfer](#) also requires all Parameters for configuring a [TransferInterface](#)

Definition at line 167 of file Multicast\_transfer.cc.

### 6.140.3 Member Function Documentation

#### 6.140.3.1 bool artdaq::MulticastTransfer::isRunning ( ) [inline],[override],[virtual]

Determine whether the [TransferInterface](#) plugin is able to send/receive data.

##### Returns

True if the [TransferInterface](#) plugin is currently able to send/receive data

Reimplemented from [artdaq::TransferInterface](#).

Definition at line 101 of file Multicast\_transfer.cc.

#### 6.140.3.2 int artdaq::MulticastTransfer::receiveFragment ( artdaq::Fragment & *fragment*, size\_t *receiveTimeout* ) [override],[virtual]

Receive a Fragment using Multicast.

##### Parameters

out	<i>fragment</i>	Received Fragment
	<i>receiveTimeout</i>	<a href="#">Timeout</a> for receive, in microseconds

##### Returns

Rank of sender or RECV\_TIMEOUT

Reimplemented from [artdaq::TransferInterface](#).

Definition at line 251 of file Multicast\_transfer.cc.

#### 6.140.3.3 int artdaq::MulticastTransfer::receiveFragmentData ( RawDataType \* *destination*, size\_t *wordCount* ) [override],[virtual]

Receive the body of a Fragment to the given destination pointer.

##### Parameters

<i>destination</i>	Pointer to memory region where Fragment data should be stored
<i>wordCount</i>	Number of words of Fragment data to receive

**Returns**

The rank the Fragment was received from (should be `source_rank`), or `RECV_TIMEOUT`

Implements [artdaq::TransferInterface](#).

Definition at line 396 of file `Multicast_transfer.cc`.

**6.140.3.4** `int artdaq::MulticastTransfer::receiveFragmentHeader ( detail::RawFragmentHeader & header, size_t receiveTimeout )`  
`[override], [virtual]`

Receive a Fragment Header from the transport mechanism.

**Parameters**

<code>out</code>	<code>header</code>	Received Fragment Header
	<code>receiveTimeout</code>	<a href="#">Timeout</a> for receive

**Returns**

The rank the Fragment was received from (should be `source_rank`), or `RECV_TIMEOUT`

Implements [artdaq::TransferInterface](#).

Definition at line 385 of file `Multicast_transfer.cc`.

**6.140.3.5** `artdaq::TransferInterface::CopyStatus artdaq::MulticastTransfer::transfer_fragment_min_blocking_mode ( artdaq::Fragment const & fragment, size_t send_timeout_usec )`  
`[override], [virtual]`

Copy a Fragment to the destination. Multicast is always unreliable.

**Parameters**

<code>fragment</code>	Fragment to copy
<code>send_timeout_ - usec</code>	How long to try to send before discarding data

**Returns**

`CopyStatus` detailing result of copy

Implements [artdaq::TransferInterface](#).

Definition at line 415 of file `Multicast_transfer.cc`.

**6.140.3.6** `artdaq::TransferInterface::CopyStatus artdaq::MulticastTransfer::transfer_fragment_reliable_mode ( artdaq::Fragment && fragment )`  
`[override], [virtual]`

Move a Fragment to the destination. Multicast is always unreliable.

**Parameters**

<code>fragment</code>	Fragment to move
-----------------------	------------------

**Returns**

CopyStatus detailing result of copy

Implements [artdaq::TransferInterface](#).

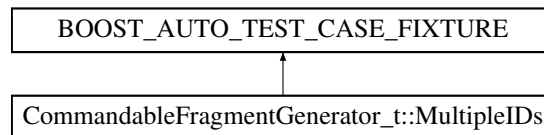
Definition at line 409 of file Multicast\_transfer.cc.

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

- artdaq/artdaq/TransferPlugins/Multicast\_transfer.cc

## 6.141 CommandableFragmentGenerator\_t::MultipleIDs Struct Reference

Inheritance diagram for CommandableFragmentGenerator\_t::MultipleIDs:

**Public Member Functions**

- void **test\_method** ()

### 6.141.1 Detailed Description

Definition at line 308 of file CommandableFragmentGenerator\_t.cc.

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

- artdaq/test/Generators/CommandableFragmentGenerator\_t.cc

## 6.142 CommandableFragmentGenerator\_t::MultipleIDs\_id Struct Reference

### 6.142.1 Detailed Description

Definition at line 308 of file CommandableFragmentGenerator\_t.cc.

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

- artdaq/test/Generators/CommandableFragmentGenerator\_t.cc

## 6.143 artdaq::NetMonHeader Struct Reference

Header with length information for NetMonTransport messages.

```
#include <artdaq/DAQdata/NetMonHeader.hh>
```

## Public Attributes

- uint64\_t [data\\_length](#) {0}  
*The length of the message.*

### 6.143.1 Detailed Description

Header with length information for NetMonTransport messages.

Definition at line 13 of file NetMonHeader.hh.

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

- artdaq/artdaq/DAQdata/NetMonHeader.hh

## 6.144 art::RootDAQOut::Config::NewSubStringForApp Struct Reference

### Public Attributes

- fhicl::Atom< string > **appName** {fhicl::Name("appName")}
- fhicl::Atom< string > **newString** {fhicl::Name("newString")}

### 6.144.1 Detailed Description

Definition at line 135 of file RootDAQOut\_module.cc.

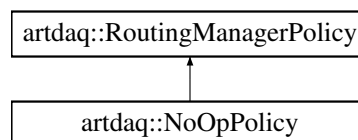
The documentation for this struct was generated from the following file:

- artdaq/artdaq/ArtModules/RootDAQOutput-s124/RootDAQOut\_module.cc

## 6.145 artdaq::NoOpPolicy Class Reference

A [RoutingManagerPolicy](#) which simply assigns Sequence IDs to tokens in the order they were received.

Inheritance diagram for artdaq::NoOpPolicy:



### Public Member Functions

- [NoOpPolicy](#) (fhicl::ParameterSet const &ps)  
*NoOpPolicy Constructor.*
- [~NoOpPolicy](#) () override=default  
*Default virtual Destructor.*

- void [CreateRoutingTable](#) ([detail::RoutingPacket](#) &table) override  
*Add entries to the given RoutingPacket using currently-held tokens.*
- [detail::RoutingPacketEntry CreateRouteForSequenceID](#) (artdaq::Fragment::sequence\_id\_t seq, int requesting\_rank) override  
*Get an [artdaq::detail::RoutingPacketEntry](#) for a given sequence ID and rank. Used by RequestBasedEventBuilder and DataFlow RoutingManagerMode.*

## Additional Inherited Members

### 6.145.1 Detailed Description

A [RoutingManagerPolicy](#) which simply assigns Sequence IDs to tokens in the order they were received.

Definition at line 15 of file NoOp\_policy.cc.

### 6.145.2 Constructor & Destructor Documentation

6.145.2.1 [artdaq::NoOpPolicy::NoOpPolicy](#) ( [fhicl::ParameterSet](#) const & *ps* ) [\[inline\]](#), [\[explicit\]](#)

[NoOpPolicy](#) Constructor.

Parameters

<i>ps</i>	ParameterSet used to configure the <a href="#">NoOpPolicy</a>
-----------	---

[NoOpPolicy](#) takes no additional Parameters at this time

Definition at line 24 of file NoOp\_policy.cc.

### 6.145.3 Member Function Documentation

6.145.3.1 [detail::RoutingPacketEntry](#) [artdaq::NoOpPolicy::CreateRouteForSequenceID](#) ( [artdaq::Fragment::sequence\\_id\\_t](#) *seq*, [int](#) *requesting\_rank* ) [\[override\]](#), [\[virtual\]](#)

Get an [artdaq::detail::RoutingPacketEntry](#) for a given sequence ID and rank. Used by RequestBasedEventBuilder and DataFlow RoutingManagerMode.

Parameters

<i>seq</i>	Sequence Number to get route for
<i>requesting_rank</i>	Rank to route for

Returns

[artdaq::detail::RoutingPacketEntry](#) connecting sequence ID to destination rank

Implements [artdaq::RoutingManagerPolicy](#).

Definition at line 67 of file NoOp\_policy.cc.

6.145.3.2 void [artdaq::NoOpPolicy::CreateRoutingTable](#) ( [detail::RoutingPacket](#) & *table* ) [\[override\]](#), [\[virtual\]](#)

Add entries to the given RoutingPacket using currently-held tokens.

## Parameters

<i>table</i>	RoutingPacket to add entries to
--------------	---------------------------------

NoOp\_policy will add entries for all tokens in the order that they were received

Implements [artdaq::RoutingManagerPolicy](#).

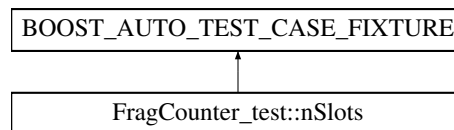
Definition at line 56 of file NoOp\_policy.cc.

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

- [artdaq/artdaq/RoutingPolicies/NoOp\\_policy.cc](#)

## 6.146 FragCounter\_test::nSlots Struct Reference

Inheritance diagram for FragCounter\_test::nSlots:



### Public Member Functions

- void **test\_method** ()

#### 6.146.1 Detailed Description

Definition at line 16 of file FragCounter\_t.cc.

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

- [artdaq/test/DAQrate/FragCounter\\_t.cc](#)

## 6.147 FragCounter\_test::nSlots\_id Struct Reference

#### 6.147.1 Detailed Description

Definition at line 16 of file FragCounter\_t.cc.

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

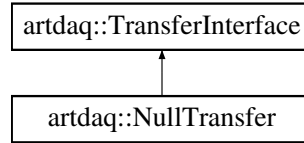
- [artdaq/test/DAQrate/FragCounter\\_t.cc](#)

## 6.148 artdaq::NullTransfer Class Reference

[NullTransfer](#) does not send or receive data, but acts as if it did.



Inheritance diagram for artdaq::NullTransfer:



## Public Member Functions

- [NullTransfer](#) (const fhicl::ParameterSet &pset, [Role](#) role)  
*NullTransfer* constructor.
- [~NullTransfer](#) () override=default  
*NullTransfer* default Destructor.
- int [receiveFragment](#) (artdaq::Fragment &, size\_t) override  
*Pretend to receive a Fragment.*
- int [receiveFragmentHeader](#) (detail::RawFragmentHeader &, size\_t) override  
*Pretend to receive a Fragment Header.*
- int [receiveFragmentData](#) (RawDataType \*, size\_t) override  
*Pretend to receive Fragment Data.*
- [CopyStatus](#) transfer\_fragment\_min\_blocking\_mode (artdaq::Fragment const &, size\_t) override  
*Pretend to send a Fragment to a destination.*
- [CopyStatus](#) transfer\_fragment\_reliable\_mode (artdaq::Fragment &&) override  
*Pretend to send a Fragment to a destination.*
- bool [isRunning](#) () override  
*Determine whether the [TransferInterface](#) plugin is able to send/receive data.*
- void [flush\\_buffers](#) () override  
*Flush any in-flight data. This should be used by the receiver after the receive loop has ended.*

## Additional Inherited Members

### 6.148.1 Detailed Description

[NullTransfer](#) does not send or receive data, but acts as if it did.

Definition at line 7 of file Null\_transfer.cc.

### 6.148.2 Constructor & Destructor Documentation

#### 6.148.2.1 artdaq::NullTransfer::NullTransfer ( const fhicl::ParameterSet & pset, [Role](#) role )

[NullTransfer](#) constructor.

Parameters

<i>pset</i>	ParameterSet used to configure <a href="#">TransferInterface</a>
<i>role</i>	Role of this <a href="#">NullTransfer</a> instance (kSend or kReceive)

[NullTransfer](#) only requires the Parameters for configuring a [TransferInterface](#)

Definition at line 86 of file Null\_transfer.cc.

### 6.148.3 Member Function Documentation

#### 6.148.3.1 `bool artdaq::NullTransfer::isRunning ( ) [inline],[override],[virtual]`

Determine whether the [TransferInterface](#) plugin is able to send/receive data.

##### Returns

True if the [TransferInterface](#) plugin is currently able to send/receive data

Reimplemented from [artdaq::TransferInterface](#).

Definition at line 70 of file `Null_transfer.cc`.

#### 6.148.3.2 `int artdaq::NullTransfer::receiveFragment ( artdaq::Fragment & , size_t ) [inline],[override],[virtual]`

Pretend to receive a Fragment.

##### Returns

Source Rank (Success code)

WARNING: This function may create unintended side-effets. [NullTransfer](#) should only really be used in [Role::kSend!](#)

Reimplemented from [artdaq::TransferInterface](#).

Definition at line 31 of file `Null_transfer.cc`.

#### 6.148.3.3 `int artdaq::NullTransfer::receiveFragmentData ( RawDataType * , size_t ) [inline],[override],[virtual]`

Pretend to receive Fragment Data.

##### Returns

Source Rank (Success code)

WARNING: This function may create unintended side-effets. [NullTransfer](#) should only really be used in [Role::kSend!](#)

Implements [artdaq::TransferInterface](#).

Definition at line 49 of file `Null_transfer.cc`.

#### 6.148.3.4 `int artdaq::NullTransfer::receiveFragmentHeader ( detail::RawFragmentHeader & , size_t ) [inline],[override],[virtual]`

Pretend to receive a Fragment Header.

##### Returns

Source Rank (Success code)

WARNING: This function may create unintended side-effets. [NullTransfer](#) should only really be used in [Role::kSend!](#)

Implements [artdaq::TransferInterface](#).

Definition at line 40 of file `Null_transfer.cc`.

6.148.3.5 **CopyStatus** artdaq::NullTransfer::transfer\_fragment\_min\_blocking\_mode ( artdaq::Fragment const & , size\_t )  
 [inline], [override], [virtual]

Pretend to send a Fragment to a destination.

Returns

[CopyStatus::kSuccess](#) (No-Op)

Implements [artdaq::TransferInterface](#).

Definition at line 55 of file Null\_transfer.cc.

6.148.3.6 **CopyStatus** artdaq::NullTransfer::transfer\_fragment\_reliable\_mode ( artdaq::Fragment && ) [inline],  
 [override], [virtual]

Pretend to send a Fragment to a destination.

Returns

[CopyStatus::kSuccess](#) (No-Op)

Implements [artdaq::TransferInterface](#).

Definition at line 64 of file Null\_transfer.cc.

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

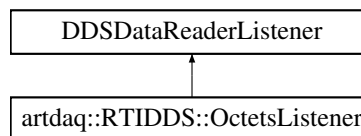
- artdaq/artdaq/TransferPlugins/Null\_transfer.cc

## 6.149 artdaq::RTIDDS::OctetsListener Class Reference

A class that reads data from DDS.

```
#include <artdaq/RTIDDS/RTIDDS.hh>
```

Inheritance diagram for artdaq::RTIDDS::OctetsListener:



### Public Member Functions

- void [on\\_data\\_available](#) (DDSDataReader \*reader)  
*Action to perform when data is available.*
- bool [receiveFragmentFromDDS](#) (artdaq::Fragment &fragment, const size\_t receiveTimeout)  
*Receive a Fragment from DDS.*

### 6.149.1 Detailed Description

A class that reads data from DDS.

Definition at line 68 of file RTIDDS.hh.

### 6.149.2 Member Function Documentation

#### 6.149.2.1 void artdaq::RTIDDS::OctetsListener::on\_data\_available ( DDSDataReader \* *reader* )

Action to perform when data is available.

Parameters

<i>reader</i>	Reader reference to read data from
---------------	------------------------------------

Definition at line 161 of file RTIDDS.cc.

#### 6.149.2.2 bool artdaq::RTIDDS::OctetsListener::receiveFragmentFromDDS ( artdaq::Fragment & *fragment*, const size\_t *receiveTimeout* )

Receive a Fragment from DDS.

Parameters

out	<i>fragment</i>	Received Fragment
	<i>receiveTimeout</i>	<a href="#">Timeout</a> for receive operation

Returns

Whether the receive succeeded in receiveTimeout

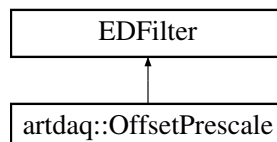
Definition at line 211 of file RTIDDS.cc.

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

- artdaq/artdaq/RTIDDS/RTIDDS.hh
- artdaq/artdaq/RTIDDS/RTIDDS.cc

## 6.150 artdaq::OffsetPrescale Class Reference

Inheritance diagram for artdaq::OffsetPrescale:



### Public Member Functions

- **OffsetPrescale** (fhicl::ParameterSet const &p)

- **OffsetPrescale** ([OffsetPrescale](#) const &)=delete
- **OffsetPrescale** ([OffsetPrescale](#) &&)=delete
- [OffsetPrescale](#) & **operator=** ([OffsetPrescale](#) const &)=delete
- [OffsetPrescale](#) & **operator=** ([OffsetPrescale](#) &&)=delete
- bool **filter** (art::Event &e) override

### 6.150.1 Detailed Description

Definition at line 29 of file OffsetPrescale\_module.cc.

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

- artdaq/artdaq/ArtModules/OffsetPrescale\_module.cc

## 6.151 art::OutputItem Struct Reference

### Public Member Functions

- **OutputItem** (BranchDescription const &bd)

### Public Attributes

- BranchDescription const **branchDescription**
- void const \* **product**

### 6.151.1 Detailed Description

Definition at line 37 of file RootDAQOutFile.h.

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

- artdaq/artdaq/ArtModules/RootDAQOutput-s124/RootDAQOutFile.h
- artdaq/artdaq/ArtModules/RootDAQOutput-s124/RootDAQOutFile.cc

## 6.152 art::OutputsConfig Struct Reference

Configuration for the outputs block of artdaq art processes

```
#include <artdaq/ArtModules/detail/ArtConfig.hh>
```

### Public Attributes

- fhicl::OptionalTable  
 < art::OutputModule::Config > [rootNetOutput](#) {fhicl::Name{"rootNetOutput"}}  
*For transferring data from EventBuilders to DataLoggers.*
- fhicl::OptionalTable  
 < [RootOutputConfig](#) > [normalOutput](#) {fhicl::Name{"normalOutput"}}

*Normal art/ROOT output.*

- fhicl::OptionalTable  
 < [RootOutputConfig](#) > [rootDAQOutFile](#) {fhicl::Name{"rootDAQOutFile"}}  
*art/ROOT output where the filename can be specified at initialization (e.g. to /dev/null), for testing*

### 6.152.1 Detailed Description

Configuration for the outputs block of artdaq art processes

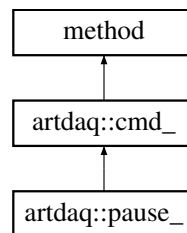
Definition at line 129 of file ArtConfig.hh.

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

- artdaq/artdaq/ArtModules/detail/ArtConfig.hh

## 6.153 artdaq::pause\_ Class Reference

Inheritance diagram for artdaq::pause\_:



### Public Member Functions

- [pause\\_](#) (xmlrpc\_commander &c)

### Static Public Attributes

- static const uint64\_t [defaultTimeout](#) = 45
- static const uint64\_t [defaultTimestamp](#) = std::numeric\_limits<const uint64\_t>::max()

### Additional Inherited Members

#### 6.153.1 Detailed Description

[pause\\_](#) Command class

Definition at line 920 of file xmlrpc\_commander.cc.

#### 6.153.2 Constructor & Destructor Documentation

6.153.2.1 artdaq::pause\_::pause\_( xmlrpc\_commander & c ) [inline]

[pause\\_](#) Constructor \

## Parameters

<code>c</code>	<code>xmlrpc_commander</code> to send transition commands to
----------------	--

Definition at line 920 of file `xmlrpc_commander.cc`.

### 6.153.3 Member Data Documentation

6.153.3.1 `const uint64_t artdaq::pause_::defaultTimeout = 45` `[static]`

Default timeout for command

Definition at line 920 of file `xmlrpc_commander.cc`.

6.153.3.2 `const uint64_t artdaq::pause_::defaultTimestamp = std::numeric_limits<const uint64_t>::max()` `[static]`

Default timestamp for Command

Definition at line 920 of file `xmlrpc_commander.cc`.

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

- `artdaq/artdaq/ExternalComms/xmlrpc_commander.cc`

## 6.154 art::PhysicsConfig Struct Reference

Configuration of the physics block for artdaq art processes

```
#include <artdaq/ArtModules/detail/ArtConfig.hh>
```

### Public Attributes

- `fhicl::Table< AnalyzersConfig > analyzers` `{fhicl::Name{"analyzers"}}`  
*Analyzer module configuration.*
- `fhicl::Table< ProducersConfig > producers` `{fhicl::Name{"producers"}}`  
*Producer module configuration.*
- `fhicl::Table< FiltersConfig > filters` `{fhicl::Name{"filters"}}`
- `fhicl::Sequence< std::string > my_output_modules` `{fhicl::Name{"my_output_modules"}, fhicl::Comment{"Output modules (configured in the outputs block) to use"}}`  
*Output modules (configured in the outputs block) to use.*

### 6.154.1 Detailed Description

Configuration of the physics block for artdaq art processes

Definition at line 69 of file `ArtConfig.hh`.



## 6.154.2 Member Data Documentation

### 6.154.2.1 fhicl::Table<FiltersConfig> art::PhysicsConfig::filters {fhicl::Name{"filters"}}

Filter module configuration

Definition at line 73 of file ArtConfig.hh.

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

- artdaq/artdaq/ArtModules/detail/ArtConfig.hh

## 6.155 artdaq::PortManager Class Reference

[PortManager](#) attempts to automatically detect interfaces and ports used for the various TCP and UDP sockets used by artdaq.

```
#include <artdaq/DAQdata/PortManager.hh>
```

### Classes

- struct [Config](#)  
*Configuration of [PortManager](#). May be used for parameter validation*

### Public Types

- using [Parameters](#) = fhicl::WrappedTable< [Config](#) >  
*Used for ParameterSet validation (if desired)*

### Public Member Functions

- [PortManager](#) ()  
*[PortManager](#) Construator.*
- void [UpdateConfiguration](#) (fhicl::ParameterSet const &ps)  
*Override the default configuration.*
- int [GetRoutingTokenPort](#) (int subsystemID=0)  
*Get the port that should be used for Routing Tokens.*
- int [GetRoutingAckPort](#) (int subsystemID=0)  
*Get the port that should be used for Routing Acknowledgements.*
- int [GetXMLRPCPort](#) (int rank)  
*Get the XMLRPC port for the given rank.*
- int [GetTCPSocketTransferPort](#) (int rank)  
*Get the TCP Socket transfer port for the given rank.*
- int [GetRequestMessagePort](#) ()  
*Get the port that should be used for multicast request messages.*
- std::string [GetRequestMessageGroupAddress](#) (int subsystemID=0)  
*Get the multicast address for request messages.*
- int [GetRoutingTablePort](#) ()

*Get the port that should be used for multicast Routing Tables.*

- `std::string GetRoutingTableGroupAddress (int subsystemID=0)`

*Get the multicast address for Routing Tables.*

- `int GetMulticastTransferPort (int rank)`

*Get the multicast transfer port for the given rank.*

- `std::string GetMulticastTransferGroupAddress ()`

*Get the multicast address for multicast transfers.*

- `in_addr GetMulticastOutputAddress (const std::string &interface_name="", const std::string &interface_address="")`

*Determine the output interface address, using the hints provided.*

### 6.155.1 Detailed Description

`PortManager` attempts to automatically detect interfaces and ports used for the various TCP and UDP sockets used by `artdaq`.

Definition at line 30 of file `PortManager.hh`.

### 6.155.2 Member Function Documentation

- 6.155.2.1 `in_addr artdaq::PortManager::GetMulticastOutputAddress ( const std::string & interface_name = " ", const std::string & interface_address = " " )`

Determine the output interface address, using the hints provided.

#### Parameters

<i>interface_name</i>	If set, the name of the interface that should be used for multicast (e.g. "eth0"). Default: ""
<i>interface_ - address</i>	If set, the address of the interface that should be used for multicast (e.g. 192.168.0.1). Default: ""

#### Returns

`in_addr` struct populated with selected interface's info

If neither `interface_name` or `interface_address` are set, then the interface will be auto-detected, giving preference to private network addresses.

Definition at line 356 of file `PortManager.cc`.

- 6.155.2.2 `std::string artdaq::PortManager::GetMulticastTransferGroupAddress ( )`

Get the multicast address for multicast transfers.

#### Returns

Multicast address for multicast transfers

Definition at line 345 of file `PortManager.cc`.

- 6.155.2.3 `int artdaq::PortManager::GetMulticastTransferPort ( int rank )`

Get the multicast transfer port for the given rank.

## Parameters

<i>rank</i>	Rank to get multicast transfer port for
-------------	---

## Returns

multicast transfer port for the given rank

Definition at line 334 of file PortManager.cc.

#### 6.155.2.4 std::string artdaq::PortManager::GetRequestMessageGroupAddress ( int *subsystemID* = 0 )

Get the multicast address for request messages.

## Parameters

<i>subsystemID</i>	Subsystem that this artdaq process belongs to
--------------------	---

## Returns

Multicast address for request messages

Definition at line 301 of file PortManager.cc.

#### 6.155.2.5 int artdaq::PortManager::GetRequestMessagePort ( )

Get the port that should be used for multicast request messages.

## Returns

Port used for multicast request messages

Definition at line 291 of file PortManager.cc.

#### 6.155.2.6 int artdaq::PortManager::GetRoutingAckPort ( int *subsystemID* = 0 )

Get the port that should be used for Routing Acknowledgements.

## Parameters

<i>subsystemID</i>	Subsystem that this artdaq process belongs to
--------------------	---

## Returns

Port number for Routing Acknowledgements

Definition at line 261 of file PortManager.cc.

#### 6.155.2.7 std::string artdaq::PortManager::GetRoutingTableGroupAddress ( int *subsystemID* = 0 )

Get the multicast address for Routing Tables.

**Parameters**

<i>subsystemID</i>	Subsystem that this artdaq process belongs to
--------------------	---

**Returns**

Multicast address for Routing Tables

Definition at line 323 of file PortManager.cc.

**6.155.2.8 int artdaq::PortManager::GetRoutingTablePort ( )**

Get the port that should be used for multicast Routing Tables.

**Returns**

Port used for multicast Routing Tables

Definition at line 312 of file PortManager.cc.

**6.155.2.9 int artdaq::PortManager::GetRoutingTokenPort ( int *subsystemID* = 0 )**

Get the port that should be used for Routing Tokens.

**Parameters**

<i>subsystemID</i>	Subsystem that this artdaq process belongs to
--------------------	---

**Returns**

Port number for Routing Tokens

Definition at line 250 of file PortManager.cc.

**6.155.2.10 int artdaq::PortManager::GetTCPSocketTransferPort ( int *rank* )**

Get the TCP Socket transfer port for the given rank.

**Parameters**

<i>rank</i>	Rank to get TCP Socket transfer port for
-------------	--

**Returns**

TCP Socket transfer port for the given rank

Definition at line 281 of file PortManager.cc.

**6.155.2.11 int artdaq::PortManager::GetXMLRPCPort ( int *rank* )**

Get the XMLRPC port for the given rank.

## Parameters

<i>rank</i>	Rank to get XMLRPC port for
-------------	-----------------------------

## Returns

XMLRPC port for the given rank

Definition at line 271 of file PortManager.cc.

6.155.2.12 void artdaq::PortManager::UpdateConfiguration ( fhicl::ParameterSet const & *ps* )

Override the default configuration.

## Parameters

<i>ps</i>	ParameterSet containing overridden parameters
-----------	---

Definition at line 22 of file PortManager.cc.

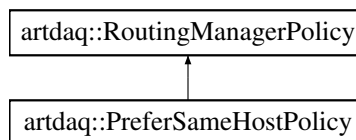
The documentation for this class was generated from the following files:

- artdaq/artdaq/DAQdata/PortManager.hh
- artdaq/artdaq/DAQdata/PortManager.cc

## 6.156 artdaq::PreferSameHostPolicy Class Reference

A [RoutingManagerPolicy](#) which tries to keep data on the same host. For EventBuilding mode, performs RoundRobin.

Inheritance diagram for artdaq::PreferSameHostPolicy:



### Public Member Functions

- [PreferSameHostPolicy](#) (const fhicl::ParameterSet &ps)  
*PreferSameHostPolicy Constructor.*
- [~PreferSameHostPolicy](#) () override=default  
*Default virtual Destructor.*
- void [CreateRoutingTable](#) (detail::RoutingPacket &output) override  
*Generate a set of Routing Tables using received tokens.*
- detail::RoutingPacketEntry [CreateRouteForSequenceID](#) (artdaq::Fragment::sequence\_id\_t seq, int requesting\_rank) override  
*Get an [artdaq::detail::RoutingPacketEntry](#) for a given sequence ID and rank. Used by RequestBasedEventBuilder and DataFlow RoutingManagerMode.*

## Additional Inherited Members

### 6.156.1 Detailed Description

A [RoutingManagerPolicy](#) which tries to keep data on the same host. For EventBuilding mode, performs RoundRobin.

Definition at line 17 of file `PreferSameHost_policy.cc`.

### 6.156.2 Constructor & Destructor Documentation

6.156.2.1 `artdaq::PreferSameHostPolicy::PreferSameHostPolicy ( const fhicl::ParameterSet & ps )` `[inline]`, `[explicit]`

[PreferSameHostPolicy](#) Constructor.

Parameters

<i>ps</i>	ParameterSet used to configure <a href="#">PreferSameHostPolicy</a>
-----------	---

[PreferSameHostPolicy](#) accepts the following Parameter: "minimum\_participants" (Default: 0): Minimum number of receivers to distribute between. Use negative number to indicate how many can be missing from total. If the number of allowed missing receivers is greater than the number that exist, then the minimum number of participants will be set to 1.

Definition at line 27 of file `PreferSameHost_policy.cc`.

### 6.156.3 Member Function Documentation

6.156.3.1 `detail::RoutingPacketEntry` `artdaq::PreferSameHostPolicy::CreateRouteForSequenceID (`  
`artdaq::Fragment::sequence_id_t seq, int requesting_rank )` `[override]`, `[virtual]`

Get an [artdaq::detail::RoutingPacketEntry](#) for a given sequence ID and rank. Used by RequestBasedEventBuilder and DataFlow RoutingManagerMode.

Parameters

<i>seq</i>	Sequence Number to get route for
<i>requesting_rank</i>	Rank to route for

Returns

[artdaq::detail::RoutingPacketEntry](#) connecting sequence ID to destination rank

Implements [artdaq::RoutingManagerPolicy](#).

Definition at line 105 of file `PreferSameHost_policy.cc`.

6.156.3.2 `void` `artdaq::PreferSameHostPolicy::CreateRoutingTable (` `detail::RoutingPacket & output )` `[override]`,  
`[virtual]`

Generate a set of Routing Tables using received tokens.

## Parameters

<i>output</i>	The RoutingPacket to add entries to
---------------	-------------------------------------

[PreferSameHostPolicy](#) will go through the list of receivers as many times as it can, until one or more receivers have no tokens. It always does full "turns" through the receiver list.

Implements [artdaq::RoutingManagerPolicy](#).

Definition at line 70 of file `PreferSameHost_policy.cc`.

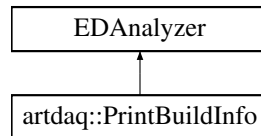
The documentation for this class was generated from the following file:

- `artdaq/artdaq/RoutingPolicies/PreferSameHost_policy.cc`

## 6.157 artdaq::PrintBuildInfo Class Reference

An `art::EDAnalyzer` which prints any [artdaq::BuildInfo](#) objects stored in the run.

Inheritance diagram for `artdaq::PrintBuildInfo`:



### Public Member Functions

- [PrintBuildInfo](#) (`fhicl::ParameterSet const &p`)  
*PrintBuildInfo Constructor.*
- [~PrintBuildInfo](#) () `override=default`  
*Default virtual Destructor.*
- `void analyze` (`art::Event const &`) `override`  
*Called for each event. Required overload for art::EDAnalyzer, No-Op here.*
- `void beginRun` (`art::Run const &run`) `override`  
*Perform actions at the beginning of the run.*

### 6.157.1 Detailed Description

An `art::EDAnalyzer` which prints any [artdaq::BuildInfo](#) objects stored in the run.

Definition at line 34 of file `PrintBuildInfo_module.cc`.

### 6.157.2 Constructor & Destructor Documentation

6.157.2.1 `artdaq::PrintBuildInfo::PrintBuildInfo ( fhicl::ParameterSet const & p )` `[explicit]`

[PrintBuildInfo](#) Constructor.

**Parameters**

<i>p</i>	ParameterSet used to configure <a href="#">PrintBuildInfo</a>
----------	---

```
* PrintBuildInfo accepts the following Parameters:
* "buildinfo_module_label" (REQUIRED): The module label for the BuildInfo objects
* "buildinfo_instance_label" (REQUIRED): The instance label for the BuildInfo objects
*
```

These parameters should match those given to the [BuildInfo](#) module

Definition at line 80 of file PrintBuildInfo\_module.cc.

**6.157.3 Member Function Documentation****6.157.3.1 void artdaq::PrintBuildInfo::beginRun ( art::Run const & run ) [override]**

Perform actions at the beginning of the run.

**Parameters**

<i>run</i>	art::Run object
------------	-----------------

This function pretty-prints the [BuildInfo](#) information from the run object with the configured module label and instance label.

Definition at line 85 of file PrintBuildInfo\_module.cc.

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

- artdaq/artdaq/ArtModules/PrintBuildInfo\_module.cc

**6.158 art::ProducersConfig Struct Reference**

Artdaq does not provide any producers.

```
#include <artdaq/ArtModules/detail/ArtConfig.hh>
```

**6.158.1 Detailed Description**

Artdaq does not provide any producers.

Definition at line 61 of file ArtConfig.hh.

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

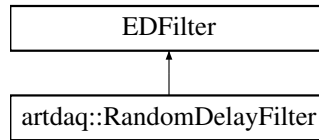
- artdaq/artdaq/ArtModules/detail/ArtConfig.hh

**6.159 artdaq::RandomDelayFilter Class Reference**

A filter which delays for a random amount of time, then drops a random fraction of events. Used to simulate the delays and efficiency of real filters.

Inheritance diagram for artdaq::RandomDelayFilter:





## Public Member Functions

- [RandomDelayFilter](#) (fhicl::ParameterSet const &p)  
*RandomDelayFilter Constructor.*
- [RandomDelayFilter](#) ([RandomDelayFilter](#) const &)=delete  
*Copy Constructor is deleted.*
- [RandomDelayFilter](#) ([RandomDelayFilter](#) &&)=delete  
*Move Constructor is deleted.*
- [RandomDelayFilter](#) & operator= ([RandomDelayFilter](#) const &)=delete  
*Copy Assignment operator is deleted.*
- [RandomDelayFilter](#) & operator= ([RandomDelayFilter](#) &&)=delete  
*Move Assignment operator is deleted.*
- bool [filter](#) (art::Event &e) override  
*Filter is a required override of art::EDFilter, and is called for each event.*

### 6.159.1 Detailed Description

A filter which delays for a random amount of time, then drops a random fraction of events. Used to simulate the delays and efficiency of real filters.

Multiple RandomDelayFilters in series can simulate the effect of multiple layers of filtering

Definition at line 39 of file RandomDelayFilter\_module.cc.

### 6.159.2 Constructor & Destructor Documentation

#### 6.159.2.1 artdaq::RandomDelayFilter::RandomDelayFilter ( fhicl::ParameterSet const & p ) [explicit]

[RandomDelayFilter](#) Constructor.

Parameters

<i>p</i>	ParameterSet used to configure <a href="#">RandomDelayFilter</a>
----------	--

[RandomDelayFilter](#) accepts the following Parameters: "minimum\_delay\_ms" (Default: 0): The minimum amount of time to delay, in ms "maximum\_delay\_ms" (Default: 1000): The maximum amount of time to delay, in ms "mean\_delay\_ms" (Default: 500): If using a non-uniform distribution for delay times, the mean of the distribution, in ms (This value will be used for Fixed delays) "sigma\_delay\_ms" (Default: 100): If using a Normal distribution for delay times, the sigma of the distribution, in ms "pass\_filter\_percentage" (Default: 100): The fraction of events which will pass the filter "distribution\_type" (Default: "Uniform"): The distribution to sample for delays (Uniform, Normal, Exponential, Fixed) "cpu\_load\_ratio" (Default: 0.5): The fraction of the delay time which should be active (spinning) versus passive (sleeping) "random\_seed" (Default: time(0)): The seed for the distribution

Definition at line 120 of file RandomDelayFilter\_module.cc.

### 6.159.3 Member Function Documentation

6.159.3.1 `bool artdaq::RandomDelayFilter::filter ( art::Event & e ) [override]`

Filter is a required override of `art::EDFilter`, and is called for each event.

## Parameters

<i>e</i>	The art::Event to filter
----------	--------------------------

## Returns

Whether the event passes the filter

This function is where [RandomDelayFilter](#) performs its work, using the delay distribution to pick a delay time, spinning and/or sleeping for that amount of time, then picking an integer from the pass distribution to determine if the event should pass or not.

Definition at line 213 of file RandomDelayFilter\_module.cc.

### 6.159.3.2 RandomDelayFilter& artdaq::RandomDelayFilter::operator= ( RandomDelayFilter const & ) [delete]

Copy Assignment operator is deleted.

## Returns

[RandomDelayFilter](#) copy

### 6.159.3.3 RandomDelayFilter& artdaq::RandomDelayFilter::operator= ( RandomDelayFilter && ) [delete]

Move Assignment operator is deleted.

## Returns

[RandomDelayFilter](#) instance

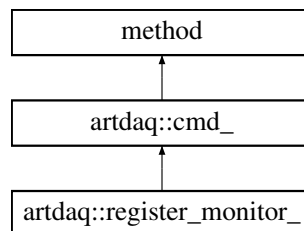
The documentation for this class was generated from the following file:

- artdaq/artdaq/ArtModules/RandomDelayFilter\_module.cc

## 6.160 artdaq::register\_monitor\_ Class Reference

[register\\_monitor\\_](#) Command class

Inheritance diagram for artdaq::register\_monitor\_:



## Public Member Functions

- [register\\_monitor\\_](#) (xmlrpc\_commander &c)  
*register\_monitor\_* Constructor

## Additional Inherited Members

### 6.160.1 Detailed Description

[register\\_monitor\\_](#) Command class

Definition at line 1052 of file xmlrpc\_commander.cc.

### 6.160.2 Constructor & Destructor Documentation

6.160.2.1 `artdaq::register_monitor_::register_monitor_ ( xmlrpc_commander & c )` `[inline]`

[register\\_monitor\\_](#) Constructor

Parameters

<code>c</code>	<a href="#">xmlrpc_commander</a> to send transition commands to
----------------	---

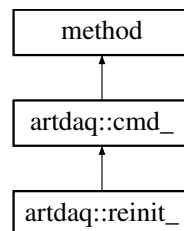
Definition at line 1059 of file xmlrpc\_commander.cc.

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

- `artdaq/artdaq/ExternalComms/xmlrpc_commander.cc`

## 6.161 artdaq::reinit\_ Class Reference

Inheritance diagram for `artdaq::reinit_`:



## Public Member Functions

- [reinit\\_](#) (`xmlrpc_commander` &c)

## Static Public Attributes

- static const uint64\_t [defaultTimeout](#) = 45
- static const uint64\_t [defaultTimestamp](#) = std::numeric\_limits<const uint64\_t>::max()

## Additional Inherited Members

### 6.161.1 Detailed Description

Command class representing an init transition

Definition at line 845 of file xmlrpc\_commander.cc.

## 6.161.2 Constructor & Destructor Documentation

6.161.2.1 `artdaq::reinit_::reinit_ ( xmlrpc_commander & c ) [inline], [explicit]`

Command class Constructor \ \*

Parameters

<code>c</code>	<code>xmlrpc_commander</code> to send parsed command to \
----------------	---

Definition at line 845 of file xmlrpc\_commander.cc.

## 6.161.3 Member Data Documentation

6.161.3.1 `const uint64_t artdaq::reinit_::defaultTimeout = 45 [static]`

Default timeout for command

Definition at line 845 of file xmlrpc\_commander.cc.

6.161.3.2 `const uint64_t artdaq::reinit_::defaultTimestamp = std::numeric_limits<const uint64_t>::max() [static]`

Default timestamp for Command

Definition at line 845 of file xmlrpc\_commander.cc.

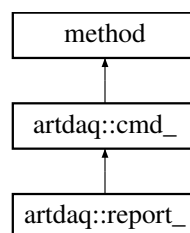
The documentation for this class was generated from the following file:

- `artdaq/artdaq/ExternalComms/xmlrpc_commander.cc`

## 6.162 artdaq::report\_ Class Reference

`report_` Command class

Inheritance diagram for `artdaq::report_`:



## Public Member Functions

- `report_ (xmlrpc_commander &c)`  
*report\_ Constructor*

## Additional Inherited Members

### 6.162.1 Detailed Description

[report\\_](#) Command class

Definition at line 986 of file xmlrpc\_commander.cc.

### 6.162.2 Constructor & Destructor Documentation

6.162.2.1 `artdaq::report_::report_ ( xmlrpc_commander & c ) [inline]`

[report\\_](#) Constructor

Parameters

<code>c</code>	<a href="#">xmlrpc_commander</a> to send transition commands to
----------------	---

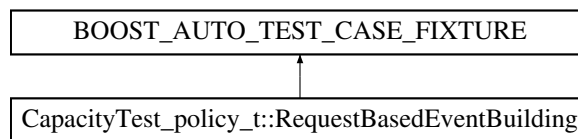
Definition at line 993 of file xmlrpc\_commander.cc.

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

- `artdaq/artdaq/ExternalComms/xmlrpc_commander.cc`

## 6.163 CapacityTest\_policy\_t::RequestBasedEventBuilding Struct Reference

Inheritance diagram for CapacityTest\_policy\_t::RequestBasedEventBuilding:



## Public Member Functions

- `void test_method ()`

### 6.163.1 Detailed Description

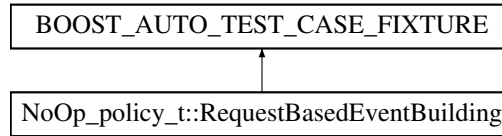
Definition at line 180 of file CapacityTest\_policy\_t.cc.

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

- `artdaq/test/RoutingPolicies/CapacityTest_policy_t.cc`

## 6.164 NoOp\_policy\_t::RequestBasedEventBuilding Struct Reference

Inheritance diagram for NoOp\_policy\_t::RequestBasedEventBuilding:



### Public Member Functions

- void **test\_method** ()

#### 6.164.1 Detailed Description

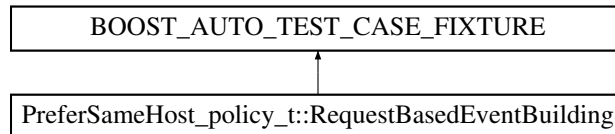
Definition at line 95 of file NoOp\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/NoOp\_policy\_t.cc

## 6.165 PreferSameHost\_policy\_t::RequestBasedEventBuilding Struct Reference

Inheritance diagram for PreferSameHost\_policy\_t::RequestBasedEventBuilding:



### Public Member Functions

- void **test\_method** ()

#### 6.165.1 Detailed Description

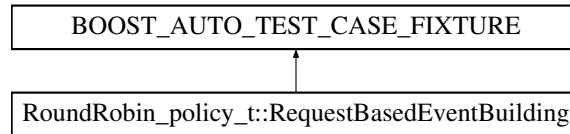
Definition at line 218 of file PreferSameHost\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/PreferSameHost\_policy\_t.cc

## 6.166 RoundRobin\_policy\_t::RequestBasedEventBuilding Struct Reference

Inheritance diagram for RoundRobin\_policy\_t::RequestBasedEventBuilding:



## Public Member Functions

- void **test\_method** ()

### 6.166.1 Detailed Description

Definition at line 245 of file RoundRobin\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/RoundRobin\_policy\_t.cc

## 6.167 CapacityTest\_policy\_t::RequestBasedEventBuilding\_id Struct Reference

### 6.167.1 Detailed Description

Definition at line 180 of file CapacityTest\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/CapacityTest\_policy\_t.cc

## 6.168 PreferSameHost\_policy\_t::RequestBasedEventBuilding\_id Struct Reference

### 6.168.1 Detailed Description

Definition at line 218 of file PreferSameHost\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/PreferSameHost\_policy\_t.cc

## 6.169 RoundRobin\_policy\_t::RequestBasedEventBuilding\_id Struct Reference

### 6.169.1 Detailed Description

Definition at line 245 of file RoundRobin\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/RoundRobin\_policy\_t.cc



## 6.170 NoOp\_policy\_t::RequestBasedEventBuilding\_id Struct Reference

### 6.170.1 Detailed Description

Definition at line 95 of file NoOp\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/NoOp\_policy\_t.cc

## 6.171 artdaq::RequestBuffer Class Reference

Holds requests from [RequestReceiver](#) while they are being processed.

```
#include <artdaq/DAQrate/RequestBuffer.hh>
```

### Public Member Functions

- [RequestBuffer](#) (Fragment::sequence\_id\_t request\_increment=1)  
*RequestBuffer Constructor.*
- virtual [~RequestBuffer](#) ()  
*RequestBuffer Destructor.*
- void [push](#) (artdaq::Fragment::sequence\_id\_t seq, artdaq::Fragment::timestamp\_t ts)  
*Add a Request to the buffer.*
- void [reset](#) ()  
*Reset RequestBuffer, discarding all requests and tracking information.*
- std::map  
  < artdaq::Fragment::sequence\_id\_t,  
  artdaq::Fragment::timestamp\_t > [GetRequests](#) () const  
*Get the current requests*
- std::pair  
  < artdaq::Fragment::sequence\_id\_t,  
  artdaq::Fragment::timestamp\_t > [GetNextRequest](#) ()  
*Get the "next" request, i.e. the first unsatisfied request that has not already been returned by GetNextRequest*
- void [RemoveRequest](#) (artdaq::Fragment::sequence\_id\_t reqID)  
*Remove the request with the given sequence ID from the request map*
- void [ClearRequests](#) ()  
*Clear all requests from the map*
- std::map  
  < artdaq::Fragment::sequence\_id\_t,  
  artdaq::Fragment::timestamp\_t > [GetAndClearRequests](#) ()  
*Get the current requests, then clear the map*
- size\_t [size](#) ()  
*Get the number of requests currently stored in the RequestReceiver*
- bool [WaitForRequests](#) (int timeout\_ms)  
*Wait for a new request message, up to the timeout given*
- std::chrono::steady\_clock::time\_point [GetRequestTime](#) (artdaq::Fragment::sequence\_id\_t reqID)  
*Get the time a given request was received*

- bool `isRunning` () const  
*Determine whether the `RequestBuffer` is active.*
- void `setRunning` (bool running)  
*Set whether the `RequestBuffer` is active.*

### 6.171.1 Detailed Description

Holds requests from `RequestReceiver` while they are being processed.

Definition at line 15 of file `RequestBuffer.hh`.

### 6.171.2 Constructor & Destructor Documentation

6.171.2.1 `artdaq::RequestBuffer::RequestBuffer ( Fragment::sequence_id_t request_increment = 1 ) [explicit]`

`RequestBuffer` Constructor.

Parameters

<code>request_increment</code>	Expected increase in request sequence ID each request
--------------------------------	---

Definition at line 6 of file `RequestBuffer.cc`.

### 6.171.3 Member Function Documentation

6.171.3.1 `void artdaq::RequestBuffer::ClearRequests ( )`

Clear all requests from the map

Definition at line 134 of file `RequestBuffer.cc`.

6.171.3.2 `std::map< artdaq::Fragment::sequence_id_t, artdaq::Fragment::timestamp_t > artdaq::RequestBuffer::GetAndClearRequests ( )`

Get the current requests, then clear the map

Returns

Map relating sequence IDs to timestamps

Definition at line 145 of file `RequestBuffer.cc`.

6.171.3.3 `std::pair< artdaq::Fragment::sequence_id_t, artdaq::Fragment::timestamp_t > artdaq::RequestBuffer::GetNextRequest ( )`

Get the "next" request, i.e. the first unsatisfied request that has not already been returned by `GetNextRequest`

**Returns**

Request data for "next" request. Will return (0,0) if there is no "next" request

This function uses `last_next_request_` to ensure that it does not return the same request more than once

Definition at line 74 of file RequestBuffer.cc.

6.171.3.4 `std::map< artdaq::Fragment::sequence_id_t, artdaq::Fragment::timestamp_t > artdaq::RequestBuffer::GetRequests ( )`  
`const`

Get the current requests

**Returns**

Map relating sequence IDs to timestamps

Definition at line 63 of file RequestBuffer.cc.

6.171.3.5 `std::chrono::steady_clock::time_point artdaq::RequestBuffer::GetRequestTime ( artdaq::Fragment::sequence_id_t reqID )`

Get the time a given request was received

**Parameters**

<i>reqID</i>	Request ID of the request
--------------	---------------------------

**Returns**

`steady_clock::time_point` corresponding to when the request was received

Definition at line 192 of file RequestBuffer.cc.

6.171.3.6 `bool artdaq::RequestBuffer::isRunning ( ) const` `[inline]`

Determine whether the [RequestBuffer](#) is active.

**Returns**

Definition at line 94 of file RequestBuffer.hh.

6.171.3.7 `void artdaq::RequestBuffer::push ( artdaq::Fragment::sequence_id_t seq, artdaq::Fragment::timestamp_t ts )`

Add a Request to the buffer.

**Parameters**

<i>seq</i>	Sequence ID of the request
<i>ts</i>	Timestamp for the request

Definition at line 20 of file RequestBuffer.cc.

6.171.3.8 void artdaq::RequestBuffer::RemoveRequest ( artdaq::Fragment::sequence\_id\_t *reqID* )

Remove the request with the given sequence ID from the request map

## Parameters

<i>reqID</i>	Request ID to remove
--------------	----------------------

Definition at line 90 of file RequestBuffer.cc.

6.171.3.9 void artdaq::RequestBuffer::setRunning ( bool *running* ) [inline]

Set whether the [RequestBuffer](#) is active.

## Parameters

<i>running</i>	Whether the <a href="#">RequestBuffer</a> is active
----------------	---

Definition at line 99 of file RequestBuffer.hh.

6.171.3.10 size\_t artdaq::RequestBuffer::size ( )

Get the number of requests currently stored in the [RequestReceiver](#)

## Returns

The number of requests stored in the [RequestReceiver](#)

Definition at line 165 of file RequestBuffer.cc.

6.171.3.11 bool artdaq::RequestBuffer::WaitForRequests ( int *timeout\_ms* )

Wait for a new request message, up to the timeout given

## Parameters

<i>timeout_ms</i>	Milliseconds to wait for a new request to arrive
-------------------	--

## Returns

True if any requests are present in the request map

Definition at line 177 of file RequestBuffer.cc.

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

- artdaq/artdaq/DAQrate/RequestBuffer.hh
- artdaq/artdaq/DAQrate/RequestBuffer.cc

## 6.172 artdaq::detail::RequestHeader Struct Reference

Header of a [RequestMessage](#). Contains magic bytes for validation and a count of expected RequestPackets.

```
#include <artdaq/DAQrate/detail/RequestMessage.hh>
```

### Public Member Functions

- bool [isValid](#) () const  
*Check the magic bytes of the packet.*

## Public Attributes

- uint32\_t [header](#) {0x48454452}
- uint32\_t [packet\\_count](#) {0}  
*The number of RequestPackets in this Request message.*
- int [rank](#) {my\_rank}  
*Rank of the sender.*
- uint32\_t [run\\_number](#) {0}  
*The Run with which this request should be associated.*
- [RequestMessageMode](#) [mode](#) {RequestMessageMode::Normal}  
*Communicates additional information to the Request receiver.*

### 6.172.1 Detailed Description

Header of a [RequestMessage](#). Contains magic bytes for validation and a count of expected RequestPackets.

Definition at line 85 of file RequestMessage.hh.

### 6.172.2 Member Function Documentation

#### 6.172.2.1 bool artdaq::detail::RequestHeader::isValid ( ) const [inline]

Check the magic bytes of the packet.

#### Returns

Whether the correct magic bytes were found

Definition at line 100 of file RequestMessage.hh.

### 6.172.3 Member Data Documentation

#### 6.172.3.1 uint32\_t artdaq::detail::RequestHeader::header {0x48454452}

The magic bytes for the request header

Definition at line 88 of file RequestMessage.hh.

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

- artdaq/artdaq/DAQrate/detail/RequestMessage.hh

## 6.173 artdaq::detail::RequestMessage Class Reference

A [RequestMessage](#) consists of a [RequestHeader](#) and zero or more RequestPackets. They will usually be sent in two calls to send()

```
#include <artdaq/DAQrate/detail/RequestMessage.hh>
```

## Public Member Functions

- [RequestMessage](#) ()  
*Default Constructor.*
- `std::vector< uint8_t > GetMessage ()`  
*Get the contents of the [RequestMessage](#).*
- `void setMode (RequestMessageMode mode)`  
*Set the Request Message Mode for this request.*
- `void setRank (int rank)`  
*Set the rank in the header for this request. This will be the rank from which the request originates.*
- `void setRunNumber (int run)`  
*Set the run number in the header for this request. This will be the Run for which the request is valid.*
- `size_t size () const`  
*Get the number of RequestPackets in the [RequestMessage](#).*
- `void addRequest (const Fragment::sequence\_id\_t &seq, const Fragment::timestamp\_t &time)`  
*Add a request for a sequence ID and timestamp combination.*

## Static Public Member Functions

- `static size_t max\_request\_count ()`  
*Get the maximum number of requests that can be sent in a single [RequestMessage](#).*

### 6.173.1 Detailed Description

A [RequestMessage](#) consists of a [RequestHeader](#) and zero or more RequestPackets. They will usually be sent in two calls to `send()`

Definition at line 106 of file RequestMessage.hh.

### 6.173.2 Member Function Documentation

**6.173.2.1** `void artdaq::detail::RequestMessage::addRequest ( const Fragment::sequence\_id\_t &seq, const Fragment::timestamp\_t &time ) [inline]`

Add a request for a sequence ID and timestamp combination.

Parameters

<i>seq</i>	Sequence ID to request
<i>time</i>	Timestamp of request

Definition at line 172 of file RequestMessage.hh.

**6.173.2.2** `std::vector<uint8_t> artdaq::detail::RequestMessage::GetMessage ( ) [inline]`

Get the contents of the [RequestMessage](#).

Returns

Vector of bytes corresponding to the full [RequestMessage](#) (may span multiple packets)

Definition at line 121 of file RequestMessage.hh.

6.173.2.3 `static size_t artdaq::detail::RequestMessage::max_request_count ( ) [inline],[static]`

Get the maximum number of requests that can be sent in a single [RequestMessage](#).

#### Returns

The maximum number of requests that fit in a single UDP datagram

Definition at line 181 of file RequestMessage.hh.

6.173.2.4 `void artdaq::detail::RequestMessage::setMode ( RequestMessageMode mode ) [inline]`

Set the Request Message Mode for this request.

#### Parameters

<i>mode</i>	Mode for this Request Message
-------------	-------------------------------

Definition at line 137 of file RequestMessage.hh.

6.173.2.5 `void artdaq::detail::RequestMessage::setRank ( int rank ) [inline]`

Set the rank in the header for this request. This will be the rank from which the request originates.

#### Parameters

<i>rank</i>	Rank for this Request Message
-------------	-------------------------------

Definition at line 146 of file RequestMessage.hh.

6.173.2.6 `void artdaq::detail::RequestMessage::setRunNumber ( int run ) [inline]`

Set the run number in the header for this request. This will be the Run for which the request is valid.

#### Parameters

<i>run</i>	Run number for this Request Message
------------	-------------------------------------

Definition at line 156 of file RequestMessage.hh.

6.173.2.7 `size_t artdaq::detail::RequestMessage::size ( ) const [inline]`

Get the number of RequestPackets in the [RequestMessage](#).

#### Returns

The number of RequestPackets in the [RequestMessage](#)

Definition at line 165 of file RequestMessage.hh.

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

- artdaq/artdaq/DAQrate/detail/RequestMessage.hh



## 6.174 artdaq::detail::RequestPacket Struct Reference

The [RequestPacket](#) contains information about a single data request.

```
#include <artdaq/DAQrate/detail/RequestMessage.hh>
```

### Public Member Functions

- [RequestPacket](#) (const Fragment::sequence\_id\_t &seq, const Fragment::timestamp\_t &ts)  
*Create a [RequestPacket](#) using the given sequence ID and timestmap.*
- bool [isValid](#) () const  
*Check the magic bytes of the packet.*

### Public Attributes

- uint32\_t [header](#) {0}
- Fragment::sequence\_id\_t [sequence\\_id](#) {Fragment::InvalidSequenceID}  
*The sequence ID that responses to this request should use.*
- Fragment::timestamp\_t [timestamp](#) {Fragment::InvalidTimestamp}  
*The timestamp of the request.*

### 6.174.1 Detailed Description

The [RequestPacket](#) contains information about a single data request.

Definition at line 54 of file RequestMessage.hh.

### 6.174.2 Constructor & Destructor Documentation

6.174.2.1 `artdaq::detail::RequestPacket::RequestPacket ( const Fragment::sequence_id_t &seq, const Fragment::timestamp_t &ts ) [inline]`

Create a [RequestPacket](#) using the given sequence ID and timestmap.

Parameters

<code>seq</code>	Sequence ID of <a href="#">RequestPacket</a>
<code>ts</code>	Timestamp of RequestPAcket

Definition at line 69 of file RequestMessage.hh.

### 6.174.3 Member Function Documentation

6.174.3.1 `bool artdaq::detail::RequestPacket::isValid ( ) const [inline]`

Check the magic bytes of the packet.

Returns

Whether the correct magic bytes were found

Definition at line 79 of file RequestMessage.hh.

### 6.174.4 Member Data Documentation

#### 6.174.4.1 uint32\_t artdaq::detail::RequestPacket::header {0}

The magic bytes for the request packet

Definition at line 58 of file RequestMessage.hh.

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

- artdaq/artdaq/DAQrate/detail/RequestMessage.hh

## 6.175 artdaq::RequestReceiver Class Reference

Receive data requests and make them available to [CommandableFragmentGenerator](#) or other interested parties. Track received requests and report errors when inconsistency is detected.

```
#include <artdaq/DAQrate/detail/RequestReceiver.hh>
```

### Classes

- struct [Config](#)  
*Configuration of the [RequestReceiver](#). May be used for parameter validation*

### Public Types

- using [Parameters](#) = fhicl::WrappedTable< [Config](#) >  
*Used for ParameterSet validation (if desired)*

### Public Member Functions

- [RequestReceiver](#) ()  
*[RequestReceiver](#) Default Constructor.*
- [RequestReceiver](#) (const fhicl::ParameterSet &ps, std::shared\_ptr< [RequestBuffer](#) > output\_buffer)  
*[RequestReceiver](#) Constructor.*
- virtual [~RequestReceiver](#) ()  
*[RequestReceiver](#) Destructor.*
- void [setupRequestListener](#) ()  
*Opens the socket used to listen for data requests.*
- void [stopRequestReception](#) (bool force=false)  
*Disables (stops) the reception of data requests.*
- void [startRequestReception](#) ()  
*Enables (starts) the reception of data requests.*
- void [receiveRequestsLoop](#) ()  
*This function receives data request packets, adding new requests to the request list.*
- bool [isRunning](#) ()  
*Determine if the [RequestReceiver](#) is receiving requests*
- void [SetRunNumber](#) (uint32\_t run)  
*Sets the current run number*
- size\_t [GetReceivedMessageCount](#) ()

### 6.175.1 Detailed Description

Receive data requests and make them available to [CommandableFragmentGenerator](#) or other interested parties. Track received requests and report errors when inconsistency is detected.

Definition at line 25 of file RequestReceiver.hh.

### 6.175.2 Constructor & Destructor Documentation

6.175.2.1 `artdaq::RequestReceiver::RequestReceiver ( const fhicl::ParameterSet & ps, std::shared_ptr< RequestBuffer > output_buffer )`

[RequestReceiver](#) Constructor.

Parameters

<i>ps</i>	ParameterSet used to configure <a href="#">RequestReceiver</a> . See <a href="#">artdaq::RequestReceiver::Config</a>
<i>output_buffer</i>	Pointer to <a href="#">RequestBuffer</a> where Requests should be stored

Definition at line 42 of file RequestReceiver.cc.

### 6.175.3 Member Function Documentation

6.175.3.1 `bool artdaq::RequestReceiver::isRunning ( ) [inline]`

Determine if the [RequestReceiver](#) is receiving requests

Returns

True if the request receiver is running

Definition at line 89 of file RequestReceiver.hh.

6.175.3.2 `void artdaq::RequestReceiver::SetRunNumber ( uint32_t run ) [inline]`

Sets the current run number

Parameters

<i>run</i>	The current run number
------------	------------------------

Definition at line 95 of file RequestReceiver.hh.

6.175.3.3 `void artdaq::RequestReceiver::stopRequestReception ( bool force = false )`

Disables (stops) the reception of data requests.

Parameters

<i>force</i>	Whether to suppress any error messages (used if called from destructor)
--------------	---

Definition at line 121 of file RequestReceiver.cc.

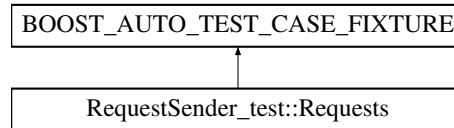
The documentation for this class was generated from the following files:

- `artdaq/artdaq/DAQrate/detail/RequestReceiver.hh`

- [artdaq/artdaq/DAQrate/detail/RequestReceiver.cc](#)

## 6.176 RequestSender\_test::Requests Struct Reference

Inheritance diagram for RequestSender\_test::Requests:



### Public Member Functions

- void **test\_method** ()

#### 6.176.1 Detailed Description

Definition at line 44 of file RequestSender\_t.cc.

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

- [artdaq/test/DAQrate/RequestSender\\_t.cc](#)

## 6.177 RequestSender\_test::Requests\_id Struct Reference

#### 6.177.1 Detailed Description

Definition at line 44 of file RequestSender\_t.cc.

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

- [artdaq/test/DAQrate/RequestSender\\_t.cc](#)

## 6.178 artdaq::RequestSender Class Reference

The [RequestSender](#) contains methods used to send data requests and Routing tokens.

```
#include <artdaq/DAQrate/detail/RequestSender.hh>
```

### Classes

- struct [Config](#)

*Configuration of the [RequestSender](#). May be used for parameter validation*

## Public Types

- using [Parameters](#) = fhicl::WrappedTable< [Config](#) >  
*Used for ParameterSet validation (if desired)*

## Public Member Functions

- [RequestSender](#) ()=delete  
*Default Constructor is deleted.*
- [RequestSender](#) ([RequestSender](#) const &)=delete  
*Copy Constructor is deleted.*
- [RequestSender](#) & operator= ([RequestSender](#) const &)=delete  
*Copy Assignment operator is deleted.*
- [RequestSender](#) ([RequestSender](#) &&)=delete  
*Move Constructor is deleted.*
- [RequestSender](#) & operator= ([RequestSender](#) &&)=delete  
*Move-assignment operator is deleted.*
- [RequestSender](#) (const fhicl::ParameterSet &pset)  
*[RequestSender](#) Constructor.*
- virtual ~[RequestSender](#) ()  
*[RequestSender](#) Destructor.*
- void [SetRequestMode](#) (detail::RequestMessageMode mode)  
*Set the mode for RequestMessages. Used to indicate when [RequestSender](#) should enter "EndOfRun" mode.*
- detail::RequestMessageMode [GetRequestMode](#) () const  
*Get the mode for RequestMessages.*
- void [SendRequest](#) (bool endOfRunOnly=false)  
*Send a request message containing all current requests.*
- void [AddRequest](#) (Fragment::sequence\_id\_t seqID, Fragment::timestamp\_t timestamp)  
*Add a request to the request list.*
- void [RemoveRequest](#) (Fragment::sequence\_id\_t seqID)  
*Remove a request from the request list.*
- void [SetRunNumber](#) (uint32\_t run)  
*Set the run number to be used in request messages.*
- bool [RequestsInFlight](#) ()  
*Determine if the [RequestSender](#) is currently sending any requests.*
- size\_t [GetSentMessageCount](#) ()  
*Get the number of requests sent by this [RequestSender](#).*

### 6.178.1 Detailed Description

The [RequestSender](#) contains methods used to send data requests and Routing tokens.

Definition at line 26 of file RequestSender.hh.

### 6.178.2 Constructor & Destructor Documentation

6.178.2.1 artdaq::RequestSender::RequestSender ( const fhicl::ParameterSet & pset ) [explicit]

[RequestSender](#) Constructor.

## Parameters

<i>pset</i>	ParameterSet used to configured <a href="#">RequestSender</a> . See <a href="#">artdaq::RequestSender::Config</a>
-------------	---

Definition at line 23 of file RequestSender.cc.

### 6.178.3 Member Function Documentation

6.178.3.1 void `artdaq::RequestSender::AddRequest ( Fragment::sequence_id_t seqID, Fragment::timestamp_t timestamp )`

Add a request to the request list.

## Parameters

<i>seqID</i>	Sequence ID for request
<i>timestamp</i>	Timestamp to request

Definition at line 224 of file RequestSender.cc.

6.178.3.2 `detail::RequestMessageMode` `artdaq::RequestSender::GetRequestMode ( ) const` `[inline]`

Get the mode for RequestMessages.

## Returns

Current RequestMessageMode of the [RequestSender](#)

Definition at line 91 of file RequestSender.hh.

6.178.3.3 `size_t` `artdaq::RequestSender::GetSentMessageCount ( )` `[inline]`

Get the number of requests sent by this [RequestSender](#).

## Returns

The number of requests sent

Definition at line 130 of file RequestSender.hh.

6.178.3.4 `RequestSender&` `artdaq::RequestSender::operator= ( RequestSender const & )` `[delete]`

Copy Assignment operator is deleted.

## Returns

[RequestSender](#) copy

6.178.3.5 void `artdaq::RequestSender::RemoveRequest ( Fragment::sequence_id_t seqID )`

Remove a request from the request list.

## Parameters

<i>seqID</i>	Sequence ID of request
--------------	------------------------

Definition at line 248 of file RequestSender.cc.

#### 6.178.3.6 bool artdaq::RequestSender::RequestsInFlight ( ) [inline]

Determine if the [RequestSender](#) is currently sending any requests.

## Returns

True if [RequestSender](#) has requests to send

This function is used for testing

Definition at line 124 of file RequestSender.hh.

#### 6.178.3.7 void artdaq::RequestSender::SendRequest ( bool *endOfRunOnly* = false )

Send a request message containing all current requests.

## Parameters

<i>endOfRunOnly</i>	Whether the request should only be sent in EndOfRun RequestMessageMode (default: false)
---------------------	---

Definition at line 200 of file RequestSender.cc.

#### 6.178.3.8 void artdaq::RequestSender::SetRequestMode ( detail::RequestMessageMode *mode* )

Set the mode for RequestMessages. Used to indicate when [RequestSender](#) should enter "EndOfRun" mode.

## Parameters

<i>mode</i>	Mode to set
-------------	-------------

Definition at line 74 of file RequestSender.cc.

#### 6.178.3.9 void artdaq::RequestSender::SetRunNumber ( uint32\_t *run* ) [inline]

Set the run number to be used in request messages.

## Parameters

<i>run</i>	Run number
------------	------------

Definition at line 116 of file RequestSender.hh.

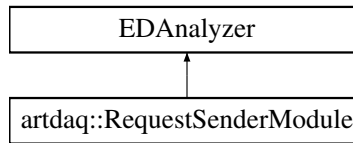
The documentation for this class was generated from the following files:

- artdaq/artdaq/DAQrate/detail/RequestSender.hh
- artdaq/artdaq/DAQrate/detail/RequestSender.cc

## 6.179 artdaq::RequestSenderModule Class Reference

An art::EDAnalyzer module which sends requests for events (e.g. for the Mu2e CRV system)

Inheritance diagram for `artdaq::RequestSenderModule`:



## Public Member Functions

- [RequestSenderModule](#) (`fhicl::ParameterSet const &pset`)  
*RequestSender Constructor.*
- [~RequestSenderModule](#) () override  
*Virtual Destructor. Shuts down MetricManager if one is present.*
- void [analyze](#) (`art::Event const &evt`) override  
*Analyze each event, using the configured mode bitmask.*
- void [beginRun](#) (`art::Run const &run`) override  
*Perform begin Run actions.*
- void [endRun](#) (`art::Run const &run`) override  
*PERform end Run actions.*

### 6.179.1 Detailed Description

An `art::EDAnalyzer` module which sends requests for events (e.g. for the Mu2e CRV system)

Definition at line 30 of file `RequestSender_module.cc`.

### 6.179.2 Constructor & Destructor Documentation

6.179.2.1 `artdaq::RequestSenderModule::RequestSenderModule ( fhicl::ParameterSet const & pset ) [explicit]`

[RequestSender](#) Constructor.

Parameters

<i>pset</i>	ParameterSet used to configure <a href="#">RequestSender</a>
-------------	--

See [artdaq/DAQrate/detail/RequestSender.hh](#) for [RequestSender](#) configuration Additional Parameters: "request\_list\_max\_size": Maximum number of requests to send at once

Definition at line 78 of file `RequestSender_module.cc`.

### 6.179.3 Member Function Documentation

6.179.3.1 `void artdaq::RequestSenderModule::analyze ( art::Event const & evt ) [override]`

Analyze each event, using the configured mode bitmask.



## Parameters

<i>evt</i>	art::Event to analyze
------------	-----------------------

Definition at line 89 of file RequestSender\_module.cc.

6.179.3.2 void artdaq::RequestSenderModule::beginRun ( art::Run const & *run* ) [override]

Perform begin Run actions.

## Parameters

<i>run</i>	Run object
------------	------------

Definition at line 118 of file RequestSender\_module.cc.

6.179.3.3 void artdaq::RequestSenderModule::endRun ( art::Run const & *run* ) [override]

Perform end Run actions.

## Parameters

<i>run</i>	Run object
------------	------------

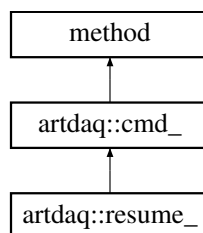
Definition at line 125 of file RequestSender\_module.cc.

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

- artdaq/artdaq/ArtModules/RequestSender\_module.cc

## 6.180 artdaq::resume\_ Class Reference

Inheritance diagram for artdaq::resume\_:



### Public Member Functions

- [resume\\_](#) (xmlrpc\_commander &c)

### Static Public Attributes

- static const uint64\_t [defaultTimeout](#) = 45
- static const uint64\_t [defaultTimestamp](#) = std::numeric\_limits<const uint64\_t>::max()

## Additional Inherited Members

### 6.180.1 Detailed Description

[resume\\_](#) Command class

Definition at line 922 of file xmlrpc\_commander.cc.

### 6.180.2 Constructor & Destructor Documentation

6.180.2.1 `artdaq::resume_::resume_( xmlrpc_commander & c ) [inline]`

[resume\\_](#) Constructor \

Parameters

<code>c</code>	<a href="#">xmlrpc_commander</a> to send transition commands to
----------------	---

Definition at line 922 of file xmlrpc\_commander.cc.

### 6.180.3 Member Data Documentation

6.180.3.1 `const uint64_t artdaq::resume_::defaultTimeout = 45 [static]`

Default timeout for command

Definition at line 922 of file xmlrpc\_commander.cc.

6.180.3.2 `const uint64_t artdaq::resume_::defaultTimestamp = std::numeric_limits<const uint64_t>::max() [static]`

Default timestamp for Command

Definition at line 922 of file xmlrpc\_commander.cc.

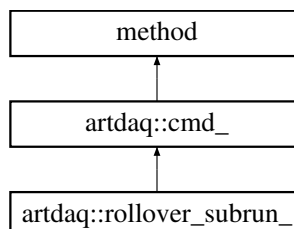
The documentation for this class was generated from the following file:

- artdaq/artdaq/ExternalComms/xmlrpc\_commander.cc

## 6.181 artdaq::rollover\_subrun\_ Class Reference

[rollover\\_subrun\\_](#) Command class

Inheritance diagram for `artdaq::rollover_subrun_`:



## Public Member Functions

- [rollover\\_subrun\\_](#) ([xmlrpc\\_commander](#) &c)  
*shutdown\_ Constructor*

## Static Public Attributes

- static const uint64\_t [defaultSequenceID](#) = 0xFFFFFFFFFFFFFFFF  
*Default Sequence ID for command.*
- static const uint32\_t [defaultSubrunNumber](#) = 1  
*Default subrun number for command.*

## Additional Inherited Members

### 6.181.1 Detailed Description

[rollover\\_subrun\\_](#) Command class

Definition at line 1213 of file `xmlrpc_commander.cc`.

### 6.181.2 Constructor & Destructor Documentation

6.181.2.1 `artdaq::rollover_subrun_::rollover_subrun_ ( xmlrpc\_commander & c )` `[inline]`

[shutdown\\_](#) Constructor

Parameters

<code>c</code>	<a href="#">xmlrpc_commander</a> to send transition commands to
----------------	---

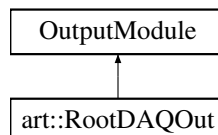
Definition at line 1220 of file `xmlrpc_commander.cc`.

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

- `artdaq/artdaq/ExternalComms/xmlrpc_commander.cc`

## 6.182 art::RootDAQOut Class Reference

Inheritance diagram for `art::RootDAQOut`:



## Classes

- struct [Config](#)

## Public Types

- using **Parameters** = fhicl::WrappedTable< [Config](#), [Config::KeysToIgnore](#) >

## Public Member Functions

- **RootDAQOut** (Parameters const &)
- **RootDAQOut** ([RootDAQOut](#) const &)=delete
- **RootDAQOut** ([RootDAQOut](#) &&)=delete
- [RootDAQOut](#) & **operator=** ([RootDAQOut](#) const &)=delete
- [RootDAQOut](#) & **operator=** ([RootDAQOut](#) &&)=delete
- void **postSelectProducts** () override
- void **beginJob** () override
- void **endJob** () override
- void **beginRun** (RunPrincipal const &) override
- void **endRun** (RunPrincipal const &) override
- void **beginSubRun** (SubRunPrincipal const &) override
- void **endSubRun** (SubRunPrincipal const &) override
- void **event** (EventPrincipal const &) override

## Static Public Attributes

- static constexpr char const \* **default\_tmpDir** {"<parent-path-of-filename>"}

### 6.182.1 Detailed Description

Definition at line 97 of file RootDAQOut\_module.cc.

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

- artdaq/artdaq/ArtModules/RootDAQOutput-s124/RootDAQOut\_module.cc

## 6.183 art::RootDAQOutFile Class Reference

### Public Types

- enum **ClosureRequestMode** { **MaxEvents** = 0, **MaxSize** = 1, **Unset** = 2 }
- using **RootOutputTreePtrArray** = std::array< std::unique\_ptr< [RootOutputTree](#) >, NumBranchTypes >

### Public Member Functions

- **RootDAQOutFile** (OutputModule \*, std::string const &fileName, ClosingCriteria const &fileSwitchCriteria, int compressionLevel, unsigned freePercent, unsigned freeMB, int64\_t saveMemoryObjectThreshold, int64\_t treeMaxVirtualSize, int splitLevel, int basketSize, DropMetaData dropMetaData, bool dropMetaDataForDroppedData)
- **RootDAQOutFile** ([RootDAQOutFile](#) const &)=delete
- **RootDAQOutFile** ([RootDAQOutFile](#) &&)=delete
- [RootDAQOutFile](#) & **operator=** ([RootDAQOutFile](#) const &)=delete
- [RootDAQOutFile](#) & **operator=** ([RootDAQOutFile](#) &&)=delete

- void **createDatabaseTables** ()
- void **writeTTrees** ()
- void **writeOne** (EventPrincipal const &)
- void **writeSubRun** (SubRunPrincipal const &)
- void **writeRun** (RunPrincipal const &)
- void **writeFileFormatVersion** ()
- void **writeFileIndex** ()
- void **writeProcessConfigurationRegistry** ()
- void **writeProcessHistoryRegistry** ()
- void **writeParameterSetRegistry** ()
- void **writeProductDescriptionRegistry** ()
- void **writeParentageRegistry** ()
- void **writeProductDependencies** ()
- void **writeFileCatalogMetadata** (FileStatsCollector const &stats, FileCatalogMetadata::collection\_type const &, FileCatalogMetadata::collection\_type const &)
- void **writeResults** (ResultsPrincipal &resp)
- void **setRunAuxiliaryRangeSetID** (RangeSet const &)
- void **setSubRunAuxiliaryRangeSetID** (RangeSet const &)
- void **beginInputFile** (RootFileBlock const \*, FastCloningEnabled fastCloningEnabled)
- void **incrementInputFileNumber** ()
- void **respondToCloseInputFile** (FileBlock const &)
- bool **requestsToCloseFile** ()
- void **setFileStatus** (OutputFileStatus ofs)
- void **selectProducts** ()
- std::string const & **currentFileName** () const
- bool **maxEventsPerFileReached** (FileIndex::EntryNumber\_t maxEventsPerFile) const
- bool **maxSizeReached** (unsigned maxFileSize) const

### 6.183.1 Detailed Description

Definition at line 46 of file RootDAQOutFile.h.

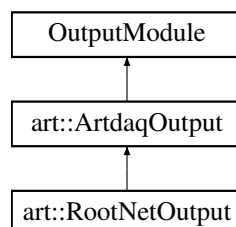
The documentation for this class was generated from the following files:

- artdaq/artdaq/ArtModules/RootDAQOutput-s124/RootDAQOutFile.h
- artdaq/artdaq/ArtModules/RootDAQOutput-s124/RootDAQOutFile.cc

## 6.184 art::RootNetOutput Class Reference

An art::OutputModule which sends events using DataSenderManager. This module is designed for transporting Fragment-wrapped art::Events after they have been read into art, for example between the EventBuilder and the Aggregator.

Inheritance diagram for art::RootNetOutput:



## Public Member Functions

- [RootNetOutput](#) (fhicl::ParameterSet const &ps)  
*RootNetOutput Constructor.*
- [~RootNetOutput](#) () override  
*RootNetOutput Destructor.*
- `size_t dataReceiverCount ()` const  
*Get the number of data receivers.*

## Protected Member Functions

- `void SendMessage (artdaq::FragmentPtr &fragment)` override  
*Send a message using DataSenderManager*

### 6.184.1 Detailed Description

An `art::OutputModule` which sends events using `DataSenderManager`. This module is designed for transporting Fragment-wrapped `art::Events` after they have been read into art, for example between the `EventBuilder` and the `Aggregator`.

Definition at line 28 of file `RootNetOutput_module.cc`.

### 6.184.2 Constructor & Destructor Documentation

6.184.2.1 `art::RootNetOutput::RootNetOutput ( fhicl::ParameterSet const & ps )` `[explicit]`

[RootNetOutput](#) Constructor.

Parameters

<code>ps</code>	ParameterSet used to configure <a href="#">RootNetOutput</a>
-----------------	--

[RootNetOutput](#) accepts no Parameters beyond those which `art::OutputModule` takes. See the `art::OutputModule` documentation for more details on those Parameters.

Definition at line 72 of file `RootNetOutput_module.cc`.

### 6.184.3 Member Function Documentation

6.184.3.1 `size_t art::RootNetOutput::dataReceiverCount ( )` const `[inline]`

Get the number of data receivers.

Returns

The number of data receivers

Definition at line 49 of file `RootNetOutput_module.cc`.

6.184.3.2 `void art::RootNetOutput::SendMessage ( artdaq::FragmentPtr & fragment )` `[override]`, `[protected]`, `[virtual]`

Send a message using `DataSenderManager`

## Parameters

<i>fragment</i>	Fragment to send
-----------------	------------------

Implements [art::ArtdaqOutput](#).

Definition at line 92 of file RootNetOutput\_module.cc.

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

- `artdaq/artdaq/ArtModules/RootNetOutput_module.cc`

## 6.185 art::RootOutputConfig Struct Reference

Configuration for ROOT output modules

```
#include <artdaq/ArtModules/detail/ArtConfig.hh>
```

### Classes

- struct [KeysToIgnore](#)  
*These keys should be ignored by the configuration validation processor*

### Public Types

- using [Name](#) = fhicl::Name  
*Parameter Name.*
- using [Comment](#) = fhicl::Comment
- template<typename T >  
using [Atom](#) = fhicl::Atom< T >  
*Configuration Parameter.*
- template<typename T >  
using [OptionalAtom](#) = fhicl::OptionalAtom< T >  
*Optional Configuration Parameter.*

### Public Attributes

- fhicl::TableFragment  
< art::OutputModule::Config > [omConfig](#)  
*Configuration common to all OutputModules.*
- [Atom](#)< std::string > [catalog](#) {[Name](#)("catalog"), ""}  
*???*
- [OptionalAtom](#)< bool > [dropAllEvents](#) {[Name](#)("dropAllEvents")}  
*Whether to drop all events ???*
- [Atom](#)< bool > [dropAllSubRuns](#) {[Name](#)("dropAllSubRuns"), false}  
*Whether to drop all subruns ???*
- [OptionalAtom](#)< bool > [fastCloning](#) {[Name](#)("fastCloning")}  
*Whether to try to use fastCloning on the file.*
- [Atom](#)< std::string > [tmpDir](#) {[Name](#)("tmpDir"), "/tmp"}

*Temporary directory.*

- `Atom< int > compressionLevel {Name("compressionLevel"), 7}`  
*Compression level to use. artdaq recommends  $\leq 3$ .*
- `Atom< int64_t > saveMemoryObjectThreshold {Name("saveMemoryObjectThreshold"), -1}`  
*???*
- `Atom< int64_t > treeMaxVirtualSize {Name("treeMaxVirtualSize"), -1}`  
*???*
- `Atom< int > splitLevel {Name("splitLevel"), 99}`  
*???*
- `Atom< int > basketSize {Name("basketSize"), 16384}`  
*???*
- `Atom< bool > dropMetaDataForDroppedData {Name("dropMetaDataForDroppedData"), false}`  
*???*
- `Atom< std::string > dropMetaData {Name("dropMetaData"), "NONE"}`  
*Which metadata to drop (Default: "NONE")*
- `Atom< bool > writeParameterSets {Name("writeParameterSets"), true}`  
*Write art ParameterSet to output file (Default: true)*
- `fhicl::Table`  
`< ClosingCriteria::Config > fileProperties {Name("fileProperties")}`  
*When should the file be closed.*

### 6.185.1 Detailed Description

Configuration for ROOT output modules

Definition at line 81 of file ArtConfig.hh.

### 6.185.2 Member Typedef Documentation

#### 6.185.2.1 using art::RootOutputConfig::Comment = fhicl::Comment

Parameter Comment

Definition at line 84 of file ArtConfig.hh.

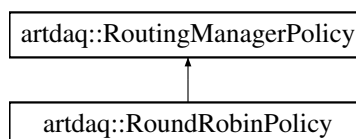
The documentation for this struct was generated from the following file:

- `artdaq/artdaq/ArtModules/detail/ArtConfig.hh`

## 6.186 artdaq::RoundRobinPolicy Class Reference

A [RoutingManagerPolicy](#) which evenly distributes Sequence IDs to all receivers. If an uneven number of tokens have been received, extra tokens are stored for the next table update.

Inheritance diagram for `artdaq::RoundRobinPolicy`:





## Public Member Functions

- [RoundRobinPolicy](#) (const fhicl::ParameterSet &ps)  
*RoundRobinPolicy Constructor.*
- [~RoundRobinPolicy](#) () override=default  
*Default virtual Destructor.*
- void [CreateRoutingTable](#) (detail::RoutingPacket &output) override  
*Add entries to the given RoutingPacket using currently-held tokens.*
- detail::RoutingPacketEntry [CreateRouteForSequenceID](#) (artdaq::Fragment::sequence\_id\_t seq, int requesting\_rank) override  
*Get an [artdaq::detail::RoutingPacketEntry](#) for a given sequence ID and rank. Used by RequestBasedEventBuilder and DataFlow RoutingManagerMode.*

## Additional Inherited Members

### 6.186.1 Detailed Description

A [RoutingManagerPolicy](#) which evenly distributes Sequence IDs to all receivers. If an uneven number of tokens have been received, extra tokens are stored for the next table update.

Definition at line 15 of file RoundRobin\_policy.cc.

### 6.186.2 Constructor & Destructor Documentation

6.186.2.1 [artdaq::RoundRobinPolicy::RoundRobinPolicy](#) ( const fhicl::ParameterSet & ps ) `[inline]`, `[explicit]`

[RoundRobinPolicy](#) Constructor.

Parameters

<i>ps</i>	ParameterSet used to configure <a href="#">RoundRobinPolicy</a>
-----------	---

[RoundRobinPolicy](#) accepts the following Parameter: "minimum\_participants" (Default: 0): Minimum number of receivers to distribute between. Use negative number to indicate how many can be missing from total. If the number of allowed missing receivers is greater than the number that exist, then the minimum number of participants will be set to 1.

Definition at line 25 of file RoundRobin\_policy.cc.

### 6.186.3 Member Function Documentation

6.186.3.1 detail::RoutingPacketEntry [artdaq::RoundRobinPolicy::CreateRouteForSequenceID](#) ( artdaq::Fragment::sequence\_id\_t seq, int requesting\_rank ) `[override]`, `[virtual]`

Get an [artdaq::detail::RoutingPacketEntry](#) for a given sequence ID and rank. Used by RequestBasedEventBuilder and DataFlow RoutingManagerMode.

Parameters

<i>seq</i>	Sequence Number to get route for
<i>requesting_rank</i>	Rank to route for

**Returns**

[artdaq::detail::RoutingPacketEntry](#) connecting sequence ID to destination rank

Implements [artdaq::RoutingManagerPolicy](#).

Definition at line 102 of file RoundRobin\_policy.cc.

**6.186.3.2** `void artdaq::RoundRobinPolicy::CreateRoutingTable ( detail::RoutingPacket & output ) [override], [virtual]`

Add entries to the given RoutingPacket using currently-held tokens.

**Parameters**

<i>output</i>	RoutingPacket to add entries to
---------------	---------------------------------

[RoundRobinPolicy](#) will go through the list of receivers as many times as it can, until one or more receivers have no tokens. It always does full "turns" through the receiver list.

Implements [artdaq::RoutingManagerPolicy](#).

Definition at line 67 of file RoundRobin\_policy.cc.

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

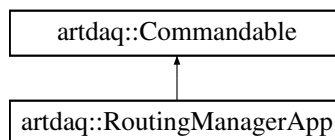
- `artdaq/artdaq/RoutingPolicies/RoundRobin_policy.cc`

## 6.187 artdaq::RoutingManagerApp Class Reference

[RoutingManagerApp](#) is an [artdaq::Commandable](#) derived class which controls the [RoutingManagerCore](#) state machine.

```
#include <artdaq/Application/RoutingManagerApp.hh>
```

Inheritance diagram for `artdaq::RoutingManagerApp`:

**Public Member Functions**

- [RoutingManagerApp \(\)](#)  
*[RoutingManagerApp](#) Constructor.*
- [RoutingManagerApp \(RoutingManagerApp const &\)=delete](#)  
*Copy Constructor is deleted.*
- `virtual ~RoutingManagerApp ()=default`  
*Default Destructor.*
- [RoutingManagerApp & operator= \(RoutingManagerApp const &\)=delete](#)  
*Copy Assignment Operator is deleted.*
- [RoutingManagerApp \(RoutingManagerApp &&\)=delete](#)  
*Move Constructor is deleted.*

- [RoutingManagerApp](#) & operator= ([RoutingManagerApp](#) &&)=delete  
*Move Assignment Operator is deleted.*
- bool [do\\_initialize](#) (fhicl::ParameterSet const &pset, uint64\_t timeout, uint64\_t timestamp) override  
*Initialize the [RoutingManagerCore](#).*
- bool [do\\_start](#) (art::RunID id, uint64\_t timeout, uint64\_t timestamp) override  
*Start the [RoutingManagerCore](#).*
- bool [do\\_stop](#) (uint64\_t timeout, uint64\_t timestamp) override  
*Stop the [RoutingManagerCore](#).*
- bool [do\\_pause](#) (uint64\_t timeout, uint64\_t timestamp) override  
*Pause the [RoutingManagerCore](#).*
- bool [do\\_resume](#) (uint64\_t timeout, uint64\_t timestamp) override  
*Resume the [RoutingManagerCore](#).*
- bool [do\\_shutdown](#) (uint64\_t timeout) override  
*Shutdown the [RoutingManagerCore](#).*
- bool [do\\_soft\\_initialize](#) (fhicl::ParameterSet const &pset, uint64\_t timeout, uint64\_t timestamp) override  
*Soft-Initialize the [RoutingManagerCore](#).*
- bool [do\\_reinitialize](#) (fhicl::ParameterSet const &pset, uint64\_t timeout, uint64\_t timestamp) override  
*Reinitialize the [RoutingManagerCore](#).*
- void [BootedEnter](#) () override  
*Action taken upon entering the "Booted" state.*
- std::string [report](#) (std::string const &) const override  
*If which is "transition\_status", report the status of the last transition. Otherwise pass through to AggregatorCore.*

## Additional Inherited Members

### 6.187.1 Detailed Description

[RoutingManagerApp](#) is an [artdaq::Commandable](#) derived class which controls the [RoutingManagerCore](#) state machine.  
Definition at line 18 of file [RoutingManagerApp.hh](#).

### 6.187.2 Constructor & Destructor Documentation

#### 6.187.2.1 artdaq::RoutingManagerApp::RoutingManagerApp ( ) [default]

[RoutingManagerApp](#) Constructor.

Default constructor.

### 6.187.3 Member Function Documentation

#### 6.187.3.1 void artdaq::RoutingManagerApp::BootedEnter ( ) [override],[virtual]

Action taken upon entering the "Booted" state.

This resets the [RoutingManagerCore](#) pointer

Reimplemented from [artdaq::Commandable](#).

Definition at line 198 of file [RoutingManagerApp.cc](#).

6.187.3.2 `bool artdaq::RoutingManagerApp::do_initialize ( fhicl::ParameterSet const & pset, uint64_t timeout, uint64_t timestamp )`  
[`override`], [`virtual`]

Initialize the [RoutingManagerCore](#).

## Parameters

<i>pset</i>	ParameterSet used to configure the <a href="#">RoutingManagerCore</a>
<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

## Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 23 of file RoutingManagerApp.cc.

6.187.3.3 `bool artdaq::RoutingManagerApp::do_pause ( uint64_t timeout, uint64_t timestamp )` `[override],[virtual]`

Pause the [RoutingManagerCore](#).

## Parameters

<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

## Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 106 of file RoutingManagerApp.cc.

6.187.3.4 `bool artdaq::RoutingManagerApp::do_reinitialize ( fhicl::ParameterSet const & pset, uint64_t timeout, uint64_t timestamp )` `[override],[virtual]`

Reinitialize the [RoutingManagerCore](#).

## Parameters

<i>pset</i>	ParameterSet used to configure the <a href="#">RoutingManagerCore</a>
<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

## Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 186 of file RoutingManagerApp.cc.

6.187.3.5 `bool artdaq::RoutingManagerApp::do_resume ( uint64_t timeout, uint64_t timestamp )` `[override],[virtual]`

Resume the [RoutingManagerCore](#).

## Parameters

<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

## Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 127 of file RoutingManagerApp.cc.

6.187.3.6 `bool artdaq::RoutingManagerApp::do_shutdown ( uint64_t timeout )` `[override],[virtual]`

Shutdown the [RoutingManagerCore](#).

## Parameters

<i>timeout</i>	<a href="#">Timeout</a> for transition
----------------	--

## Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 161 of file RoutingManagerApp.cc.

6.187.3.7 `bool artdaq::RoutingManagerApp::do_soft_initialize ( fhicl::ParameterSet const & pset, uint64_t timeout, uint64_t timestamp )` `[override],[virtual]`

Soft-Initialize the [RoutingManagerCore](#).

## Parameters

<i>pset</i>	ParameterSet used to configure the <a href="#">RoutingManagerCore</a>
<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

## Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 173 of file RoutingManagerApp.cc.

6.187.3.8 `bool artdaq::RoutingManagerApp::do_start ( art::RunID id, uint64_t timeout, uint64_t timestamp )` `[override],[virtual]`

Start the [RoutingManagerCore](#).

## Parameters

<i>id</i>	Run ID of new run
<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

## Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 45 of file RoutingManagerApp.cc.

6.187.3.9 `bool artdaq::RoutingManagerApp::do_stop ( uint64_t timeout, uint64_t timestamp )` `[override],[virtual]`

Stop the [RoutingManagerCore](#).

## Parameters

<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

## Returns

Whether the transition succeeded

Reimplemented from [artdaq::Commandable](#).

Definition at line 83 of file RoutingManagerApp.cc.

6.187.3.10 `RoutingManagerApp& artdaq::RoutingManagerApp::operator= ( RoutingManagerApp const & )` `[delete]`

Copy Assignment Operator is deleted.

## Returns

[RoutingManagerApp](#) copy

6.187.3.11 `std::string artdaq::RoutingManagerApp::report ( std::string const & which ) const` `[override],[virtual]`

If which is "transition\_status", report the status of the last transition. Otherwise pass through to AggregatorCore.

## Parameters

<i>which</i>	What to report on
--------------	-------------------

## Returns

Report string. Empty for unknown "which" parameter

Reimplemented from [artdaq::Commandable](#).

Definition at line 209 of file RoutingManagerApp.cc.

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

- [artdaq/artdaq/Application/RouterManagerApp.hh](#)
- [artdaq/artdaq/Application/RouterManagerApp.cc](#)

## 6.188 artdaq::RouterManagerCore Class Reference

[RouterManagerCore](#) implements the state machine for the RouterManager artdaq application. [RouterManagerCore](#) collects tokens from receivers, and at regular intervals uses these tokens to build Routing Tables that are sent to the senders.

```
#include <artdaq/Application/RouterManagerCore.hh>
```

### Public Member Functions

- [RouterManagerCore](#) ()  
*RouterManagerCore Constructor.*
- [RouterManagerCore](#) ([RouterManagerCore](#) const &)=delete  
*Copy Constructor is deleted.*
- [~RouterManagerCore](#) ()
- [RouterManagerCore](#) & operator= ([RouterManagerCore](#) const &)=delete  
*Copy Assignment operator is deleted.*
- [RouterManagerCore](#) ([RouterManagerCore](#) &&)=delete  
*Move Constructor is deleted.*
- [RouterManagerCore](#) & operator= ([RouterManagerCore](#) &&)=delete  
*Move Assignment Operator is deleted.*
- bool [initialize](#) (fhicl::ParameterSet const &pset, uint64\_t, uint64\_t)  
*Processes the initialize request.*
- bool [start](#) (art::RunID id, uint64\_t, uint64\_t)  
*Start the RouterManagerCore.*
- bool [stop](#) (uint64\_t, uint64\_t)  
*Stops the RouterManagerCore.*
- bool [pause](#) (uint64\_t, uint64\_t)  
*Pauses the RouterManagerCore.*
- bool [resume](#) (uint64\_t, uint64\_t)  
*Resumes the RouterManagerCore.*
- bool [shutdown](#) (uint64\_t)  
*Shuts Down the RouterManagerCore.*
- bool [soft\\_initialize](#) (fhicl::ParameterSet const &pset, uint64\_t timeout, uint64\_t timestamp)  
*Soft-Initializes the RouterManagerCore.*
- bool [reinitialize](#) (fhicl::ParameterSet const &pset, uint64\_t timeout, uint64\_t timestamp)  
*Reinitializes the RouterManagerCore.*
- void [process\\_event\\_table](#) ()  
*Main loop of the RouterManagerCore. Determines when to send the next table update, asks the RouterManagerPolicy for the table to send, and sends it.*
- void [send\\_event\\_table](#) (detail::RoutingPacket packet)  
*Sends a detail::RoutingPacket to the table receivers.*
- std::string [report](#) (std::string const &) const  
*Send a report on the current status of the RouterManagerCore.*
- size\_t [get\\_update\\_count](#) () const  
*Get the number of table updates sent by this RouterManager this run.*



## Static Public Attributes

- static const std::string [TABLE\\_UPDATES\\_STAT\\_KEY](#)  
*Key for Table Update count MonnitoredQuantity.*
- static const std::string [TOKENS\\_RECEIVED\\_STAT\\_KEY](#)  
*Key for the Tokens Received MonitoredQuantity.*
- static const std::string [CURRENT\\_TABLE\\_INTERVAL\\_STAT\\_KEY](#)  
*Key for the Current Table Interval MonitoredQuantity.*

### 6.188.1 Detailed Description

[RoutingManagerCore](#) implements the state machine for the RoutingManager artdaq application. [RoutingManagerCore](#) collects tokens from receivers, and at regular intervals uses these tokens to build Routing Tables that are sent to the senders.

Definition at line 34 of file RoutingManagerCore.hh.

### 6.188.2 Constructor & Destructor Documentation

#### 6.188.2.1 artdaq::RoutingManagerCore::~~RoutingManagerCore ( )

Destructor.

Definition at line 41 of file RoutingManagerCore.cc.

### 6.188.3 Member Function Documentation

#### 6.188.3.1 size\_t artdaq::RoutingManagerCore::get\_update\_count ( ) const [inline]

Get the number of table updates sent by this RoutingManager this run.

Returns

The number of table updates sent by this RoutingManager this run

Definition at line 168 of file RoutingManagerCore.hh.

#### 6.188.3.2 bool artdaq::RoutingManagerCore::initialize ( fhicl::ParameterSet const & pset, uint64\_t , uint64\_t )

Processes the initialize request.

Parameters

<i>pset</i>	ParameterSet used to configure the <a href="#">RoutingManagerCore</a>
-------------	---

## Returns

Whether the initialize attempt succeeded

```
* RoutingManagerCore accepts the following Parameters:
* "daq" (REQUIRED): FHiCL table containing DAQ configuration
* "policy" (REQUIRED): FHiCL table containing the RoutingManagerPolicy configuration
* "policy" (Default: ""): Name of the RoutingManagerPolicy plugin to load
* "rt_priority" (Default: 0): Unix process priority to assign to RoutingManagerCore
* "table_update_interval_ms" (Default: 1000): Maximum amount of time between table updates
* "table_update_interval_high_frac" (Default: 0.75): Fraction of the maximum seen table size at which the interval
* "table_update_interval_low_frac" (Default: 0.5): Fraction of the maximum seen table size at which the interval
* "senders_send_by_send_count" (Default: false): If true, senders will use the current send count to lookup routing
* "table_ack_retry_count" (Default: 5): The number of times the table will be resent while waiting for acknowledgment
* "table_update_port" (Default: 35556): The port on which to send table updates
* "table_acknowledge_port" (Default: 35557): The port on which to listen for RoutingAckPacket datagrams
* "table_update_address" (Default: "227.128.12.28"): Multicast address to send table updates to
* "routing_manager_hostname" (Default: "localhost"): Hostname to send table updates from
* "metrics": FHiCL table containing configuration for MetricManager
*
```

Definition at line 48 of file RoutingManagerCore.cc.

### 6.188.3.3 RoutingManagerCore& artdaq::RoutingManagerCore::operator= ( RoutingManagerCore const & ) [delete]

Copy Assignment operator is deleted.

## Returns

[RoutingManagerCore](#) copy

### 6.188.3.4 bool artdaq::RoutingManagerCore::pause ( uint64\_t, uint64\_t )

Pauses the [RoutingManagerCore](#).

## Returns

True if no exception

Definition at line 220 of file RoutingManagerCore.cc.

### 6.188.3.5 bool artdaq::RoutingManagerCore::reinitialize ( fhi::ParameterSet const & pset, uint64\_t timeout, uint64\_t timestamp )

Reinitializes the [RoutingManagerCore](#).

## Parameters

<i>pset</i>	ParameterSet for configuring <a href="#">RoutingManagerCore</a>
<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

## Returns

Returns initialize status

Definition at line 258 of file RoutingManagerCore.cc.

6.188.3.6 `std::string artdaq::RoutingManagerCore::report ( std::string const & ) const`

Sends a report on the current status of the [RoutingManagerCore](#).

#### Returns

A string containing the report on the current status of the [RoutingManagerCore](#)

Definition at line 392 of file `RoutingManagerCore.cc`.

6.188.3.7 `bool artdaq::RoutingManagerCore::resume ( uint64_t, uint64_t )`

Resumes the [RoutingManagerCore](#).

#### Returns

True if no exception

Definition at line 229 of file `RoutingManagerCore.cc`.

6.188.3.8 `void artdaq::RoutingManagerCore::send_event_table ( detail::RoutingPacket packet )`

Sends a [detail::RoutingPacket](#) to the table receivers.

#### Parameters

<i>packet</i>	The <a href="#">detail::RoutingPacket</a> to send
---------------	---

`send_event_table` checks the table update socket and the acknowledge socket before sending the table update the first time. It then enters a loop where it sends the table update, then waits for acknowledgement packets. It keeps track of which senders have sent their acknowledgement packets, and discards duplicate acks. It leaves this loop once all senders have sent a valid acknowledgement packet.

Definition at line 365 of file `RoutingManagerCore.cc`.

6.188.3.9 `bool artdaq::RoutingManagerCore::shutdown ( uint64_t )`

Shuts Down the [RoutingManagerCore](#).

#### Returns

If the shutdown was successful

Definition at line 237 of file `RoutingManagerCore.cc`.

6.188.3.10 `bool artdaq::RoutingManagerCore::soft_initialize ( fhicl::ParameterSet const & pset, uint64_t timeout, uint64_t timestamp )`

Soft-Initializes the [RoutingManagerCore](#).

## Parameters

<i>pset</i>	ParameterSet for configuring <a href="#">RoutingManagerCore</a>
<i>timeout</i>	<a href="#">Timeout</a> for transition
<i>timestamp</i>	Timestamp of transition

## Returns

Returns initialize status

Definition at line 250 of file RoutingManagerCore.cc.

6.188.3.11 `bool artdaq::RoutingManagerCore::start ( art::RunID id, uint64_t, uint64_t )`

Start the [RoutingManagerCore](#).

## Parameters

<i>id</i>	Run ID of the current run
-----------	---------------------------

## Returns

True if no exception

Definition at line 192 of file RoutingManagerCore.cc.

6.188.3.12 `bool artdaq::RoutingManagerCore::stop ( uint64_t, uint64_t )`

Stops the [RoutingManagerCore](#).

## Returns

True if no exception

Definition at line 209 of file RoutingManagerCore.cc.

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

- `artdaq/artdaq/Application/RoutingManagerCore.hh`
- `artdaq/artdaq/Application/RoutingManagerCore.cc`

## 6.189 artdaq::detail::RoutingManagerModeConverter Class Reference

Convert RoutingManagerMode to/from strings.

```
#include <artdaq/DAQrate/detail/RoutingPacket.hh>
```

### Static Public Member Functions

- static [RoutingManagerMode](#) `stringToRoutingManagerMode` (std::string const &modeString)  
*Convert String to RoutingManagerMode.*
- static std::string `routingManagerModeToString` ([RoutingManagerMode](#) mode)  
*Convert RoutingManagerMode to string.*

### 6.189.1 Detailed Description

Convert RoutingManagerMode to/from strings.

Definition at line 37 of file RoutingPacket.hh.

### 6.189.2 Member Function Documentation

**6.189.2.1** `static std::string artdaq::detail::RoutingManagerModeConverter::routingManagerModeToString ( RoutingManagerMode mode ) [inline],[static]`

Convert RoutingManagerMode to string.

Parameters

<i>mode</i>	Mode to convert
-------------	-----------------

Returns

String representation of mode

Definition at line 57 of file RoutingPacket.hh.

**6.189.2.2** `static RoutingManagerMode artdaq::detail::RoutingManagerModeConverter::stringToRoutingManagerMode ( std::string const & modeString ) [inline],[static]`

Convert String to RoutingManagerMode.

Parameters

<i>modeString</i>	String to convert
-------------------	-------------------

Returns

Resultant RoutingManagerMode, RoutingManagerMode::INVALID if no match

Definition at line 45 of file RoutingPacket.hh.

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

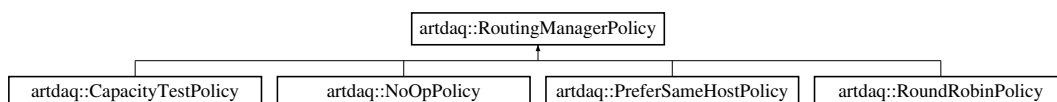
- artdaq/artdaq/DAQrate/detail/RoutingPacket.hh

## 6.190 artdaq::RoutingManagerPolicy Class Reference

The interface through which [RoutingManagerCore](#) obtains Routing Tables using received Routing Tokens.

```
#include <artdaq/RoutingPolicies/RoutingManagerPolicy.hh>
```

Inheritance diagram for artdaq::RoutingManagerPolicy:



## Public Member Functions

- [RoutingManagerPolicy](#) (const fhicl::ParameterSet &ps)  
*[RoutingManagerPolicy](#) Constructor.*
- virtual [~RoutingManagerPolicy](#) ()=default  
*Default virtual Destructor.*
- size\_t [GetReceiverCount](#) () const  
*Get the number of configured receivers.*
- size\_t [GetMaxNumberOfTokens](#) () const  
*Get the largest number of tokens that the [RoutingManagerPolicy](#) has seen at any one time.*
- size\_t [GetTokensUsedSinceLastUpdate](#) () const  
*Get the number of tokens that have been used since the last update.*
- void [ResetTokensUsedSinceLastUpdate](#) ()  
*Reset the number of tokens used.*
- void [AddReceiverToken](#) (int rank, unsigned new\_slots\_free)  
*Add a token to the token list.*
- void [Reset](#) ()  
*Reset the policy, setting the next sequence ID to be used to 1, and removing any tokens.*
- Fragment::sequence\_id\_t [GetNextSequenceID](#) () const  
*Get the next sequence ID to be routed.*
- size\_t [GetHeldTokenCount](#) () const  
*Get the number of tokens that are waiting to be used.*
- detail::RoutingPacket [GetCurrentTable](#) ()  
*Create a [RoutingPacket](#) from currently-owned tokens. Used by [EventBuilder](#) and [RequestBasedEventBuilder](#) [RoutingManagerMode](#).*
- detail::RoutingPacketEntry [GetRouteForSequenceID](#) (artdaq::Fragment::sequence\_id\_t seq, int requesting\_rank)  
*Get an [artdaq::detail::RoutingPacketEntry](#) for a given sequence ID and rank. Used by [RequestBasedEventBuilder](#) and [DataFlow](#) [RoutingManagerMode](#).*
- detail::RoutingManagerMode [GetRoutingMode](#) () const  
*Get the current [RoutingManagerMode](#) of this [RoutingManager](#).*
- size\_t [GetCacheSize](#) () const  
*Get the size of the routing cache. For testing.*
- bool [CacheHasRoute](#) (artdaq::Fragment::sequence\_id\_t seq) const  
*Determine whether the routing cache has a route for the given sequence ID. For testing.*

## Protected Member Functions

- virtual void [CreateRoutingTable](#) (detail::RoutingPacket &tables)=0  
*Generate entries to add to the given table.*
- virtual [detail::RoutingPacketEntry](#) [CreateRouteForSequenceID](#) (artdaq::Fragment::sequence\_id\_t seq, int requesting\_rank)=0  
*Generate a route for the given sequence ID and source rank.*

## Protected Attributes

- `std::deque< int > tokens_`  
*The list of tokens which are available for use.*
- `std::atomic< size_t > tokens_used_since_last_update_`  
*Number of tokens consumed since last metric update.*
- `Fragment::sequence_id_t next_sequence_id_`  
*The next sequence ID to be assigned.*
- `std::unordered_set< int > receiver_ranks_`  
*Configured receiver (e.g. EventBuilder for BR->EB routing) ranks.*
- `detail::RoutingManagerMode routing_mode_`  
*Current routing mode.*

### 6.190.1 Detailed Description

The interface through which [RoutingManagerCore](#) obtains Routing Tables using received Routing Tokens.  
Definition at line 21 of file `RoutingManagerPolicy.hh`.

### 6.190.2 Constructor & Destructor Documentation

6.190.2.1 `artdaq::RoutingManagerPolicy::RoutingManagerPolicy ( const fhicl::ParameterSet & ps ) [explicit]`

[RoutingManagerPolicy](#) Constructor.

Parameters

<i>ps</i>	ParameterSet used to configure the <a href="#">RoutingManagerPolicy</a>
-----------	---

\* `RoutingManagerPolicy` accepts the following Parameters:  
\*

Definition at line 9 of file `RoutingManagerPolicy.cc`.

### 6.190.3 Member Function Documentation

6.190.3.1 `void artdaq::RoutingManagerPolicy::AddReceiverToken ( int rank, unsigned new_slots_free )`

Add a token to the token list.

Parameters

<i>rank</i>	Rank that the token is from
<i>new_slots_free</i>	Number of slots that are now free (should usually be 1)

Definition at line 37 of file `RoutingManagerPolicy.cc`.

6.190.3.2 `bool artdaq::RoutingManagerPolicy::CacheHasRoute ( artdaq::Fragment::sequence_id_t seq ) const [inline]`

Determine whether the routing cache has a route for the given sequence ID. For testing.

## Parameters

<i>seq</i>	Sequence ID to check
------------	----------------------

## Returns

True if the sequence ID is in the cache

Definition at line 120 of file RoutingManagerPolicy.hh.

**6.190.3.3** `virtual detail::RoutingPacketEntry artdaq::RoutingManagerPolicy::CreateRouteForSequenceID ( artdaq::Fragment::sequence_id_t seq, int requesting_rank ) [protected],[pure virtual]`

Generate a route for the given sequence ID and source rank.

## Parameters

<i>seq</i>	Sequence ID to route
<i>requesting_rank</i>	Source rank requesting routing information

## Returns

An [artdaq::detail::RoutingPacketEntry](#) linking the sequence ID to a destination rank

Implemented in [artdaq::PreferSameHostPolicy](#), [artdaq::RoundRobinPolicy](#), [artdaq::NoOpPolicy](#), and [artdaq::Capacity-TestPolicy](#).

**6.190.3.4** `virtual void artdaq::RoutingManagerPolicy::CreateRoutingTable ( detail::RoutingPacket & tables ) [protected],[pure virtual]`

Generate entries to add to the given table.

## Parameters

<i>tables</i>	The RoutingPacket to add entries to
---------------	-------------------------------------

Implemented in [artdaq::PreferSameHostPolicy](#), [artdaq::RoundRobinPolicy](#), [artdaq::NoOpPolicy](#), and [artdaq::Capacity-TestPolicy](#).

**6.190.3.5** `size_t artdaq::RoutingManagerPolicy::GetCacheSize ( ) const [inline]`

Get the size of the routing cache. For testing.

## Returns

Size of the routing cache

Definition at line 114 of file RoutingManagerPolicy.hh.

**6.190.3.6** `artdaq::detail::RoutingPacket artdaq::RoutingManagerPolicy::GetCurrentTable ( )`

Create a RoutingPacket from currently-owned tokens. Used by EventBuilder and RequestBasedEventBuilder RoutingManagerMode.



**Returns**

[artdaq::detail::RoutingPacket](#) created from tokens held by Routing Manager.

Definition at line 18 of file RoutingManagerPolicy.cc.

6.190.3.7 `size_t artdaq::RoutingManagerPolicy::GetHeldTokenCount ( ) const [inline]`

Get the number of tokens that are waiting to be used.

**Returns**

size of the held tokens list

Definition at line 83 of file RoutingManagerPolicy.hh.

6.190.3.8 `size_t artdaq::RoutingManagerPolicy::GetMaxNumberOfTokens ( ) const [inline]`

Get the largest number of tokens that the [RoutingManagerPolicy](#) has seen at any one time.

**Returns**

The largest number of tokens that the [RoutingManagerPolicy](#) has seen at any one time

Definition at line 49 of file RoutingManagerPolicy.hh.

6.190.3.9 `Fragment::sequence_id_t artdaq::RoutingManagerPolicy::GetNextSequenceID ( ) const [inline]`

Get the next sequence ID to be routed.

**Returns**

Next sequence ID to be routed

Definition at line 77 of file RoutingManagerPolicy.hh.

6.190.3.10 `size_t artdaq::RoutingManagerPolicy::GetReceiverCount ( ) const [inline]`

Get the number of configured receivers.

**Returns**

The size of the receiver\_ranks list

Definition at line 43 of file RoutingManagerPolicy.hh.

6.190.3.11 `artdaq::detail::RoutingPacketEntry artdaq::RoutingManagerPolicy::GetRouteForSequenceID ( artdaq::Fragment::sequence_id_t seq, int requesting_rank )`

Get an [artdaq::detail::RoutingPacketEntry](#) for a given sequence ID and rank. Used by RequestBasedEventBuilder and DataFlow RoutingManagerMode.

## Parameters

<i>seq</i>	Sequence Number to get route for
<i>requesting_rank</i>	Rank to route for

## Returns

[artdaq::detail::RoutingPacketEntry](#) connecting sequence ID to destination rank

Definition at line 79 of file RoutingManagerPolicy.cc.

**6.190.3.12** `detail::RoutingManagerMode` `artdaq::RoutingManagerPolicy::GetRoutingMode ( ) const` `[inline]`

Get the current RoutingManagerMode of this RoutingManager.

## Returns

[artdaq::detail::RoutingManagerMode](#) value

Definition at line 107 of file RoutingManagerPolicy.hh.

**6.190.3.13** `size_t` `artdaq::RoutingManagerPolicy::GetTokensUsedSinceLastUpdate ( ) const` `[inline]`

Get the number of tokens that have been used since the last update.

## Returns

Current value of the token counter

Definition at line 55 of file RoutingManagerPolicy.hh.

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

- `artdaq/artdaq/RoutingPolicies/RoutingManagerPolicy.hh`
- `artdaq/artdaq/RoutingPolicies/RoutingManagerPolicy.cc`

## 6.191 `artdaq::detail::RoutingPacketEntry` Struct Reference

A row of the Routing Table.

```
#include <artdaq/DAQrate/detail/RoutingPacket.hh>
```

### Public Member Functions

- [RoutingPacketEntry](#) ()  
*Default Constructor.*
- [RoutingPacketEntry](#) (Fragment::sequence\_id\_t seq, int rank)  
*Construct a [RoutingPacketEntry](#) with the given sequence ID and destination rank.*

## Public Attributes

- `Fragment::sequence_id_t sequence_id` {`Fragment::InvalidSequenceID`}  
The sequence ID of the [RoutingPacketEntry](#).
- `int32_t destination_rank` {-1}  
The destination rank for this sequence ID.

### 6.191.1 Detailed Description

A row of the Routing Table.

Definition at line 78 of file `RoutingPacket.hh`.

### 6.191.2 Constructor & Destructor Documentation

6.191.2.1 `artdaq::detail::RoutingPacketEntry::RoutingPacketEntry ( Fragment::sequence_id_t seq, int rank )` `[inline]`

Construct a [RoutingPacketEntry](#) with the given sequence ID and destination rank.

Parameters

<code>seq</code>	The sequence ID of the <a href="#">RoutingPacketEntry</a>
<code>rank</code>	The destination rank for this sequence ID

Definition at line 89 of file `RoutingPacket.hh`.

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

- `artdaq/artdaq/DAQrate/detail/RoutingPacket.hh`

## 6.192 artdaq::detail::RoutingPacketHeader Struct Reference

The header of the Routing Table, containing the magic bytes and the number of entries.

```
#include <artdaq/DAQrate/detail/RoutingPacket.hh>
```

## Public Member Functions

- [RoutingPacketHeader](#) (`size_t n`)  
Construct a [RoutingPacketHeader](#) declaring a given number of entries.
- [RoutingPacketHeader](#) ()  
Default Constructor.

## Public Attributes

- `uint32_t header` {0}  
Magic bytes to make sure the packet wasn't garbled.
- `uint64_t nEntries` {0}  
The number of [RoutingPacketEntries](#) in the [RoutingPacket](#).

### 6.192.1 Detailed Description

The header of the Routing Table, containing the magic bytes and the number of entries.

Definition at line 104 of file RoutingPacket.hh.

### 6.192.2 Constructor & Destructor Documentation

#### 6.192.2.1 `artdaq::detail::RoutingPacketHeader::RoutingPacketHeader ( size_t n )` `[inline]`, `[explicit]`

Construct a [RoutingPacketHeader](#) declaring a given number of entries.

Parameters

<code>n</code>	The number of RoutingPacketEntries in the associated RoutingPacket
----------------	--

Definition at line 113 of file RoutingPacket.hh.

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

- `artdaq/artdaq/DAQrate/detail/RoutingPacket.hh`

## 6.193 `artdaq::RoutingReceiverConfig` Struct Reference

Class which receives routing tables and prints updates.

### Public Attributes

- `fhicl::Atom< size_t > collection_time_ms` {fhicl::Name{"collection\_time\_ms"}, fhicl::Comment{"Time to collect routing table updates between printing summaries"}, 1000}  
*"collection\_time\_ms": Time to collect routing table updates between printing summaries*
- `fhicl::Atom< bool > print_verbose_info` {fhicl::Name{"print\_verbose\_info"}, fhicl::Comment{"Print verbose information about each receiver detected in routing tables"}, true}  
*"print\_verbose\_info" (Default: true): Print verbose information about each receiver detected in routing tables*
- `fhicl::Atom< size_t > graph_width` {fhicl::Name{"graph\_width"}, fhicl::Comment{"Width of the summary graph"}, 40}  
*"graph\_width": Width of the summary graph*
- `fhicl::OptionalTable`  
`< artdaq::TableReceiver::Config > routingTableConfig` {fhicl::Name{"routing\_table\_config"}, fhicl::Comment{"Configuration for the TableReceiver"}}  
*Configuration for the TableReceiver. See artdaq::TableReceiver::Config.*
- `fhicl::TableFragment`  
`< artdaq::artdaqapp::Config > artdaqAppConfig`  
*Configuration for artdaq Application (BoardReader, etc)*

### 6.193.1 Detailed Description

Class which receives routing tables and prints updates.

Definition at line 26 of file routingReceiver.cc.

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

- artdaq/proto/routingReceiver.cc

## 6.194 artdaq::detail::RoutingRequest Struct Reference

Represents a request sent to the RoutingManager for routing information.

```
#include <artdaq/DAQrate/detail/RoutingPacket.hh>
```

### Public Types

- enum [RequestMode](#) : uint8\_t { **Connect** = 0, **Disconnect** = 1, **Request** = 2, **Invalid** = 255 }
- The mode of this request, whether Request or Connect/Disconnect control messages.*

### Public Member Functions

- [RoutingRequest](#) (int r, [RequestMode](#) m=RequestMode::Connect)  
*Create a request using the given rank and mode.*
- [RoutingRequest](#) (int r, [Fragment::sequence\\_id\\_t](#) seq)  
*Create a [RoutingRequest](#) using the given rank and sequence ID.*
- [RoutingRequest](#) ()  
*Default constructor.*

### Static Public Member Functions

- static std::string [RequestModeToString](#) ([RequestMode](#) m)  
*Convert a RequestMode enumeration value to string.*

### Public Attributes

- uint32\_t [header](#) {0}  
*Magic bytes for identifying message type on wire.*
- int32\_t [rank](#) {-1}  
*The rank of the request sender.*
- [Fragment::sequence\\_id\\_t](#) [sequence\\_id](#) {artdaq::Fragment::InvalidSequenceID}  
*The sequence ID being requested in Request mode.*
- [RequestMode](#) [mode](#) {RequestMode::Invalid}  
*Mode of the request.*

#### 6.194.1 Detailed Description

Represents a request sent to the RoutingManager for routing information.

Definition at line 124 of file RoutingPacket.hh.

## 6.194.2 Constructor & Destructor Documentation

6.194.2.1 `artdaq::detail::RoutingRequest::RoutingRequest ( int r, RequestMode m = RequestMode::Connect )`  
`[inline]`

Create a request using the given rank and mode.

## Parameters

<i>r</i>	Rank of the requestor
<i>m</i>	Mode of this request

This constructor is primarily used to send RequestMode::Connect and RequestMode::Disconnect control messages

Definition at line 170 of file RoutingPacket.hh.

#### 6.194.2.2 artdaq::detail::RoutingRequest::RoutingRequest ( int *r*, Fragment::sequence\_id\_t *seq* ) [inline]

Create a [RoutingRequest](#) using the given rank and sequence ID.

## Parameters

<i>r</i>	Rank of the requestor
<i>seq</i>	Sequence ID of request

Definition at line 178 of file RoutingPacket.hh.

### 6.194.3 Member Function Documentation

#### 6.194.3.1 static std::string artdaq::detail::RoutingRequest::RequestModeToString ( RequestMode *m* ) [inline], [static]

Convert a RequestMode enumeration value to string.

## Parameters

<i>m</i>	RequestMode to convert
----------	------------------------

## Returns

String representation of RequestMode

Definition at line 142 of file RoutingPacket.hh.

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

- artdaq/artdaq/DAQrate/detail/RoutingPacket.hh

## 6.195 artdaq::detail::RoutingToken Struct Reference

The [RoutingToken](#) contains the magic bytes, the rank of the token sender, and the number of slots free. This is a TCP message, so additional verification is not necessary.

```
#include <artdaq/DAQrate/detail/RoutingPacket.hh>
```

## Public Attributes

- uint32\_t [header](#)  
The magic bytes that help validate the [RoutingToken](#).
- int [rank](#)  
The rank from which the [RoutingToken](#) came.

- unsigned [new\\_slots\\_free](#)  
*The number of slots free in the token sender (usually 1)*
- unsigned [run\\_number](#)  
*The Run with which this token should be associated.*

### 6.195.1 Detailed Description

The [RoutingToken](#) contains the magic bytes, the rank of the token sender, and the number of slots free. This is a TCP message, so additional verification is not necessary.

Definition at line 193 of file RoutingPacket.hh.

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

- `artdaq/artdaq/DAQrate/detail/RoutingPacket.hh`

## 6.196 artdaq::RTIDDS Class Reference

DDS Transport Implementation.

```
#include <artdaq/RTIDDS/RTIDDS.hh>
```

### Classes

- class [OctetsListener](#)  
*A class that reads data from DDS.*

### Public Types

- enum [IOType](#) { **reader**, **writer** }  
*Whether this DDS instance is a reader or a writer.*

### Public Member Functions

- [RTIDDS](#) (std::string name, [IOType](#) iotype, std::string max\_size="1000000")  
*Construct a [RTIDDS](#) transmitter.*
- virtual [~RTIDDS](#) ()=default  
*Default virtual Destructor.*
- void [transfer\\_fragment\\_min\\_blocking\\_mode\\_via\\_DDS\\_](#) (artdaq::Fragment const &fragment)  
*Copy a Fragment to DDS.*
- void [transfer\\_fragment\\_reliable\\_mode\\_via\\_DDS\\_](#) (artdaq::Fragment &&fragment)  
*Move a Fragment to DDS.*

### Public Attributes

- [OctetsListener](#) [octets\\_listener\\_](#)  
*The receiver.*



### 6.196.1 Detailed Description

DDS Transport Implementation.

Definition at line 20 of file RTIDDS.hh.

### 6.196.2 Constructor & Destructor Documentation

#### 6.196.2.1 artdaq::RTIDDS::RTIDDS ( std::string *name*, IOType *iotype*, std::string *max\_size* = "1000000" )

Construct a [RTIDDS](#) transmitter.

Parameters

<i>name</i>	Name of the module
<i>iotype</i>	Direction of transmission
<i>max_size</i>	Maximum size to transmit

Definition at line 11 of file RTIDDS.cc.

### 6.196.3 Member Function Documentation

#### 6.196.3.1 void artdaq::RTIDDS::transfer\_fragment\_min\_blocking\_mode\_via\_DDS\_ ( artdaq::Fragment const & *fragment* )

Copy a Fragment to DDS.

Parameters

<i>fragment</i>	Fragment to copy
-----------------	------------------

This function may be non-reliable, and induces a memcpy of the Fragment

Definition at line 109 of file RTIDDS.cc.

#### 6.196.3.2 void artdaq::RTIDDS::transfer\_fragment\_reliable\_mode\_via\_DDS\_ ( artdaq::Fragment && *fragment* )

Move a Fragment to DDS.

Parameters

<i>fragment</i>	Fragment to move
-----------------	------------------

This function should be reliable, and minimize copies. Currently implemented via transfer\_fragment\_min\_blocking\_mode\_via\_DDS\_

Definition at line 107 of file RTIDDS.cc.

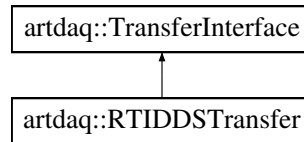
The documentation for this class was generated from the following files:

- artdaq/artdaq/RTIDDS/RTIDDS.hh
- artdaq/artdaq/RTIDDS/RTIDDS.cc

## 6.197 artdaq::RTIDDSTransfer Class Reference

[RTIDDSTransfer](#) is a [TransferInterface](#) implementation plugin that transfers data using RTI DDS.

Inheritance diagram for artdaq::RTIDDSTransfer:



## Public Member Functions

- virtual `~RTIDDSTransfer()`=default  
*RTIDDSTransfer default Destructor.*
- `RTIDDSTransfer(fhicl::ParameterSet const &ps, Role role)`  
*RTIDDSTransfer Constructor.*
- int `receiveFragment` (artdaq::Fragment &fragment, size\_t receiveTimeout) override  
*Receive a Fragment using DDS.*
- `CopyStatus transfer_fragment_min_blocking_mode` (artdaq::Fragment const &fragment, size\_t send\_timeout\_usec=std::numeric\_limits< size\_t >::max()) override  
*Transfer a Fragment to the destination. May not necessarily be reliable, but will not block longer than send\_timeout\_usec.*
- `CopyStatus transfer_fragment_reliable_mode` (artdaq::Fragment &&fragment) override  
*Transfer a Fragment to the destination. This should be reliable, if the underlying transport mechanism supports reliable sending.*
- bool `isRunning` () override  
*Determine whether the TransferInterface plugin is able to send/receive data.*
- void `flush_buffers` () override  
*Flush any in-flight data. This should be used by the receiver after the receive loop has ended.*

## Additional Inherited Members

### 6.197.1 Detailed Description

`RTIDDSTransfer` is a `TransferInterface` implementation plugin that transfers data using RTI DDS.

Definition at line 19 of file `RTIDDS_transfer.cc`.

### 6.197.2 Constructor & Destructor Documentation

6.197.2.1 `artdaq::RTIDDSTransfer::RTIDDSTransfer ( fhicl::ParameterSet const &ps, Role role )` `[inline]`

`RTIDDSTransfer` Constructor.

Parameters

<i>ps</i>	ParameterSet used to configure <code>RTIDDSTransfer</code>
<i>role</i>	Role of this <code>RTIDDSTransfer</code> instance (kSend or kReceive)

`RTIDDSTransfer` only requires the Parameters for configuring a `TransferInterface`

Definition at line 34 of file `RTIDDS_transfer.cc`.

### 6.197.3 Member Function Documentation

6.197.3.1 `bool artdaq::RTIDDSTransfer::isRunning ( ) [inline],[override],[virtual]`

Determine whether the [TransferInterface](#) plugin is able to send/receive data.

#### Returns

True if the [TransferInterface](#) plugin is currently able to send/receive data

Reimplemented from [artdaq::TransferInterface](#).

Definition at line 68 of file RTIDDS\_transfer.cc.

6.197.3.2 `int artdaq::RTIDDSTransfer::receiveFragment ( artdaq::Fragment & fragment, size_t receiveTimeout ) [override],[virtual]`

Receive a Fragment using DDS.

#### Parameters

<i>out</i>	<i>fragment</i>	Received Fragment
	<i>receiveTimeout</i>	<a href="#">Timeout</a> for receive, in microseconds

#### Returns

Rank of sender or RECV\_TIMEOUT

Reimplemented from [artdaq::TransferInterface](#).

Definition at line 91 of file RTIDDS\_transfer.cc.

6.197.3.3 `artdaq::TransferInterface::CopyStatus artdaq::RTIDDSTransfer::transfer_fragment_min_blocking_mode ( artdaq::Fragment const & fragment, size_t send_timeout_usec = std::numeric_limits<size_t>::max() ) [override],[virtual]`

Transfer a Fragment to the destination. May not necessarily be reliable, but will not block longer than *send\_timeout\_usec*.

#### Parameters

<i>fragment</i>	Fragment to transfer
<i>send_timeout_usec</i>	<a href="#">Timeout</a> for send, in microseconds

#### Returns

CopyStatus detailing result of transfer

Implements [artdaq::TransferInterface](#).

Definition at line 136 of file RTIDDS\_transfer.cc.

6.197.3.4 `artdaq::TransferInterface::CopyStatus artdaq::RTIDDSTransfer::transfer_fragment_reliable_mode ( artdaq::Fragment && fragment ) [override],[virtual]`

Transfer a Fragment to the destination. This should be reliable, if the underlying transport mechanism supports reliable sending.

## Parameters

<i>fragment</i>	Fragment to transfer
-----------------	----------------------

## Returns

CopyStatus detailing result of copy

Implements [artdaq::TransferInterface](#).

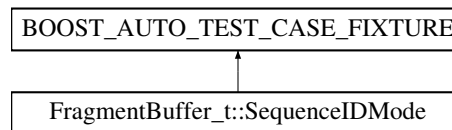
Definition at line 129 of file RTIDDS\_transfer.cc.

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

- artdaq/artdaq/TransferPlugins/RTIDDS\_transfer.cc

## 6.198 FragmentBuffer\_t::SequenceIDMode Struct Reference

Inheritance diagram for FragmentBuffer\_t::SequenceIDMode:



## Public Member Functions

- void **test\_method** ()

### 6.198.1 Detailed Description

Definition at line 1092 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.199 FragmentBuffer\_t::SequenceIDMode\_id Struct Reference

### 6.199.1 Detailed Description

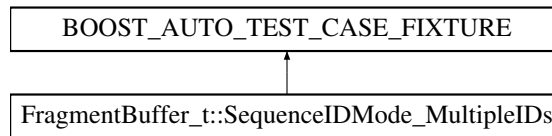
Definition at line 1092 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.200 FragmentBuffer\_t::SequenceIDMode\_MultipleIDs Struct Reference

Inheritance diagram for FragmentBuffer\_t::SequenceIDMode\_MultipleIDs:



### Public Member Functions

- void **test\_method** ()

#### 6.200.1 Detailed Description

Definition at line 1911 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.201 FragmentBuffer\_t::SequenceIDMode\_MultipleIDs\_id Struct Reference

#### 6.201.1 Detailed Description

Definition at line 1911 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.202 art::ServicesConfig Struct Reference

Configuration of the services block for artdaq art processes

```
#include <artdaq/ArtModules/detail/ArtConfig.hh>
```

### Public Attributes

- fhicl::Table  
 < [ArtdaqSharedMemoryServiceInterfaceConfig](#) > [ArtdaqSharedMemoryServiceInterface](#) {fhicl::Name{"Artdaq-SharedMemoryServiceInterface"}}  
*Configuration for the [ArtdaqSharedMemoryServiceInterface](#).*
- fhicl::OptionalTable  
 < [ArtdaqFragmentNamingServiceInterfaceConfig](#) > [ArtdaqFragmentNamingServiceInterface](#) {fhicl::Name{"Artdaq-FragmentNamingServiceInterface"}}  
*Configuration for the [ArtdaqFragmentNamingServiceInterface](#).*

### 6.202.1 Detailed Description

Configuration of the services block for artdaq art processes

Definition at line 53 of file ArtConfig.hh.

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

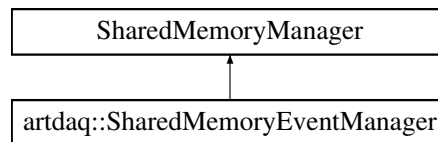
- artdaq/artdaq/ArtModules/detail/ArtConfig.hh

## 6.203 artdaq::SharedMemoryEventManager Class Reference

The [SharedMemoryEventManager](#) is a SharedMemoryManger which tracks events as they are built.

```
#include <artdaq/DAQrate/SharedMemoryEventManager.hh>
```

Inheritance diagram for artdaq::SharedMemoryEventManager:



### Classes

- struct [Config](#)  
*Configuration of the [SharedMemoryEventManager](#). May be used for parameter validation*

### Public Types

- typedef RawEvent::run\_id\_t [run\\_id\\_t](#)  
*Copy RawEvent::run\_id\_t into local scope.*
- typedef RawEvent::subrun\_id\_t [subrun\\_id\\_t](#)  
*Copy RawEvent::subrun\_id\_t into local scope.*
- typedef Fragment::sequence\_id\_t [sequence\\_id\\_t](#)  
*Copy Fragment::sequence\_id\_t into local scope.*
- typedef std::map  
 < [sequence\\_id\\_t](#), RawEvent\_ptr > [EventManager](#)  
*An EventMap is a map of RawEvent\_ptr objects, keyed by sequence ID.*
- using [Parameters](#) = fhicl::WrappedTable< [Config](#) >  
*Used for ParameterSet validation (if desired)*

### Public Member Functions

- [SharedMemoryEventManager](#) (const fhicl::ParameterSet &pset, fhicl::ParameterSet art\_pset)  
*SharedMemoryEventManager Constructor.*
- virtual [~SharedMemoryEventManager](#) ()  
*SharedMemoryEventManager Destructor.*

- bool [AddFragment](#) (FragmentPtr frag, size\_t timeout\_usec, FragmentPtr &outfrag)  
*Copy a Fragment into the [SharedMemoryEventManager](#).*
- RawDataType \* [WriteFragmentHeader](#) (detail::RawFragmentHeader frag, bool dropIfNoBuffersAvailable=false)  
*Get a pointer to a reserved memory area for the given Fragment header.*
- void [DoneWritingFragment](#) (detail::RawFragmentHeader frag)  
*Used to indicate that the given Fragment is now completely in the buffer. Will check for buffer completeness, and unset the pending flag.*
- size\_t [GetOpenEventCount](#) ()  
*Returns the number of buffers which contain data but are not yet complete.*
- size\_t [GetPendingEventCount](#) ()  
*Returns the number of events which are complete but waiting on lower sequenced events to finish.*
- size\_t [GetLockedBufferCount](#) ()  
*Returns the number of buffers currently owned by this manager.*
- size\_t [GetArtEventCount](#) ()  
*Returns the number of events sent to art this run.*
- size\_t [GetFragmentCount](#) (Fragment::sequence\_id\_t seqID, Fragment::type\_t type=Fragment::InvalidFragmentType)  
*Get the count of Fragments of a given type in an event.*
- size\_t [GetFragmentCountInBuffer](#) (int buffer, Fragment::type\_t type=Fragment::InvalidFragmentType)  
*Get the count of Fragments of a given type in a buffer.*
- void [UpdateFragmentHeader](#) (int buffer, detail::RawFragmentHeader hdr)
- void [RunArt](#) (size\_t process\_index, const std::shared\_ptr< std::atomic< pid\_t >> &pid\_out)  
*Run an art instance, recording the return codes and restarting it until the end flag is raised.*
- void [StartArt](#) ()  
*Start all the art processes.*
- pid\_t [StartArtProcess](#) (fhicl::ParameterSet pset, size\_t process\_index)  
*Start one art process.*
- void [ShutdownArtProcesses](#) (std::set< pid\_t > &pids)  
*Shutdown a set of art processes.*
- void [ReconfigureArt](#) (fhicl::ParameterSet art\_pset, run\_id\_t newRun=0, int n\_art\_processes=-1)  
*Restart all art processes, using the given fhicl code to configure the new art processes.*
- bool [endOfData](#) ()  
*Indicate that the end of input has been reached to the art processes.*
- void [startRun](#) (run\_id\_t runID)  
*Start a Run.*
- run\_id\_t [runID](#) () const  
*Get the current Run number.*
- bool [endRun](#) ()  
*Send an EndOfRunFragment to the art thread.*
- void [rolloverSubrun](#) (sequence\_id\_t boundary, subrun\_id\_t subrun)  
*Rollover the subrun after the specified event.*
- void [rolloverSubrun](#) ()  
*Add a subrun transition immediately after the highest currently define sequence ID.*
- void [sendMetrics](#) ()  
*Send metrics to the MetricManager, if one has been instantiated in the application.*
- void [setRequestMode](#) (detail::RequestMessageMode mode)  
*Set the RequestMessageMode for all outgoing data requests.*

- void [setOverwrite](#) (bool overwrite)  
*Set the overwrite flag (non-reliable data transfer) for the Shared Memory.*
- void [AddInitFragment](#) (FragmentPtr &frag)  
*Set the stored Init fragment, if one has not yet been set already.*
- uint32\_t [GetBroadcastKey](#) ()  
*Gets the shared memory key of the broadcast SharedMemoryManager.*
- RawDataType \* [GetDroppedDataAddress](#) (detail::RawFragmentHeader frag)  
*Gets the address of the "dropped data" fragment. Used for testing.*
- void [UpdateArtConfiguration](#) (fhicl::ParameterSet art\_pset)  
*Updates the internally-stored copy of the art configuration.*
- void [CheckPendingBuffers](#) ()  
*Check for buffers which are ready to be marked incomplete and released to art and issue tokens for any buffers which are available.*
- subrun\_id\_t [GetSubrunForSequenceID](#) (Fragment::sequence\_id\_t seqID)  
*Get the subrun number that the given Sequence ID would be assigned to.*
- subrun\_id\_t [GetCurrentSubrun](#) ()  
*Get the current subrun number (Gets the last defined subrun)*
- std::string [BuildStatisticsString](#) () const

## Static Public Attributes

- static const std::string [FRAGMENTS\\_RECEIVED\\_STAT\\_KEY](#)  
*Key for Fragments Received MonitoredQuantity.*
- static const std::string [EVENTS\\_RELEASED\\_STAT\\_KEY](#)  
*Key for the Events Released MonitoredQuantity.*

### 6.203.1 Detailed Description

The [SharedMemoryEventManager](#) is a SharedMemoryManger which tracks events as they are built.

Definition at line 101 of file SharedMemoryEventManager.hh.

### 6.203.2 Constructor & Destructor Documentation

- 6.203.2.1 `artdaq::SharedMemoryEventManager::SharedMemoryEventManager ( const fhicl::ParameterSet & pset, fhicl::ParameterSet art_pset )`

[SharedMemoryEventManager](#) Constructor.

Parameters

<i>pset</i>	ParameterSet used to configure <a href="#">SharedMemoryEventManager</a> . See <a href="#">artdaq::SharedMemoryEventManager::Config</a> for description of parameters
<i>art_pset</i>	ParameterSet used to configure art. See <a href="#">art::Config</a> for description of expected document format

Definition at line 25 of file SharedMemoryEventManager.cc.



### 6.203.3 Member Function Documentation

6.203.3.1 `bool artdaq::SharedMemoryEventManager::AddFragment ( FragmentPtr frag, size_t timeout_usec, FragmentPtr & outfrag )`

Copy a Fragment into the [SharedMemoryEventManager](#).

**Parameters**

	<i>frag</i>	FragmentPtr object
	<i>timeout_usec</i>	<a href="#">Timeout</a> for adding Fragment to the Shared Memory
<i>out</i>	<i>outfrag</i>	Rejected Fragment if timeout occurs

**Returns**

Whether the Fragment was successfully added

Definition at line 189 of file SharedMemoryEventManager.cc.

**6.203.3.2 void artdaq::SharedMemoryEventManager::DoneWritingFragment ( detail::RawFragmentHeader *frag* )**

Used to indicate that the given Fragment is now completely in the buffer. Will check for buffer completeness, and unset the pending flag.

**Parameters**

<i>frag</i>	Fragment that is now completely in the buffer.
-------------	--

Definition at line 299 of file SharedMemoryEventManager.cc.

**6.203.3.3 bool artdaq::SharedMemoryEventManager::endOfData ( )**

Indicate that the end of input has been reached to the art processes.

**Returns**

True if the end proceeded correctly

Put the end-of-data marker onto the RawEvent queue (if possible), wait for the reader function to exit, and fill in the reader return value. This scenario returns true. If the end-of-data marker can not be pushed onto the RawEvent queue, false is returned.

Definition at line 796 of file SharedMemoryEventManager.cc.

**6.203.3.4 bool artdaq::SharedMemoryEventManager::endRun ( )**

Send an EndOfRunFragment to the art thread.

**Returns**

True if enqueue successful

Definition at line 943 of file SharedMemoryEventManager.cc.

**6.203.3.5 size\_t artdaq::SharedMemoryEventManager::GetArtEventCount ( ) [inline]**

Returns the number of events sent to art this run.

**Returns**

The number of events sent to art this run

Definition at line 251 of file SharedMemoryEventManager.hh.

### 6.203.3.6 `uint32_t artdaq::SharedMemoryEventManager::GetBroadcastKey ( ) [inline]`

Gets the shared memory key of the broadcast SharedMemoryManager.

#### Returns

The shared memory key of the broadcast SharedMemoryManager

Definition at line 372 of file SharedMemoryEventManager.hh.

### 6.203.3.7 `subrun_id_t artdaq::SharedMemoryEventManager::GetCurrentSubrun ( ) [inline]`

Get the current subrun number (Gets the last defined subrun)

#### Returns

Number of the subrun that corresponds to events with the maximum possible sequence ID.

Definition at line 418 of file SharedMemoryEventManager.hh.

### 6.203.3.8 `RawDataType* artdaq::SharedMemoryEventManager::GetDroppedDataAddress ( detail::RawFragmentHeader frag ) [inline]`

Gets the address of the "dropped data" fragment. Used for testing.

#### Parameters

<i>frag</i>	Fragment ID to get "dropped data" for
-------------	---------------------------------------

#### Returns

Pointer to the data payload of the "dropped data" fragment

Definition at line 379 of file SharedMemoryEventManager.hh.

### 6.203.3.9 `size_t artdaq::SharedMemoryEventManager::GetFragmentCount ( Fragment::sequence_id_t seqID, Fragment::type_t type = Fragment::InvalidFragmentType )`

Get the count of Fragments of a given type in an event.

#### Parameters

<i>seqID</i>	Sequence ID of Fragments
<i>type</i>	Type of fragments to count. Use InvalidFragmentType to count all fragments (default)

#### Returns

Number of Fragments in event of given type

Definition at line 382 of file SharedMemoryEventManager.cc.

### 6.203.3.10 `size_t artdaq::SharedMemoryEventManager::GetFragmentCountInBuffer ( int buffer, Fragment::type_t type = Fragment::InvalidFragmentType )`

Get the count of Fragments of a given type in a buffer.

## Parameters

<i>buffer</i>	Buffer to count
<i>type</i>	Type of fragments to count. Use InvalidFragmentType to count all fragments (default)

## Returns

Number of Fragments in buffer of given type

Definition at line 387 of file SharedMemoryEventManager.cc.

**6.203.3.11** `size_t artdaq::SharedMemoryEventManager::GetLockedBufferCount ( ) [inline]`

Returns the number of buffers currently owned by this manager.

## Returns

The number of buffers currently owned by this manager

Definition at line 245 of file SharedMemoryEventManager.hh.

**6.203.3.12** `size_t artdaq::SharedMemoryEventManager::GetOpenEventCount ( ) [inline]`

Returns the number of buffers which contain data but are not yet complete.

## Returns

The number of buffers which contain data but are not yet complete

Definition at line 233 of file SharedMemoryEventManager.hh.

**6.203.3.13** `size_t artdaq::SharedMemoryEventManager::GetPendingEventCount ( ) [inline]`

Returns the number of events which are complete but waiting on lower sequenced events to finish.

## Returns

The number of events which are complete but waiting on lower sequenced events to finish

Definition at line 239 of file SharedMemoryEventManager.hh.

**6.203.3.14** `artdaq::SharedMemoryEventManager::subrun_id_t artdaq::SharedMemoryEventManager::GetSubrunForSequenceID ( Fragment::sequence_id_t seqID )`

Get the subrun number that the given Sequence ID would be assigned to.

## Parameters

<i>seqID</i>	Sequence ID to check
--------------	----------------------

**Returns**

Subrun number that the given sequence ID will be associated with

Definition at line 1104 of file SharedMemoryEventManager.cc.

**6.203.3.15** `void artdaq::SharedMemoryEventManager::ReconfigureArt ( fhicl::ParameterSet art_pset, run_id_t newRun = 0, int n_art_processes = -1 )`

Restart all art processes, using the given fhicl code to configure the new art processes.

**Parameters**

<i>art_pset</i>	ParameterSet used to configure art
<i>newRun</i>	New Run number for reconfigured art
<i>n_art_processes</i>	Number of art processes to start, -1 (default) leaves the number unchanged

Definition at line 762 of file SharedMemoryEventManager.cc.

**6.203.3.16** `void artdaq::SharedMemoryEventManager::rolloverSubrun ( sequence_id_t boundary, subrun_id_t subrun )`

Rollover the subrun after the specified event.

**Parameters**

<i>boundary</i>	sequence ID of the boundary (Event with seqID == boundary will be in new subrun)
<i>subrun</i>	Subrun number of subrun after boundary

Definition at line 973 of file SharedMemoryEventManager.cc.

**6.203.3.17** `run_id_t artdaq::SharedMemoryEventManager::runID ( ) const` `[inline]`

Get the current Run number.

**Returns**

The current Run number

Definition at line 323 of file SharedMemoryEventManager.hh.

**6.203.3.18** `void artdaq::SharedMemoryEventManager::setOverwrite ( bool overwrite )` `[inline]`

Set the overwrite flag (non-reliable data transfer) for the Shared Memory.

**Parameters**

<i>overwrite</i>	Whether to allow the writer to overwrite data that has not yet been read
------------------	--

Definition at line 361 of file SharedMemoryEventManager.hh.

6.203.3.19 `void artdaq::SharedMemoryEventManager::setRequestMode ( detail::RequestMessageMode mode )`  
`[inline]`

Set the RequestMessageMode for all outgoing data requests.

## Parameters

<i>mode</i>	Mode to set
-------------	-------------

Definition at line 352 of file SharedMemoryEventManager.hh.

6.203.3.20 void artdaq::SharedMemoryEventManager::ShutdownArtProcesses ( std::set< pid\_t > & pids )

Shutdown a set of art processes.

## Parameters

<i>pids</i>	PIDs of the art processes
-------------	---------------------------

Definition at line 624 of file SharedMemoryEventManager.cc.

6.203.3.21 pid\_t artdaq::SharedMemoryEventManager::StartArtProcess ( fhicl::ParameterSet pset, size\_t process\_index )

Start one art process.

## Parameters

<i>pset</i>	ParameterSet to send to this art process
<i>process_index</i>	Index of this art process (when starting multiple)

## Returns

pid\_t of the started process

Definition at line 579 of file SharedMemoryEventManager.cc.

6.203.3.22 void artdaq::SharedMemoryEventManager::startRun ( run\_id\_t runID )

Start a Run.

## Parameters

<i>runID</i>	Run number of the new run
--------------	---------------------------

Definition at line 895 of file SharedMemoryEventManager.cc.

6.203.3.23 void artdaq::SharedMemoryEventManager::UpdateArtConfiguration ( fhicl::ParameterSet art\_pset )

Updates the internally-stored copy of the art configuration.

## Parameters

<i>art_pset</i>	ParameterSet used to configure art
-----------------	------------------------------------

This method updates the internally-stored copy of the art configuration, but it does not restart art processes. So, if this method is called while art processes are running, it will have no effect until the next restart, such as the next Start of run. Typically, this method is intended to be called between runs, when no art processes are running.

Definition at line 1523 of file SharedMemoryEventManager.cc.

6.203.3.24 `artdaq::RawDataType * artdaq::SharedMemoryEventManager::WriteFragmentHeader ( detail::RawFragmentHeader frag,  
bool dropIfNoBuffersAvailable = false )`

Get a pointer to a reserved memory area for the given Fragment header.



## Parameters

<i>frag</i>	Fragment header (contains sequence ID and size information)
<i>dropIfNoBuffers-Available</i>	Whether to drop the fragment (instead of returning nullptr) when no buffers are available (Default: false)

## Returns

Pointer to memory location for Fragment body (Header is copied into buffer here)

Definition at line 212 of file SharedMemoryEventManager.cc.

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

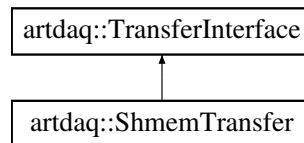
- artdaq/artdaq/DAQrate/SharedMemoryEventManager.hh
- artdaq/artdaq/DAQrate/SharedMemoryEventManager.cc

## 6.204 artdaq::ShmemTransfer Class Reference

A [TransferInterface](#) implementation plugin that transfers data using Shared Memory.

```
#include <artdaq/TransferPlugins/ShmemTransfer.hh>
```

Inheritance diagram for artdaq::ShmemTransfer:



### Public Member Functions

- [ShmemTransfer](#) (fhicl::ParameterSet const &pset, [Role](#) role)  
*ShmemTransfer* Constructor.
- virtual [~ShmemTransfer](#) () noexcept  
*ShmemTransfer* Destructor.
- int [receiveFragment](#) (Fragment &fragment, size\_t receiveTimeout) override  
*Receive a Fragment from Shared Memory.*
- int [receiveFragmentHeader](#) (detail::RawFragmentHeader &header, size\_t receiveTimeout) override  
*Receive a Fragment Header from the transport mechanism.*
- int [receiveFragmentData](#) (RawDataType \*destination, size\_t word\_count) override  
*Receive the body of a Fragment to the given destination pointer.*
- [CopyStatus](#) [transfer\\_fragment\\_min\\_blocking\\_mode](#) (Fragment const &fragment, size\_t send\_timeout\_usec) override  
*Transfer a Fragment to the destination. May not necessarily be reliable, but will not block longer than send\_timeout\_usec.*
- [CopyStatus](#) [transfer\\_fragment\\_reliable\\_mode](#) (Fragment &&fragment) override  
*Transfer a Fragment to the destination. This should be reliable, if the underlying transport mechanism supports reliable sending.*
- bool [isRunning](#) () override

Determine whether the [TransferInterface](#) plugin is able to send/receive data.

- void [flush\\_buffers](#) () override

Flush any in-flight data. This should be used by the receiver after the receive loop has ended.

## Additional Inherited Members

### 6.204.1 Detailed Description

A [TransferInterface](#) implementation plugin that transfers data using Shared Memory.

Definition at line 13 of file ShmemTransfer.hh.

### 6.204.2 Constructor & Destructor Documentation

#### 6.204.2.1 `artdaq::ShmemTransfer::ShmemTransfer ( fhicl::ParameterSet const & pset, Role role )`

[ShmemTransfer](#) Constructor.

Parameters

<i>pset</i>	ParameterSet used to configure <a href="#">ShmemTransfer</a>
<i>role</i>	Role of this <a href="#">ShmemTransfer</a> instance (kSend or kReceive)

```
* ShmemTransfer accepts the following Parameters:
* "shm_key_offset" (Default: 0): Offset to add to shared memory key (hash of uniqueLabel)
*
```

[ShmemTransfer](#) also requires all Parameters for configuring a [TransferInterface](#) Additionally, an offset can be added via the ARTDAQ\_SHMEM\_TRANSFER\_OFFSET environment variable. Note that this variable, if used, MUST have the same value for all artdaq processes communicating via [ShmemTransfer](#).

Definition at line 12 of file ShmemTransfer.cc.

### 6.204.3 Member Function Documentation

#### 6.204.3.1 `bool artdaq::ShmemTransfer::isRunning ( ) [override],[virtual]`

Determine whether the [TransferInterface](#) plugin is able to send/receive data.

Returns

True if the [TransferInterface](#) plugin is currently able to send/receive data

Reimplemented from [artdaq::TransferInterface](#).

Definition at line 237 of file ShmemTransfer.cc.

#### 6.204.3.2 `int artdaq::ShmemTransfer::receiveFragment ( Fragment & fragment, size_t receiveTimeout ) [override]`

Receive a Fragment from Shared Memory.

## Parameters

out	fragment	Received Fragment
	receiveTimeout	<a href="#">Timeout</a> for receive, in microseconds

## Returns

Rank of sender or RECV\_TIMEOUT

Definition at line 57 of file ShmemTransfer.cc.

**6.204.3.3** `int artdaq::ShmemTransfer::receiveFragmentData ( RawDataType * destination, size_t word_count )` [\[override\]](#), [\[virtual\]](#)

Receive the body of a Fragment to the given destination pointer.

## Parameters

destination	Pointer to memory region where Fragment data should be stored
word_count	Number of words of Fragment data to receive

## Returns

The rank the Fragment was received from (should be source\_rank), or RECV\_TIMEOUT

Implements [artdaq::TransferInterface](#).

Definition at line 154 of file ShmemTransfer.cc.

**6.204.3.4** `int artdaq::ShmemTransfer::receiveFragmentHeader ( detail::RawFragmentHeader & header, size_t receiveTimeout )` [\[override\]](#), [\[virtual\]](#)

Receive a Fragment Header from the transport mechanism.

## Parameters

out	header	Received Fragment Header
	receiveTimeout	<a href="#">Timeout</a> for receive

## Returns

The rank the Fragment was received from (should be source\_rank), or RECV\_TIMEOUT

Implements [artdaq::TransferInterface](#).

Definition at line 105 of file ShmemTransfer.cc.

**6.204.3.5** `artdaq::TransferInterface::CopyStatus artdaq::ShmemTransfer::transfer_fragment_min_blocking_mode ( Fragment const & fragment, size_t send_timeout_usec )` [\[override\]](#)

Transfer a Fragment to the destination. May not necessarily be reliable, but will not block longer than send\_timeout\_usec.

## Parameters

<i>fragment</i>	Fragment to transfer
<i>send_timeout_ - usec</i>	<a href="#">Timeout</a> for send, in microseconds

## Returns

CopyStatus detailing result of transfer

Definition at line 172 of file ShmemTransfer.cc.

**6.204.3.6** `artdaq::TransferInterface::CopyStatus` `artdaq::ShmemTransfer::transfer_fragment_reliable_mode ( Fragment && fragment )` `[override]`

Transfer a Fragment to the destination. This should be reliable, if the underlying transport mechanism supports reliable sending.

## Parameters

<i>fragment</i>	Fragment to transfer
-----------------	----------------------

## Returns

CopyStatus detailing result of copy

Definition at line 178 of file ShmemTransfer.cc.

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

- `artdaq/artdaq/TransferPlugins/ShmemTransfer.hh`
- `artdaq/artdaq/TransferPlugins/ShmemTransfer.cc`

## 6.205 `art::ShmemWrapper` Class Reference

This class wraps [ArtdaqSharedMemoryService](#) so that it can act as an ArtdaqInput template class.

```
#include <artdaq/ArtModules/detail/ShmemWrapper.hh>
```

### Public Member Functions

- [ShmemWrapper](#) (fhicl::ParameterSet const &ps)  
*ShmemWrapper* Constructor.
- virtual `~ShmemWrapper ()`=default  
*ShmemWrapper* Destructor.
- `artdaq::FragmentPtrs` [receiveMessage \(\)](#)  
Receive a message from the [ArtdaqSharedMemoryService](#).
- `std::unordered_map`  
  `< artdaq::Fragment::type_t,`  
  `std::unique_ptr`  
  `< artdaq::Fragments > >` [receiveMessages \(\)](#)

*Receive all messages for an event from [ArtdaqSharedMemoryService](#).*

- `artdaq::FragmentPtrs` [receiveInitMessage](#) ()

*Receive an init message from the [ArtdaqSharedMemoryService](#).*

- `std::shared_ptr`  
< `artdaq::detail::RawEventHeader` > [getEventHeader](#) ()

*Get a pointer to the last received `RawEventHeader`.*

### 6.205.1 Detailed Description

This class wraps [ArtdaqSharedMemoryService](#) so that it can act as an `ArtdaqInput` template class.

JCF, May-27-2016

This class is written with functionality such that it satisfies the requirements needed to be a template in the `ArtdaqInput` class

Definition at line 25 of file `ShmemWrapper.hh`.

### 6.205.2 Constructor & Destructor Documentation

#### 6.205.2.1 `art::ShmemWrapper::ShmemWrapper ( fhicl::ParameterSet const & ps )`

[ShmemWrapper](#) Constructor.

Parameters

<i>ps</i>	ParameterSet for <a href="#">ShmemWrapper</a>
-----------	---

Definition at line 10 of file `ShmemWrapper.cc`.

### 6.205.3 Member Function Documentation

#### 6.205.3.1 `std::shared_ptr<artdaq::detail::RawEventHeader> art::ShmemWrapper::getEventHeader ( ) [inline]`

Get a pointer to the last received `RawEventHeader`.

Returns

a `shared_ptr` to the last received `RawEventHeader`

Definition at line 60 of file `ShmemWrapper.hh`.

#### 6.205.3.2 `artdaq::FragmentPtrs art::ShmemWrapper::receiveInitMessage ( )`

Receive an init message from the [ArtdaqSharedMemoryService](#).

Returns

A list of `unique_ptr`s to `InitFragments`

Definition at line 114 of file `ShmemWrapper.cc`.

### 6.205.3.3 `artdaq::FragmentPtrs art::ShmemWrapper::receiveMessage ( )`

Receive a message from the [ArtdaqSharedMemoryService](#).

#### Returns

A list of unique\_ptrs to received Fragments

Definition at line 17 of file ShmemWrapper.cc.

### 6.205.3.4 `std::unordered_map< artdaq::Fragment::type_t, std::unique_ptr< artdaq::Fragments > > art::ShmemWrapper::receiveMessages ( )`

Receive all messages for an event from [ArtdaqSharedMemoryService](#).

#### Returns

A map of `Fragment::type_t` to a unique\_ptr to Fragments containing all Fragments in an event

Definition at line 81 of file ShmemWrapper.cc.

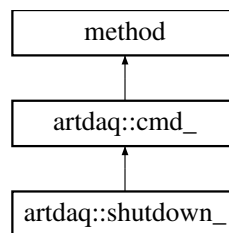
The documentation for this class was generated from the following files:

- `artdaq/artdaq/ArtModules/detail/ShmemWrapper.hh`
- `artdaq/artdaq/ArtModules/detail/ShmemWrapper.cc`

## 6.206 `artdaq::shutdown_` Class Reference

[shutdown\\_](#) Command class

Inheritance diagram for `artdaq::shutdown_`:



### Public Member Functions

- [shutdown\\_ \(xmlrpc\\_commander &c\)](#)  
[shutdown\\_ Constructor](#)

### Static Public Attributes

- static const uint64\_t [defaultTimeout](#) = 45

## Additional Inherited Members

### 6.206.1 Detailed Description

[shutdown\\_](#) Command class

Definition at line 931 of file xmlrpc\_commander.cc.

### 6.206.2 Constructor & Destructor Documentation

6.206.2.1 `artdaq::shutdown_::shutdown_( xmlrpc_commander & c )` `[inline]`

[shutdown\\_](#) Constructor

Parameters

<code>c</code>	<a href="#">xmlrpc_commander</a> to send transition commands to
----------------	---

Definition at line 938 of file xmlrpc\_commander.cc.

### 6.206.3 Member Data Documentation

6.206.3.1 `const uint64_t artdaq::shutdown_::defaultTimeout = 45` `[static]`

Default timeout for command

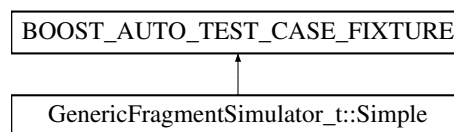
Definition at line 943 of file xmlrpc\_commander.cc.

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

- artdaq/artdaq/ExternalComms/xmlrpc\_commander.cc

## 6.207 GenericFragmentSimulator\_t::Simple Struct Reference

Inheritance diagram for GenericFragmentSimulator\_t::Simple:



## Public Member Functions

- void **test\_method** ()

### 6.207.1 Detailed Description

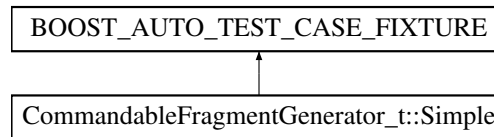
Definition at line 18 of file GenericFragmentSimulator\_t.cc.

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

- `artdaq/test/DAQdata/GenericFragmentSimulator_t.cc`

## 6.208 CommandableFragmentGenerator\_t::Simple Struct Reference

Inheritance diagram for CommandableFragmentGenerator\_t::Simple:



### Public Member Functions

- `void test_method ()`

#### 6.208.1 Detailed Description

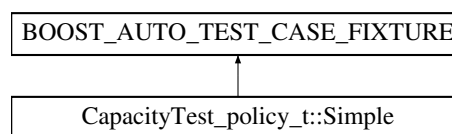
Definition at line 190 of file `CommandableFragmentGenerator_t.cc`.

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

- `artdaq/test/Generators/CommandableFragmentGenerator_t.cc`

## 6.209 CapacityTest\_policy\_t::Simple Struct Reference

Inheritance diagram for CapacityTest\_policy\_t::Simple:



### Public Member Functions

- `void test_method ()`

#### 6.209.1 Detailed Description

Definition at line 12 of file `CapacityTest_policy_t.cc`.

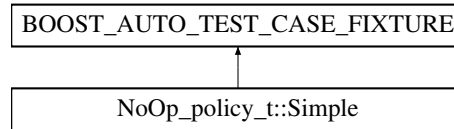
The documentation for this struct was generated from the following file:

- `artdaq/test/RoutingPolicies/CapacityTest_policy_t.cc`



## 6.210 NoOp\_policy\_t::Simple Struct Reference

Inheritance diagram for NoOp\_policy\_t::Simple:



### Public Member Functions

- void **test\_method** ()

#### 6.210.1 Detailed Description

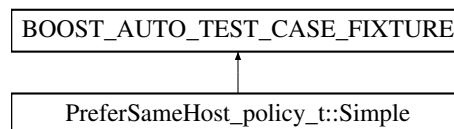
Definition at line 12 of file NoOp\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/NoOp\_policy\_t.cc

## 6.211 PreferSameHost\_policy\_t::Simple Struct Reference

Inheritance diagram for PreferSameHost\_policy\_t::Simple:



### Public Member Functions

- void **test\_method** ()

#### 6.211.1 Detailed Description

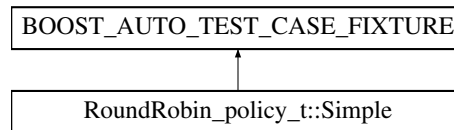
Definition at line 12 of file PreferSameHost\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/PreferSameHost\_policy\_t.cc

## 6.212 RoundRobin\_policy\_t::Simple Struct Reference

Inheritance diagram for RoundRobin\_policy\_t::Simple:



### Public Member Functions

- void **test\_method** ()

#### 6.212.1 Detailed Description

Definition at line 31 of file RoundRobin\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/RoundRobin\_policy\_t.cc

## 6.213 RoundRobin\_policy\_t::Simple\_id Struct Reference

#### 6.213.1 Detailed Description

Definition at line 31 of file RoundRobin\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/RoundRobin\_policy\_t.cc

## 6.214 CommandableFragmentGenerator\_t::Simple\_id Struct Reference

#### 6.214.1 Detailed Description

Definition at line 190 of file CommandableFragmentGenerator\_t.cc.

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

- artdaq/test/Generators/CommandableFragmentGenerator\_t.cc

## 6.215 GenericFragmentSimulator\_t::Simple\_id Struct Reference

#### 6.215.1 Detailed Description

Definition at line 18 of file GenericFragmentSimulator\_t.cc.

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

- artdaq/test/DAQdata/GenericFragmentSimulator\_t.cc

## 6.216 NoOp\_policy\_t::Simple\_id Struct Reference

### 6.216.1 Detailed Description

Definition at line 12 of file NoOp\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/NoOp\_policy\_t.cc

## 6.217 CapacityTest\_policy\_t::Simple\_id Struct Reference

### 6.217.1 Detailed Description

Definition at line 12 of file CapacityTest\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/CapacityTest\_policy\_t.cc

## 6.218 PreferSameHost\_policy\_t::Simple\_id Struct Reference

### 6.218.1 Detailed Description

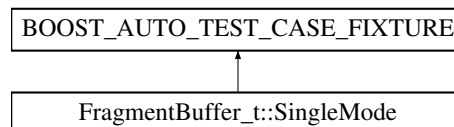
Definition at line 12 of file PreferSameHost\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/PreferSameHost\_policy\_t.cc

## 6.219 FragmentBuffer\_t::SingleMode Struct Reference

Inheritance diagram for FragmentBuffer\_t::SingleMode:



### Public Member Functions

- void **test\_method** ()

### 6.219.1 Detailed Description

Definition at line 165 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.220 FragmentBuffer\_t::SingleMode\_id Struct Reference

### 6.220.1 Detailed Description

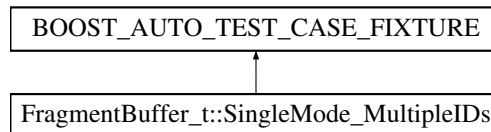
Definition at line 165 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.221 FragmentBuffer\_t::SingleMode\_MultipleIDs Struct Reference

Inheritance diagram for FragmentBuffer\_t::SingleMode\_MultipleIDs:



### Public Member Functions

- void **test\_method** ()

### 6.221.1 Detailed Description

Definition at line 1218 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.222 FragmentBuffer\_t::SingleMode\_MultipleIDs\_id Struct Reference

### 6.222.1 Detailed Description

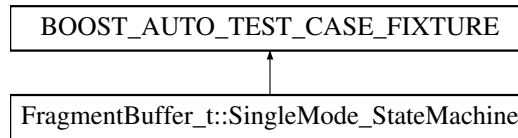
Definition at line 1218 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.223 FragmentBuffer\_t::SingleMode\_StateMachine Struct Reference

Inheritance diagram for FragmentBuffer\_t::SingleMode\_StateMachine:



### Public Member Functions

- void **test\_method** ()

#### 6.223.1 Detailed Description

Definition at line 2095 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.224 FragmentBuffer\_t::SingleMode\_StateMachine\_id Struct Reference

#### 6.224.1 Detailed Description

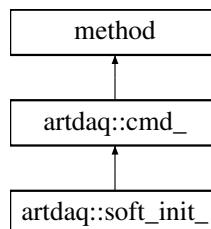
Definition at line 2095 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.225 artdaq::soft\_init\_ Class Reference

Inheritance diagram for artdaq::soft\_init\_:



### Public Member Functions

- [soft\\_init\\_](#) (xmlrpc\_commander &c)

## Static Public Attributes

- static const uint64\_t [defaultTimeout](#) = 45
- static const uint64\_t [defaultTimestamp](#) = std::numeric\_limits<const uint64\_t>::max()

## Additional Inherited Members

### 6.225.1 Detailed Description

Command class representing an init transition

Definition at line 843 of file xmlrpc\_commander.cc.

### 6.225.2 Constructor & Destructor Documentation

6.225.2.1 `artdaq::soft_init::soft_init( xmlrpc_commander & c ) [inline], [explicit]`

Command class Constructor \ \*

Parameters

<code>c</code>	<a href="#">xmlrpc_commander</a> to send parsed command to \
----------------	--

Definition at line 843 of file xmlrpc\_commander.cc.

### 6.225.3 Member Data Documentation

6.225.3.1 `const uint64_t artdaq::soft_init::defaultTimeout = 45 [static]`

Default timeout for command

Definition at line 843 of file xmlrpc\_commander.cc.

6.225.3.2 `const uint64_t artdaq::soft_init::defaultTimestamp = std::numeric_limits<const uint64_t>::max() [static]`

Default timestamp for Command

Definition at line 843 of file xmlrpc\_commander.cc.

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

- artdaq/artdaq/ExternalComms/xmlrpc\_commander.cc

## 6.226 `art::Source_generator< ArtdaqInputHelper< artdaq::ListenTransferWrapper > >` Struct Template Reference

Trait definition (must precede source typedef).

## Static Public Attributes

- static constexpr bool [value](#) = true

*Dummy variable.*

### 6.226.1 Detailed Description

```
template<>struct art::Source_generator< ArtdaqInputHelper< artdaq::ListenTransferWrapper > >
```

Trait definition (must precede source typedef).

Definition at line 12 of file TransferListenerInput\_source.cc.

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

- artdaq/artdaq/ArtModules/TransferListenerInput\_source.cc

## 6.227 art::Source\_generator< ArtdaqInputHelper< artdaq::TransferWrapper > > Struct Template Reference

Trait definition (must precede source typedef).

### Static Public Attributes

- static constexpr bool [value](#) = true

*Dummy variable.*

### 6.227.1 Detailed Description

```
template<>struct art::Source_generator< ArtdaqInputHelper< artdaq::TransferWrapper > >
```

Trait definition (must precede source typedef).

Definition at line 12 of file TransferInput\_source.cc.

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

- artdaq/artdaq/ArtModules/TransferInput\_source.cc

## 6.228 art::Source\_generator< ArtdaqInputHelper< ShmemWrapper > > Struct Template Reference

Trait definition (must precede source typedef).

### Static Public Attributes

- static constexpr bool [value](#) = true

*dummy parameter*

### 6.228.1 Detailed Description

```
template<> struct art::Source_generator< ArtdaqInputHelper< ShmemWrapper > >
```

Trait definition (must precede source typedef).

Definition at line 15 of file ArtdaqInput\_source.cc.

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

- artdaq/artdaq/ArtModules/ArtdaqInput\_source.cc

## 6.229 art::SourceConfig Struct Reference

Configuration for the source block of artdaq art processes

```
#include <artdaq/ArtModules/detail/ArtConfig.hh>
```

### Public Attributes

- fhicl::Atom< std::string > [module\\_type](#) {fhicl::Name{"module\_type"}, fhicl::Comment{"Module type of source. Should be \"ArtdaqInput\""}}  
*"module\_type": REQUIRED: Module type of source. Should be "ArtdaqInput"*
- fhicl::Atom< double > [init\\_fragment\\_timeout\\_seconds](#) {fhicl::Name{"init\_fragment\_timeout\_seconds"}, fhicl::Comment{"Amount of time [ArtdaqInput](#) should wait for an Init Fragment before simply returning Fragments"}, 600.0}  
*"init\_fragment\_timeout\_seconds" (Default: 600.0): Amount of time (in s) ArtdaqInput should wait for an Init Fragment before simply returning Fragments*
- fhicl::Atom< std::string > [raw\\_data\\_label](#) {fhicl::Name{"raw\_data\_label"}, fhicl::Comment{"Label to use for raw data (i.e. Fragments from the DAQ)"}, "daq"}  
*"raw\_data\_label" (Default: "daq"): Label to use for raw data (i.e. Fragments from the DAQ)*
- fhicl::Atom< bool > [register\\_fragment\\_types](#) {fhicl::Name{"register\_fragment\_types"}, fhicl::Comment{"Whether [ArtdaqInputHelper](#) should register the known Fragment types from the ArtdaqFragmentNamingServiceInterface. Required for EventBuilders. Disable for processes that don't handle Fragments."}, true}  
*"register\_fragment\_types" (Default: true): Whether [ArtdaqInputHelper](#) should register the known Fragment types from the [ArtdaqFragmentNamingServiceInterface](#). Required for EventBuilders. Disable for processes that don't handle Fragments.*

### 6.229.1 Detailed Description

Configuration for the source block of artdaq art processes

Definition at line 139 of file ArtConfig.hh.

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

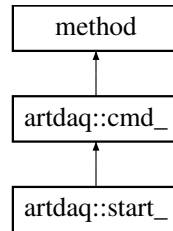
- artdaq/artdaq/ArtModules/detail/ArtConfig.hh

## 6.230 artdaq::start\_ Class Reference

Command class representing a start transition.



Inheritance diagram for artdaq::start\_:



## Public Member Functions

- [start\\_](#) ([xmlrpc\\_commander](#) &c)  
*[start\\_](#) Command ([cmd\\_](#) derived class) Constructor*

## Static Public Attributes

- static const uint64\_t [defaultTimeout](#) = 45
- static const uint64\_t [defaultTimestamp](#) = std::numeric\_limits<const uint64\_t>::max()

## Additional Inherited Members

### 6.230.1 Detailed Description

Command class representing a start transition.

Definition at line 854 of file xmlrpc\_commander.cc.

### 6.230.2 Constructor & Destructor Documentation

6.230.2.1 `artdaq::start_::start_ ( xmlrpc_commander & c )` `[inline]`, `[explicit]`

[start\\_](#) Command ([cmd\\_](#) derived class) Constructor

Parameters

<code>c</code>	<a href="#">xmlrpc_commander</a> instance to command
----------------	--

Definition at line 861 of file xmlrpc\_commander.cc.

### 6.230.3 Member Data Documentation

6.230.3.1 `const uint64_t artdaq::start_::defaultTimeout = 45` `[static]`

Default timeout for command

Definition at line 866 of file xmlrpc\_commander.cc.

6.230.3.2 `const uint64_t artdaq::start::defaultTimestamp = std::numeric_limits<const uint64_t>::max()` [static]

Default timestamp for Command

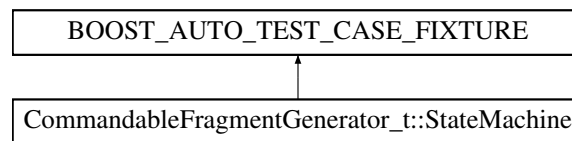
Definition at line 868 of file `xmlrpc_commander.cc`.

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

- `artdaq/artdaq/ExternalComms/xmlrpc_commander.cc`

## 6.231 CommandableFragmentGenerator\_t::StateMachine Struct Reference

Inheritance diagram for `CommandableFragmentGenerator_t::StateMachine`:



### Public Member Functions

- `void test_method ()`

### 6.231.1 Detailed Description

Definition at line 241 of file `CommandableFragmentGenerator_t.cc`.

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

- `artdaq/test/Generators/CommandableFragmentGenerator_t.cc`

## 6.232 CommandableFragmentGenerator\_t::StateMachine\_id Struct Reference

### 6.232.1 Detailed Description

Definition at line 241 of file `CommandableFragmentGenerator_t.cc`.

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

- `artdaq/test/Generators/CommandableFragmentGenerator_t.cc`

## 6.233 artdaq::StatisticsHelper Class Reference

This class manages `MonitoredQuantity` instances for the `*Core` classes.

```
#include <artdaq/DAQrate/StatisticsHelper.hh>
```

## Public Member Functions

- [StatisticsHelper](#) ()  
*StatisticsHelper default constructor.*
- [StatisticsHelper](#) ([StatisticsHelper](#) const &)=delete  
*Copy Constructor is deleted.*
- virtual [~StatisticsHelper](#) ()=default  
*Default Destructor.*
- [StatisticsHelper](#) & [operator=](#) ([StatisticsHelper](#) const &)=delete  
*Copy Assignment operator is deleted.*
- [StatisticsHelper](#) ([StatisticsHelper](#) &&)=delete  
*Move Constructor is deleted.*
- [StatisticsHelper](#) & [operator=](#) ([StatisticsHelper](#) &&)=delete  
*Move Assignment Operator is deleted.*
- void [addMonitoredQuantityName](#) (std::string const &statKey)  
*Add a MonitoredQuantity name to the list.*
- void [addSample](#) (std::string const &statKey, double value) const  
*Add a sample to the MonitoredQuantity with the given name.*
- bool [createCollectors](#) (fhicl::ParameterSet const &pset, int defaultReportIntervalFragments, double defaultReportIntervalSeconds, double defaultMonitorWindow, std::string const &primaryStatKeyName)  
*Create MonitoredQuantity objects for all names registered with the [StatisticsHelper](#).*
- void [resetStatistics](#) ()  
*Reset all MonitoredQuantity instances.*
- bool [readyToReport](#) ()  
*Determine if the reporting interval condition has been met.*
- bool [statsRollingWindowHasMoved](#) ()  
*Determine if the MonitoredQuantity "recent" window has changed since the last time this function was called.*

### 6.233.1 Detailed Description

This class manages MonitoredQuantity instances for the \*Core classes.

Definition at line 22 of file StatisticsHelper.hh.

### 6.233.2 Member Function Documentation

#### 6.233.2.1 void artdaq::StatisticsHelper::addMonitoredQuantityName ( std::string const & statKey )

Add a MonitoredQuantity name to the list.

Parameters

<i>statKey</i>	Name of the MonitoredQuantity to be added
----------------	---

Definition at line 16 of file StatisticsHelper.cc.

#### 6.233.2.2 void artdaq::StatisticsHelper::addSample ( std::string const & statKey, double value ) const

Add a sample to the MonitoredQuantity with the given name.

## Parameters

<i>statKey</i>	Name of the MonitoredQuantity
<i>value</i>	Value to record in the MonitoredQuantity

Definition at line 21 of file StatisticsHelper.cc.

6.233.2.3 `bool artdaq::StatisticsHelper::createCollectors ( fhicl::ParameterSet const & pset, int defaultReportIntervalFragments, double defaultReportIntervalSeconds, double defaultMonitorWindow, std::string const & primaryStatKeyName )`

Create MonitoredQuantity objects for all names registered with the [StatisticsHelper](#).

## Parameters

<i>pset</i>	ParameterSet used to configure reporting
<i>defaultReportIntervalFragments</i>	Default reporting interval in Fragments
<i>defaultReportIntervalSeconds</i>	Default reporting interval in Seconds
<i>defaultMonitorWindow</i>	Default monitoring window
<i>primaryStatKeyName</i>	The primary (default) MonitoredQuantity

## Returns

Whether the primary MonitoredQuantity exists

StatisticsHelper accepts the following Parameters: "reporting\_interval\_fragments" (Default given above): The reporting interval in Fragments "reporting\_interval\_seconds" (Default given above): The reporting interval in Seconds "monitor\_window" (Default given above): The monitoring window for the MonitoredQuantity "monitor\_binsize" (Default:  $1 + ((\text{monitorWindow} - 1) / 100)$ ): The monitoring bin size for the MonitoredQuantity

Definition at line 30 of file StatisticsHelper.cc.

6.233.2.4 `StatisticsHelper& artdaq::StatisticsHelper::operator= ( StatisticsHelper const & )` [delete]

Copy Assignment operator is deleted.

## Returns

[StatisticsHelper](#) copy

6.233.2.5 `bool artdaq::StatisticsHelper::readyToReport ( )`

Determine if the reporting interval condition has been met.

## Returns

Whether the [StatisticsHelper](#) is ready to report

Definition at line 72 of file StatisticsHelper.cc.

## 6.233.2.6 bool artdaq::StatisticsHelper::statsRollingWindowHasMoved ( )

Determine if the MonitoredQuantity "recent" window has changed since the last time this function was called.

## Returns

Whether the MonitoredQuantity "recent" window has changed

Definition at line 88 of file StatisticsHelper.cc.

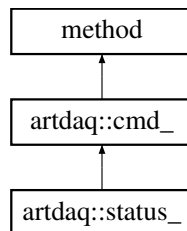
The documentation for this class was generated from the following files:

- artdaq/artdaq/DAQrate/StatisticsHelper.hh
- artdaq/artdaq/DAQrate/StatisticsHelper.cc

## 6.234 artdaq::status\_ Class Reference

[status\\_](#) Command class

Inheritance diagram for artdaq::status\_:



## Public Member Functions

- [status\\_](#) (xmlrpc\_commander &c)  
*[status\\_](#) Constructor*

## Additional Inherited Members

## 6.234.1 Detailed Description

[status\\_](#) Command class

Definition at line 964 of file xmlrpc\_commander.cc.

## 6.234.2 Constructor &amp; Destructor Documentation

## 6.234.2.1 artdaq::status\_::status\_ ( xmlrpc\_commander &amp; c ) [inline]

[status\\_](#) Constructor

## Parameters

<code>c</code>	<a href="#">xmlrpc_commander</a> to send transition commands to
----------------	---

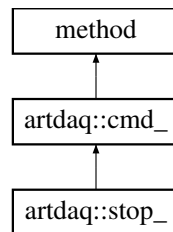
Definition at line 971 of file `xmlrpc_commander.cc`.

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

- `artdaq/artdaq/ExternalComms/xmlrpc_commander.cc`

## 6.235 artdaq::stop\_ Class Reference

Inheritance diagram for `artdaq::stop_`:



### Public Member Functions

- [stop\\_](#) ([xmlrpc\\_commander](#) &c)

### Static Public Attributes

- static const uint64\_t [defaultTimeout](#) = 45
- static const uint64\_t [defaultTimestamp](#) = std::numeric\_limits<const uint64\_t>::max()

### Additional Inherited Members

#### 6.235.1 Detailed Description

[stop\\_](#) Command class

Definition at line 924 of file `xmlrpc_commander.cc`.

#### 6.235.2 Constructor & Destructor Documentation

6.235.2.1 `artdaq::stop_::stop_ ( xmlrpc_commander &c ) [inline]`

[stop\\_](#) Constructor \

## Parameters

<code>c</code>	<code>xmlrpc_commander</code> to send transition commands to
----------------	--

Definition at line 924 of file `xmlrpc_commander.cc`.

### 6.235.3 Member Data Documentation

6.235.3.1 `const uint64_t artdaq::stop::defaultTimeout = 45` `[static]`

Default timeout for command

Definition at line 924 of file `xmlrpc_commander.cc`.

6.235.3.2 `const uint64_t artdaq::stop::defaultTimestamp = std::numeric_limits<const uint64_t>::max()` `[static]`

Default timestamp for Command

Definition at line 924 of file `xmlrpc_commander.cc`.

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

- `artdaq/artdaq/ExternalComms/xmlrpc_commander.cc`

## 6.236 swig\_artdaq Class Reference

Simple class exposing methods for TRACEing and sending metrics that can be wrapped with SWIG.

```
#include <tools/swig_artdaq.h>
```

### Public Member Functions

- `swig_artdaq` (`std::string const &config_string`)  
*swig\_artdaq constructor*
- `~swig_artdaq` ()  
*swig\_artdaq destructor*
- void `send_metric` (`std::string const &name`, `int level`, `std::string const &value`, `std::string const &unit`)  
*Send a metric using the artdaq MetricManager to the configured backends. Only the last point in a reporting\_interval will be sent to the backend.*
- void `send_metric` (`std::string const &name`, `int level`, `int value`, `std::string const &unit`)  
*Send a metric using the artdaq MetricManager to the configured backends. Only the last point in a reporting\_interval will be sent to the backend.*
- void `send_metric` (`std::string const &name`, `int level`, `double value`, `std::string const &unit`)  
*Send a metric using the artdaq MetricManager to the configured backends. Only the last point in a reporting\_interval will be sent to the backend.*
- void `send_sum_metric` (`std::string const &name`, `int level`, `std::string const &value`, `std::string const &unit`)  
*Send a metric using the artdaq MetricManager to the configured backends. Metric instances will be summed for the configured reporting\_intervals.*
- void `send_sum_metric` (`std::string const &name`, `int level`, `int value`, `std::string const &unit`)  
*Send a metric using the artdaq MetricManager to the configured backends. Metric instances will be summed for the configured reporting\_intervals.*

- void [send\\_sum\\_metric](#) (std::string const &name, int level, double value, std::string const &unit)  
*Send a metric using the artdaq MetricManager to the configured backends. Metric instances will be summed for the configured reporting\_intervals.*
- void [send\\_rate\\_metric](#) (std::string const &name, int level, std::string const &value, std::string const &unit)  
*Send a metric using the artdaq MetricManager to the configured backends. Metric instances will be summed for the configured reporting\_intervals, and the result will be reported as a rate (sum / reporting interval).*
- void [send\\_rate\\_metric](#) (std::string const &name, int level, int value, std::string const &unit)  
*Send a metric using the artdaq MetricManager to the configured backends. Metric instances will be summed for the configured reporting\_intervals, and the result will be reported as a rate (sum / reporting interval).*
- void [send\\_rate\\_metric](#) (std::string const &name, int level, double value, std::string const &unit)  
*Send a metric using the artdaq MetricManager to the configured backends. Metric instances will be summed for the configured reporting\_intervals, and the result will be reported as a rate (sum / reporting interval).*
- void [write\\_error](#) (std::string const &name, std::string const &message)  
*Write an error message to TRACE, using the TLVL\_ERROR level.*
- void [write\\_warning](#) (std::string const &name, std::string const &message)  
*Write an error message to TRACE, using the TLVL\_WARNING level.*
- void [write\\_info](#) (std::string const &name, std::string const &message)  
*Write an error message to TRACE, using the TLVL\_INFO level.*
- void [write\\_debug](#) (std::string const &name, std::string const &message)  
*Write an error message to TRACE, using the TLVL\_DEBUG level.*
- void [write\\_trace](#) (int level, std::string const &name, std::string const &message)  
*Write an error message to TRACE, using the provided level.*

## Public Attributes

- bool [initialized\\_](#)

### 6.236.1 Detailed Description

Simple class exposing methods for TRACEing and sending metrics that can be wrapped with SWIG.

Definition at line 13 of file swig\_artdaq.h.

### 6.236.2 Constructor & Destructor Documentation

6.236.2.1 [swig\\_artdaq::swig\\_artdaq](#) ( std::string const & *config\_string* ) `[explicit]`

[swig\\_artdaq](#) constructor

Parameters

<i>config_string</i>	FHiCL-formatted configuration, including application_name ("external"), id (0), and metrics ({}).
----------------------	---

Definition at line 7 of file swig\_artdaq.cc.

### 6.236.3 Member Function Documentation

6.236.3.1 void [swig\\_artdaq::send\\_metric](#) ( std::string const & *name*, int *level*, std::string const & *value*, std::string const & *unit* )

Send a metric using the artdaq MetricManager to the configured backends. Only the last point in a reporting\_interval will be sent to the backend.



## Parameters

<i>name</i>	Name of the metric
<i>level</i>	Level to report at
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric, should be consistent across all calls of the metric with this name

Definition at line 23 of file swig\_artdaq.cc.

### 6.236.3.2 void swig\_artdaq::send\_metric ( std::string const & *name*, int *level*, int *value*, std::string const & *unit* )

Send a metric using the artdaq MetricManager to the configured backends. Only the last point in a reporting\_interval will be sent to the backend.

## Parameters

<i>name</i>	Name of the metric
<i>level</i>	Level to report at
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric, should be consistent across all calls of the metric with this name

Definition at line 29 of file swig\_artdaq.cc.

### 6.236.3.3 void swig\_artdaq::send\_metric ( std::string const & *name*, int *level*, double *value*, std::string const & *unit* )

Send a metric using the artdaq MetricManager to the configured backends. Only the last point in a reporting\_interval will be sent to the backend.

## Parameters

<i>name</i>	Name of the metric
<i>level</i>	Level to report at
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric, should be consistent across all calls of the metric with this name

Definition at line 35 of file swig\_artdaq.cc.

### 6.236.3.4 void swig\_artdaq::send\_rate\_metric ( std::string const & *name*, int *level*, std::string const & *value*, std::string const & *unit* )

Send a metric using the artdaq MetricManager to the configured backends. Metric instances will be summed for the configured reporting\_intervals, and the result will be reported as a rate (sum / reporting interval).

## Parameters

<i>name</i>	Name of the metric
<i>level</i>	Level to report at
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric, should be consistent across all calls of the metric with this name

Definition at line 57 of file swig\_artdaq.cc.

6.236.3.5 void swig\_artdaq::send\_rate\_metric ( std::string const & *name*, int *level*, int *value*, std::string const & *unit* )

Send a metric using the artdaq MetricManager to the configured backends. Metric instances will be summed for the configured reporting\_intervals, and the result will be reported as a rate (sum / reporting interval).

## Parameters

<i>name</i>	Name of the metric
<i>level</i>	Level to report at
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric, should be consistent across all calls of the metric with this name

Definition at line 62 of file swig\_artdaq.cc.

#### 6.236.3.6 void swig\_artdaq::send\_rate\_metric ( std::string const & *name*, int *level*, double *value*, std::string const & *unit* )

Send a metric using the artdaq MetricManager to the configured backends. Metric instances will be summed for the configured reporting\_intervals, and the result will be reported as a rate (sum / reporting interval).

## Parameters

<i>name</i>	Name of the metric
<i>level</i>	Level to report at
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric, should be consistent across all calls of the metric with this name

Definition at line 67 of file swig\_artdaq.cc.

#### 6.236.3.7 void swig\_artdaq::send\_sum\_metric ( std::string const & *name*, int *level*, std::string const & *value*, std::string const & *unit* )

Send a metric using the artdaq MetricManager to the configured backends. Metric instances will be summed for the configured reporting\_intervals.

## Parameters

<i>name</i>	Name of the metric
<i>level</i>	Level to report at
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric, should be consistent across all calls of the metric with this name

Definition at line 41 of file swig\_artdaq.cc.

#### 6.236.3.8 void swig\_artdaq::send\_sum\_metric ( std::string const & *name*, int *level*, int *value*, std::string const & *unit* )

Send a metric using the artdaq MetricManager to the configured backends. Metric instances will be summed for the configured reporting\_intervals.

## Parameters

<i>name</i>	Name of the metric
<i>level</i>	Level to report at
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric, should be consistent across all calls of the metric with this name

Definition at line 46 of file swig\_artdaq.cc.

6.236.3.9 void swig\_artdaq::send\_sum\_metric ( std::string const & *name*, int *level*, double *value*, std::string const & *unit* )

Send a metric using the artdaq MetricManager to the configured backends. Metric instances will be summed for the configured reporting\_intervals.

## Parameters

<i>name</i>	Name of the metric
<i>level</i>	Level to report at
<i>value</i>	Value of the metric
<i>unit</i>	Units of the metric, should be consistent across all calls of the metric with this name

Definition at line 51 of file swig\_artdaq.cc.

**6.236.3.10 void swig\_artdaq::write\_debug ( std::string const & *name*, std::string const & *message* )**

Write an error message to TRACE, using the TLVL\_DEBUG level.

## Parameters

<i>name</i>	TRACE name to write to
<i>message</i>	Message to write

Definition at line 91 of file swig\_artdaq.cc.

**6.236.3.11 void swig\_artdaq::write\_error ( std::string const & *name*, std::string const & *message* )**

Write an error message to TRACE, using the TLVL\_ERROR level.

## Parameters

<i>name</i>	TRACE name to write to
<i>message</i>	Message to write

Definition at line 73 of file swig\_artdaq.cc.

**6.236.3.12 void swig\_artdaq::write\_info ( std::string const & *name*, std::string const & *message* )**

Write an error message to TRACE, using the TLVL\_INFO level.

## Parameters

<i>name</i>	TRACE name to write to
<i>message</i>	Message to write

Definition at line 85 of file swig\_artdaq.cc.

**6.236.3.13 void swig\_artdaq::write\_trace ( int *level*, std::string const & *name*, std::string const & *message* )**

Write an error message to TRACE, using the provided level.

## Parameters

<i>level</i>	TRACE level to write at
<i>name</i>	TRACE name to write to
<i>message</i>	Message to write

Definition at line 97 of file swig\_artdaq.cc.

6.236.3.14 `void swig_artdaq::write_warning ( std::string const & name, std::string const & message )`

Write an error message to TRACE, using the TLVL\_WARNING level.

## Parameters

<i>name</i>	TRACE name to write to
<i>message</i>	Message to write

Definition at line 79 of file swig\_artdaq.cc.

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

- artdaq/tools/swig\_artdaq.h
- artdaq/tools/swig\_artdaq.cc

## 6.237 artdaq::TableReceiver Class Reference

Sends Fragment objects using [TransferInterface](#) plugins. Uses Routing Tables if configured, otherwise will Round-Robin Fragments to the destinations.

```
#include <artdaq/DAQrate/detail/TableReceiver.hh>
```

## Classes

- struct [Config](#)  
*Configuration for Routing table reception*

## Public Types

- using [Parameters](#) = fhicl::WrappedTable< [Config](#) >  
*Used for ParameterSet validation (if desired)*
- using [RoutingTable](#) = std::map< artdaq::Fragment::sequence\_id\_t, int >  
*Internal representation of a routing table, relating a sequence ID to a destination rank.*

## Public Member Functions

- [TableReceiver](#) (const fhicl::ParameterSet &ps)  
*TableReceiver Constructor.*
- virtual [~TableReceiver](#) ()  
*TableReceiver Destructor.*
- [RoutingTable GetRoutingTable](#) () const  
*Get a copy of the current RoutingTable.*
- [RoutingTable GetAndClearRoutingTable](#) ()  
*Get the current RoutingTable and remove all entries.*
- int [GetRoutingTableEntry](#) (artdaq::Fragment::sequence\_id\_t seqID)  
*Get the destination rank for the given sequence ID.*
- size\_t [GetRoutingTableEntryCount](#) () const  
*Gets the current size of the Routing Table, in case other parts of the system want to use this information.*
- size\_t [GetRemainingRoutingTableEntries](#) () const  
*Gets the number of sends remaining in the routing table, in case other parts of the system want to use this information.*
- void [StopTableReceiver](#) ()

Stop the [TableReceiver](#).

- void [RemoveRoutingTableEntry](#) (Fragment::sequence\_id\_t seq)  
Remove the given sequence ID from the routing table and sent\_count lists.
- void [SendMetrics](#) () const  
Report metrics to MetricManager.
- bool [RoutingManagerEnabled](#) () const  
Whether the [TableReceiver](#) will receive tables from the RoutingManager.

## Static Public Attributes

- static constexpr int [ROUTING\\_FAILED](#) = -1111  
Value used to indicate that a route was not properly generated.

### 6.237.1 Detailed Description

Sends Fragment objects using [TransferInterface](#) plugins. Uses Routing Tables if configured, otherwise will Round-Robin Fragments to the destinations.

Definition at line 34 of file TableReceiver.hh.

### 6.237.2 Constructor & Destructor Documentation

6.237.2.1 `artdaq::TableReceiver::TableReceiver ( const fhicl::ParameterSet & ps ) [explicit]`

[TableReceiver](#) Constructor.

Parameters

<i>ps</i>	ParameterSet used to configure the <a href="#">TableReceiver</a> . See <a href="#">artdaq::TableReceiver::Config</a>
-----------	--

Definition at line 16 of file TableReceiver.cc.

### 6.237.3 Member Function Documentation

6.237.3.1 `size_t artdaq::TableReceiver::GetRemainingRoutingTableEntries ( ) const`

Gets the number of sends remaining in the routing table, in case other parts of the system want to use this information.

Returns

The number of sends remaining in the routing table

Definition at line 334 of file TableReceiver.cc.

6.237.3.2 `int artdaq::TableReceiver::GetRoutingTableEntry ( artdaq::Fragment::sequence_id_t seqID )`

Get the destination rank for the given sequence ID.



## Parameters

<i>seqID</i>	Sequence ID to query
--------------	----------------------

## Returns

Destination rank for given Sequence ID

Definition at line 74 of file TableReceiver.cc.

6.237.3.3 `size_t artdaq::TableReceiver::GetRoutingTableEntryCount ( ) const`

Gets the current size of the Routing Table, in case other parts of the system want to use this information.

## Returns

The current size of the Routing Table.

Definition at line 328 of file TableReceiver.cc.

6.237.3.4 `void artdaq::TableReceiver::RemoveRoutingTableEntry ( Fragment::sequence_id_t seq )`

Remove the given sequence ID from the routing table and sent\_count lists.

## Parameters

<i>seq</i>	Sequence ID to remove
------------	-----------------------

Definition at line 342 of file TableReceiver.cc.

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

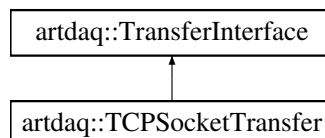
- artdaq/artdaq/DAQrate/detail/TableReceiver.hh
- artdaq/artdaq/DAQrate/detail/TableReceiver.cc

## 6.238 artdaq::TCPSocketTransfer Class Reference

[TransferInterface](#) implementation plugin that sends data using TCP sockets.

```
#include <artdaq/TransferPlugins/TCPSocketTransfer.hh>
```

Inheritance diagram for artdaq::TCPSocketTransfer:



### Public Member Functions

- [TCPSocketTransfer](#) (fhicl::ParameterSet const &ps, [Role](#) role)

[TCPSocketTransfer](#) Constructor.

- int [receiveFragmentHeader](#) (detail::RawFragmentHeader &header, size\_t timeout\_usec) override  
*Receive a Fragment Header from the transport mechanism.*
- int [receiveFragmentData](#) (RawDataType \*destination, size\_t wordCount) override  
*Receive the body of a Fragment to the given destination pointer.*
- [CopyStatus transfer\\_fragment\\_min\\_blocking\\_mode](#) (Fragment const &frag, size\_t timeout\_usec) override  
*Transfer a Fragment to the destination. May not necessarily be reliable, but will not block longer than send\_timeout\_usec.*
- [CopyStatus transfer\\_fragment\\_reliable\\_mode](#) (Fragment &&frag) override  
*Transfer a Fragment to the destination. This should be reliable, if the underlying transport mechanism supports reliable sending.*
- bool [isRunning](#) () override  
*Determine whether the [TransferInterface](#) plugin is able to send/receive data.*
- void [flush\\_buffers](#) () override  
*Flush any in-flight data. This should be used by the receiver after the receive loop has ended.*

## Additional Inherited Members

### 6.238.1 Detailed Description

[TransferInterface](#) implementation plugin that sends data using TCP sockets.

Definition at line 42 of file TCPSocketTransfer.hh.

### 6.238.2 Constructor & Destructor Documentation

#### 6.238.2.1 `artdaq::TCPSocketTransfer::TCPSocketTransfer ( fhicl::ParameterSet const & ps, TransferInterface::Role role )`

[TCPSocketTransfer](#) Constructor.

Parameters

<i>ps</i>	ParameterSet used to configure <a href="#">TCPSocketTransfer</a>
<i>role</i>	Role of this <a href="#">TCPSocketTransfer</a> instance (kSend or kReceive)

```
* TCPSocketTransfer accepts the following Parameters:
* "tcp_receive_buffer_size" (Default: 0): The TCP buffer size on the receive socket
* "send_retry_timeout_us" (Default: 1000000): Microseconds between send retries (infinite retries for moveFragments)
* "host_map" (REQUIRED): List of FHiCL tables containing information about other hosts in the system.
* Each table should contain:
*   "rank" (Default: RECV_TIMEOUT): Rank of this host
*   "host" (Default: "localhost"): Hostname of this host
*   "portOffset" (Default: 5500): To avoid collisions, each destination should specify its own port offset.
*   All TCPSocketTransfers sending to that destination will add their own rank to make a unique port number.
*
```

[TCPSocketTransfer](#) also requires all Parameters for configuring a [TransferInterface](#)

Definition at line 46 of file TCPSocketTransfer.cc.

### 6.238.3 Member Function Documentation

#### 6.238.3.1 `bool artdaq::TCPSocketTransfer::isRunning ( ) [override],[virtual]`

Determine whether the [TransferInterface](#) plugin is able to send/receive data.

### Returns

True if the [TransferInterface](#) plugin is currently able to send/receive data

Reimplemented from [artdaq::TransferInterface](#).

Definition at line 589 of file TCPSocketTransfer.cc.

**6.238.3.2** `int artdaq::TCPSocketTransfer::receiveFragmentData ( RawDataType * destination, size_t wordCount )`  
[[override](#)], [[virtual](#)]

Receive the body of a Fragment to the given destination pointer.

### Parameters

<i>destination</i>	Pointer to memory region where Fragment data should be stored
<i>wordCount</i>	Number of RawDataType words to receive

### Returns

The rank the Fragment was received from (should be `source_rank`), or `RECV_TIMEOUT`

Implements [artdaq::TransferInterface](#).

Definition at line 399 of file TCPSocketTransfer.cc.

**6.238.3.3** `int artdaq::TCPSocketTransfer::receiveFragmentHeader ( detail::RawFragmentHeader & header, size_t timeout_usec )`  
[[override](#)], [[virtual](#)]

Receive a Fragment Header from the transport mechanism.

### Parameters

<i>out</i>	<i>header</i>	Received Fragment Header
	<i>timeout_usec</i>	<a href="#">Timeout</a> for receive

### Returns

The rank the Fragment was received from (should be `source_rank`), or `RECV_TIMEOUT`

Implements [artdaq::TransferInterface](#).

Definition at line 130 of file TCPSocketTransfer.cc.

**6.238.3.4** `CopyStatus artdaq::TCPSocketTransfer::transfer_fragment_min_blocking_mode ( Fragment const & frag, size_t timeout_usec )` [[inline](#)], [[override](#)]

Transfer a Fragment to the destination. May not necessarily be reliable, but will not block longer than `send_timeout_usec`.

### Parameters

<i>frag</i>	Fragment to transfer
-------------	----------------------

<code>timeout_usec</code>	<a href="#">Timeout</a> for send, in microseconds
---------------------------	---

**Returns**

CopyStatus detailing result of transfer

Definition at line 89 of file TCPSocketTransfer.hh.

### 6.238.3.5 CopyStatus artdaq::TCPSocketTransfer::transfer\_fragment\_reliable\_mode ( Fragment && frag ) [inline], [override]

Transfer a Fragment to the destination. This should be reliable, if the underlying transport mechanism supports reliable sending.

**Parameters**

<code>frag</code>	Fragment to transfer
-------------------	----------------------

**Returns**

CopyStatus detailing result of copy

Definition at line 96 of file TCPSocketTransfer.hh.

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

- artdaq/artdaq/TransferPlugins/TCPSocketTransfer.hh
- artdaq/artdaq/TransferPlugins/TCPSocketTransfer.cc

## 6.239 anonymous\_namespace{genToArt.cc}::ThrottledGenerator Class Reference

[ThrottledGenerator](#): ensure that we only get one fragment per type at a time from the generator.

**Public Member Functions**

- [ThrottledGenerator](#) (std::string const &generator, fhicl::ParameterSet const &ps)  
*ThrottledGenerator Constructor.*
- bool [getNext](#) (artdaq::FragmentPtrs &newFrgs)  
*Get the next fragment from the generator.*
- size\_t [numFragIDs](#) () const  
*Get the number of Fragment IDs handled by this generator.*
- void [start](#) (int run, uint64\_t timeout, uint64\_t timestamp) const  
*Send start signal to FragmentGenerator, if it's a CommandableFragmentGenerator.*
- void [stop](#) (uint64\_t timeout, uint64\_t timestamp) const  
*Send stop signal to FragmentGenerator, if it's a CommandableFragmentGenerator.*

### 6.239.1 Detailed Description

[ThrottledGenerator](#): ensure that we only get one fragment per type at a time from the generator.

Definition at line 86 of file genToArt.cc.

## 6.239.2 Constructor & Destructor Documentation

6.239.2.1 anonymous\_namespace{genToArt.cc}::ThrottledGenerator::ThrottledGenerator ( std::string const & *generator*, fhicl::ParameterSet const & *ps* )

[ThrottledGenerator](#) Constructor.

Parameters

<i>generator</i>	Name of the generator plugin to load
<i>ps</i>	ParameterSet for configuring the FragmentGenerator

Definition at line 149 of file genToArt.cc.

## 6.239.3 Member Function Documentation

6.239.3.1 bool anonymous\_namespace{genToArt.cc}::ThrottledGenerator::getNext ( artdaq::FragmentPtrs & *newFrgs* )

Get the next fragment from the generator.

Parameters

<i>out</i>	<i>newFrgs</i>	New Fragment objects are added to this list
------------	----------------	---

Returns

Whether there is more data forthcoming

Definition at line 159 of file genToArt.cc.

6.239.3.2 size\_t anonymous\_namespace{genToArt.cc}::ThrottledGenerator::numFragIDs ( ) const

Get the number of Fragment IDs handled by this generator.

Returns

Definition at line 195 of file genToArt.cc.

6.239.3.3 void anonymous\_namespace{genToArt.cc}::ThrottledGenerator::start ( int *run*, uint64\_t *timeout*, uint64\_t *timestamp* ) const [\[inline\]](#)

Send start signal to FragmentGenerator, if it's a CommandableFragmentGenerator.

Parameters

<i>run</i>	Run number to pass to StartCmd
<i>timeout</i>	<a href="#">Timeout</a> to pass to StartCmd
<i>timestamp</i>	Timestamp to pass to StartCmd

Definition at line 116 of file genToArt.cc.

6.239.3.4 `void anonymous_namespace{genToArt.cc}::ThrottledGenerator::stop ( uint64_t timeout, uint64_t timestamp ) const`  
`[inline]`

Send stop signal to FragmentGenerator, if it's a CommandableFragmentGenerator.

## Parameters

<i>timeout</i>	<a href="#">Timeout</a> to pass to StopCmd
<i>timestamp</i>	Timestamp to pass to StopCmd

Definition at line 129 of file genToArt.cc.

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

- artdaq/tools/genToArt.cc

## 6.240 Timeout Class Reference

The [Timeout](#) class performs registered actions at specified intervals.

```
#include <artdaq/TransferPlugins/detail/Timeout.hh>
```

### Classes

- struct [timeoutspec](#)  
Specification for a [Timeout](#) function.

### Public Member Functions

- [Timeout](#) (int max\_tmos=100)  
Construct a [Timeout](#) object.
- void [add\\_periodic](#) (const char \*desc, void \*tag, std::function< void()> &function, uint64\_t period\_us, uint64\_t start\_us=0)  
Add a periodic timeout to the [Timeout](#) container.
- void [add\\_periodic](#) (const char \*desc, void \*tag, std::function< void()> &function, int rel\_ms)  
Add a periodic timeout to the [Timeout](#) container.
- void [add\\_periodic](#) (const char \*desc, uint64\_t period\_us, uint64\_t start\_us=0)  
Add a periodic timeout to the [Timeout](#) container.
- void [add\\_relative](#) (const char \*desc, void \*tag, std::function< void()> &function, int rel\_ms)  
Add a periodic timeout to the [Timeout](#) container.
- void [add\\_relative](#) (const std::string &desc, int rel\_ms)  
Add a periodic timeout to the [Timeout](#) container.
- void [copy\\_in\\_timeout](#) (const char \*desc, uint64\_t period\_us, uint64\_t start\_us=0)  
Add a timeout with the given parameters.
- bool [cancel\\_timeout](#) (void \*tag, const std::string &desc)  
Cancel the timeout having the given description and tag.
- int [get\\_next\\_expired\\_timeout](#) (std::string &desc, void \*\*tag, std::function< void()> &function, uint64\_t \*tmo\_tod\_us)  
Get a timeout that has expired.
- void [get\\_next\\_timeout\\_delay](#) (int64\_t \*delay\_us)  
Get the amount to wait for the next timeout to occur.
- int [get\\_next\\_timeout\\_msdlly](#) ()  
Get the amount to wait for the next timeout to occur.
- bool [is\\_consistent](#) ()

*Run a consistency check on all configured timeouts.*

- void `list_active_time` ()

*TRACE all active timeouts.*

### 6.240.1 Detailed Description

The `Timeout` class performs registered actions at specified intervals.

Definition at line 22 of file `Timeout.hh`.

### 6.240.2 Constructor & Destructor Documentation

#### 6.240.2.1 `Timeout::Timeout ( int max_tmos = 100 ) [explicit]`

Construct a `Timeout` object.

Parameters

<i>max_tmos</i>	Maximum number of registered <code>Timeout</code> functions
-----------------	---

Definition at line 59 of file `Timeout.cc`.

### 6.240.3 Member Function Documentation

#### 6.240.3.1 `void Timeout::add_periodic ( const char * desc, void * tag, std::function< void()> & function, uint64_t period_us, uint64_t start_us = 0 )`

Add a periodic timeout to the `Timeout` container.

Parameters

<i>desc</i>	Description of the periodic timeout
<i>tag</i>	Tag (fd or other) to apply to timeout
<i>function</i>	Function to execute at timeout
<i>period_us</i>	Period for timeouts
<i>start_us</i>	When to start (defaults to 0)

maybe need to return a timeout id??

Definition at line 66 of file `Timeout.cc`.

#### 6.240.3.2 `void Timeout::add_periodic ( const char * desc, void * tag, std::function< void()> & function, int rel_ms )`

Add a periodic timeout to the `Timeout` container.

Parameters

<i>desc</i>	Description of the periodic timeout
<i>tag</i>	Tag (fd or other) to apply to timeout
<i>function</i>	Function to execute at timeout



<i>rel_ms</i>	<a href="#">Timeout</a> in rem_ms milliseconds
---------------	--

maybe need to return a timeout id??

Definition at line 78 of file Timeout.cc.

6.240.3.3 void Timeout::add\_periodic ( const char \* *desc*, uint64\_t *period\_us*, uint64\_t *start\_us* = 0 )

Add a periodic timeout to the [Timeout](#) container.

Parameters

<i>desc</i>	Description of the periodic timeout
<i>period_us</i>	Period for timeouts
<i>start_us</i>	When to start (defaults to 0)

maybe need to return a timeout id??

Definition at line 90 of file Timeout.cc.

6.240.3.4 void Timeout::add\_relative ( const char \* *desc*, void \* *tag*, std::function< void()> & *function*, int *rel\_ms* )

Add a periodic timeout to the [Timeout](#) container.

Parameters

<i>desc</i>	Description of the periodic timeout
<i>tag</i>	Tag (fd or other) to apply to timeout
<i>function</i>	Function to execute at timeout
<i>rel_ms</i>	<a href="#">Timeout</a> in rem_ms milliseconds

maybe need to return a timeout id??

Definition at line 103 of file Timeout.cc.

6.240.3.5 void Timeout::add\_relative ( const std::string & *desc*, int *rel\_ms* )

Add a periodic timeout to the [Timeout](#) container.

Parameters

<i>desc</i>	Description of the periodic timeout
<i>rel_ms</i>	<a href="#">Timeout</a> in rem_ms milliseconds

maybe need to return a timeout id??

Definition at line 115 of file Timeout.cc.

6.240.3.6 bool Timeout::cancel\_timeout ( void \* *tag*, const std::string & *desc* )

Cancel the timeout having the given description and tag.

Parameters

<i>tag</i>	Tag of the cancelled timeout
<i>desc</i>	Description of the cancelled timeout

**Returns**

Whether a timeout was found and cancelled

Definition at line 315 of file Timeout.cc.

6.240.3.7 `void Timeout::copy_in_timeout ( const char * desc, uint64_t period_us, uint64_t start_us = 0 )`

Add a timeout with the given parameters.

**Parameters**

<i>desc</i>	Description of new timeout
<i>period_us</i>	Period that the timeout should execute with
<i>start_us</i>	When to start the timeout (default 0)

Definition at line 287 of file Timeout.cc.

6.240.3.8 `int Timeout::get_next_expired_timeout ( std::string & desc, void ** tag, std::function< void()> & function, uint64_t * tmo_tod_us )`

Get a timeout that has expired.

**Parameters**

out	<i>desc</i>	Description of timeout that expired
out	<i>tag</i>	Tag of timeout that expired
out	<i>function</i>	Function of timeout that expired
out	<i>tmo_tod_us</i>	When the timeout expired

**Returns**

-1 if no timeouts expired

Definition at line 128 of file Timeout.cc.

6.240.3.9 `void Timeout::get_next_timeout_delay ( int64_t * delay_us )`

Get the amount to wait for the next timeout to occur.

**Parameters**

out	<i>delay_us</i>	Microseconds until next timeout
-----	-----------------	---------------------------------

Definition at line 149 of file Timeout.cc.

6.240.3.10 `int Timeout::get_next_timeout_msdl ( )`

Get the amount to wait for the next timeout to occur.

**Returns**

Milliseconds until next timeout

Definition at line 171 of file Timeout.cc.

### 6.240.3.11 bool Timeout::is\_consistent ( )

Run a consistency check on all configured timeouts.

#### Returns

True if all timeouts pass check

Definition at line 187 of file Timeout.cc.

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

- artdaq/artdaq/TransferPlugins/detail/Timeout.hh
- artdaq/artdaq/TransferPlugins/detail/Timeout.cc

## 6.241 Timeout::timeoutspec Struct Reference

Specification for a [Timeout](#) function.

```
#include <artdaq/TransferPlugins/detail/Timeout.hh>
```

### Public Attributes

- std::string [desc](#)  
*Description of the [Timeout](#) function.*
- void \* [tag](#)  
*could be file descriptor (fd)*
- std::function< void()> [function](#)  
*Function to execute at [Timeout](#).*
- uint64\_t [tmo\\_tod\\_us](#)  
*When the function should be executed (gettimeofday, microseconds)*
- uint64\_t [period\\_us](#)  
*0 if not periodic*
- int [missed\\_periods](#)  
*Number of periods that passed while the function was executing.*
- int [check](#)  
*Check the timeoutspec.*

### 6.241.1 Detailed Description

Specification for a [Timeout](#) function.

Definition at line 28 of file Timeout.hh.

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

- artdaq/artdaq/TransferPlugins/detail/Timeout.hh

## 6.242 artdaq::TokenReceiver Class Reference

Receives event builder "free buffer" tokens and adds them to a specified RoutingPolicy.

```
#include <artdaq/DAQrate/detail/TokenReceiver.hh>
```

### Classes

- struct [Config](#)

*Configuration of the [TokenReceiver](#). May be used for parameter validation.*

### Public Types

- using [Parameters](#) = fhicl::WrappedTable< [Config](#) >

*Used for ParameterSet validation (if desired)*

### Public Member Functions

- [TokenReceiver](#) (const fhicl::ParameterSet &ps, std::shared\_ptr< [RoutingManagerPolicy](#) > policy, size\_t update\_interval\_msec)

*[TokenReceiver](#) Constructor.*

- virtual [~TokenReceiver](#) ()

*[TokenReceiver](#) Destructor.*

- void [startTokenReception](#) ()

*Starts the reception of event builder tokens.*

- void [pauseTokenReception](#) ()

*Temporarily suspends the reception of event builder tokens.*

- void [resumeTokenReception](#) ()

*Resumes the reception of event builder tokens after a suspension.*

- void [stopTokenReception](#) (bool force=false)

*Stops the reception of event builder tokens.*

- void [setStatsHelper](#) (std::shared\_ptr< [StatisticsHelper](#) > const &helper, std::string const &stat\_key)

*Specifies a [StatisticsHelper](#) instance to use when gathering statistics.*

- void [setRunNumber](#) (uint32\_t run)

*Sets the current run number.*

- size\_t [getReceivedTokenCount](#) () const

*Returns the number of tokens that have been received.*

### 6.242.1 Detailed Description

Receives event builder "free buffer" tokens and adds them to a specified RoutingPolicy.

Definition at line 25 of file TokenReceiver.hh.

## 6.242.2 Constructor & Destructor Documentation

6.242.2.1 artdaq::TokenReceiver::TokenReceiver ( const fhicl::ParameterSet & *ps*, std::shared\_ptr< RoutingManagerPolicy > *policy*, size\_t *update\_interval\_msec* ) [explicit]

[TokenReceiver](#) Constructor.

## Parameters

<i>ps</i>	ParameterSet used to configure <a href="#">TokenReceiver</a> . See <a href="#">artdaq::TokenReceiver::Config</a>
<i>policy</i>	<a href="#">RoutingManagerPolicy</a> that manages the received tokens
<i>update_interval_ - msec</i>	The amount of time to wait in <code>epoll_wait</code> for a new update to arrive

Definition at line 12 of file `TokenReceiver.cc`.

### 6.242.3 Member Function Documentation

6.242.3.1 `size_t artdaq::TokenReceiver::getReceivedTokenCount ( ) const` `[inline]`

Returns the number of tokens that have been received.

## Returns

The number of tokens that have been received since the most recent start

Definition at line 95 of file `TokenReceiver.hh`.

6.242.3.2 `void artdaq::TokenReceiver::setRunNumber ( uint32_t run )` `[inline]`

Sets the current run number.

## Parameters

<i>run</i>	The current run number
------------	------------------------

Definition at line 89 of file `TokenReceiver.hh`.

6.242.3.3 `void artdaq::TokenReceiver::setStatsHelper ( std::shared_ptr< StatisticsHelper > const & helper, std::string const & stat_key )` `[inline]`

Specifies a [StatisticsHelper](#) instance to use when gathering statistics.

## Parameters

<i>helper</i>	A shared pointer to the <a href="#">StatisticsHelper</a> instance
<i>stat_key</i>	Name to use for gathering statistics on tokens received

Definition at line 79 of file `TokenReceiver.hh`.

6.242.3.4 `void artdaq::TokenReceiver::stopTokenReception ( bool force = false )`

Stops the reception of event builder tokens.

## Parameters

<i>force</i>	Whether to suppress any error messages (used if called from destructor)
--------------	---

Definition at line 66 of file `TokenReceiver.cc`.

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

- `artdaq/artdaq/DAQrate/detail/TokenReceiver.hh`
- `artdaq/artdaq/DAQrate/detail/TokenReceiver.cc`

## 6.243 artdaq::TokenSender Class Reference

The [TokenSender](#) contains methods used to send data requests and Routing tokens.

```
#include <artdaq/DAQrate/detail/TokenSender.hh>
```

### Classes

- struct [Config](#)  
*Configuration for Routing token sending*

### Public Types

- using [Parameters](#) = fhicl::WrappedTable< [Config](#) >  
*Used for ParameterSet validation (if desired)*

### Public Member Functions

- [TokenSender](#) ()=delete  
*Default Constructor is deleted.*
- [TokenSender](#) ([TokenSender](#) const &)=delete  
*Copy Constructor is deleted.*
- [TokenSender](#) & operator= ([TokenSender](#) const &)=delete  
*Copy Assignment operator is deleted.*
- [TokenSender](#) ([TokenSender](#) &&)=delete  
*Move Constructor is deleted.*
- [TokenSender](#) & operator= ([TokenSender](#) &&)=delete  
*Move-assignment operator is deleted.*
- [TokenSender](#) (const fhicl::ParameterSet &pset)  
*TokenSender Constructor.*
- virtual ~[TokenSender](#) ()  
*TokenSender Destructor.*
- void [SendRoutingToken](#) (int nSlots, int run\_number, int rank=my\_rank)  
*Send a RoutingToken message indicating that slots are available.*
- size\_t [GetSentTokenCount](#) () const  
*Get the count of number of tokens sent.*
- void [SetRunNumber](#) (uint32\_t run)  
*Set the run number to be used in request messages.*
- bool [RoutingTokenSendsEnabled](#) ()  
*Determine if routing token sends are enabled.*

#### 6.243.1 Detailed Description

The [TokenSender](#) contains methods used to send data requests and Routing tokens.

Definition at line 28 of file TokenSender.hh.

## 6.243.2 Constructor & Destructor Documentation

6.243.2.1 `artdaq::TokenSender::TokenSender ( const fhicl::ParameterSet & pset )` `[explicit]`

[TokenSender](#) Constructor.



## Parameters

<i>pset</i>	ParameterSet used to configured <a href="#">TokenSender</a> . See <a href="#">artdaq::TokenSender::Config</a>
-------------	---

Definition at line 18 of file TokenSender.cc.

### 6.243.3 Member Function Documentation

6.243.3.1 `size_t artdaq::TokenSender::GetSentTokenCount ( ) const` `[inline]`

Get the count of number of tokens sent.

## Returns

The number of tokens sent by [TokenSender](#)

Definition at line 89 of file TokenSender.hh.

6.243.3.2 `TokenSender& artdaq::TokenSender::operator= ( TokenSender const & )` `[delete]`

Copy Assignment operator is deleted.

## Returns

[TokenSender](#) copy

6.243.3.3 `bool artdaq::TokenSender::RoutingTokenSendsEnabled ( )` `[inline]`

Determine if routing token sends are enabled.

## Returns

If routing tokens will be sent by this [TokenSender](#)

Definition at line 101 of file TokenSender.hh.

6.243.3.4 `void artdaq::TokenSender::SendRoutingToken ( int nSlots, int run_number, int rank =my_rank )`

Send a RoutingToken message indicating that slots are available.

## Parameters

<i>nSlots</i>	Number of slots available
<i>run_number</i>	Run number for token
<i>rank</i>	Rank of token

Definition at line 113 of file TokenSender.cc.

6.243.3.5 `void artdaq::TokenSender::SetRunNumber ( uint32_t run )` `[inline]`

Set the run number to be used in request messages.

## Parameters

<i>run</i>	Run number
------------	------------

Definition at line 95 of file TokenSender.hh.

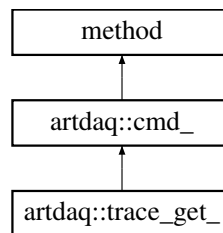
The documentation for this class was generated from the following files:

- artdaq/artdaq/DAQrate/detail/TokenSender.hh
- artdaq/artdaq/DAQrate/detail/TokenSender.cc

## 6.244 artdaq::trace\_get\_ Class Reference

[trace\\_get\\_](#) Command class

Inheritance diagram for artdaq::trace\_get\_:



### Public Member Functions

- [trace\\_get\\_](#) (xmlrpc\_commander &c)  
*trace\_msgfacility\_set\_ Constructor*

### Additional Inherited Members

#### 6.244.1 Detailed Description

[trace\\_get\\_](#) Command class

Definition at line 1149 of file xmlrpc\_commander.cc.

#### 6.244.2 Constructor & Destructor Documentation

6.244.2.1 artdaq::trace\_get\_::trace\_get\_ ( xmlrpc\_commander &c ) [inline]

*trace\_msgfacility\_set\_ Constructor*

## Parameters

<i>c</i>	<a href="#">xmlrpc_commander</a> to send transition commands to
----------	---

Definition at line 1156 of file xmlrpc\_commander.cc.

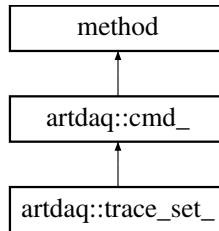
The documentation for this class was generated from the following file:

- artdaq/artdaq/ExternalComms/xmlrpc\_commander.cc

## 6.245 artdaq::trace\_set\_ Class Reference

[trace\\_set\\_](#) Command class

Inheritance diagram for artdaq::trace\_set\_:



### Public Member Functions

- [trace\\_set\\_](#) ([xmlrpc\\_commander](#) &c)  
[unregister\\_monitor\\_](#) Constructor

### Additional Inherited Members

#### 6.245.1 Detailed Description

[trace\\_set\\_](#) Command class

Definition at line 1116 of file xmlrpc\_commander.cc.

#### 6.245.2 Constructor & Destructor Documentation

6.245.2.1 artdaq::trace\_set\_::trace\_set\_ ( [xmlrpc\\_commander](#) & c ) [inline]

[unregister\\_monitor\\_](#) Constructor

Parameters

<a href="#">c</a>	<a href="#">xmlrpc_commander</a> to send transition commands to
-------------------	---

Definition at line 1123 of file xmlrpc\_commander.cc.

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

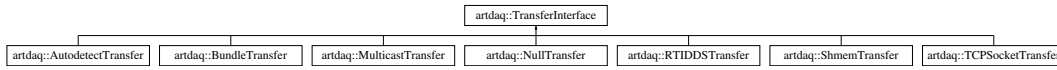
- artdaq/artdaq/ExternalComms/xmlrpc\_commander.cc

## 6.246 artdaq::TransferInterface Class Reference

This interface defines the functions used to transfer data between artdaq applications.

```
#include <artdaq/TransferPlugins/TransferInterface.hh>
```

Inheritance diagram for artdaq::TransferInterface:



## Classes

- struct [Config](#)

Configuration of the [TransferInterface](#). May be used for parameter validation

## Public Types

- enum [ReceiveReturnCode](#) : int { [DATA\\_END](#) = -2222, [RECV\\_TIMEOUT](#) = -1111, [NO\\_RANK\\_INFO](#) = -1, [RECV\\_SUCCESS](#) = 0 }
- Return codes from receive operations
- enum [Role](#) { [Role::kSend](#), [Role::kReceive](#) }
- Used to determine if a [TransferInterface](#) is a Sender or Receiver.
- enum [CopyStatus](#) { [CopyStatus::kSuccess](#), [CopyStatus::kTimeout](#), [CopyStatus::kErrorNotRequiringException](#) }
- Returned from the send functions, this enumeration describes the possible return codes. If an exception occurs, it will be thrown and should be handled normally.
- using [Parameters](#) = fhicl::WrappedTable< [Config](#) >
- Used for ParameterSet validation (if desired)

## Public Member Functions

- [TransferInterface](#) (const fhicl::ParameterSet &ps, [Role](#) role)
- [TransferInterface](#) Constructor.
- [TransferInterface](#) (const [TransferInterface](#) &)=delete
- Copy Constructor is deleted.
- [TransferInterface](#) & operator= (const [TransferInterface](#) &)=delete
- Copy Assignment operator is deleted.
- virtual ~[TransferInterface](#) ()=default
- Default virtual Destructor.
- virtual int [receiveFragment](#) (artdaq::Fragment &fragment, size\_t receive\_timeout)
- Receive a Fragment from the transport mechanism.
- virtual int [receiveFragmentHeader](#) (detail::RawFragmentHeader &header, size\_t receiveTimeout)=0
- Receive a Fragment Header from the transport mechanism.
- virtual int [receiveFragmentData](#) (RawDataType \*destination, size\_t wordCount)=0
- Receive the body of a Fragment to the given destination pointer.
- virtual [CopyStatus](#) [transfer\\_fragment\\_min\\_blocking\\_mode](#) (artdaq::Fragment const &fragment, size\_t send\_timeout\_usec)=0
- Transfer a Fragment to the destination. May not necessarily be reliable, but will not block longer than send\_timeout\_usec.
- virtual [CopyStatus](#) [transfer\\_fragment\\_reliable\\_mode](#) (artdaq::Fragment &&fragment)=0
- Transfer a Fragment to the destination. This should be reliable, if the underlying transport mechanism supports reliable sending.
- std::string [uniqueLabel](#) () const
- Get the unique label of this [TransferInterface](#) instance.
- virtual int [source\\_rank](#) () const

Get the source rank for this [TransferInterface](#) instance.

- virtual int [destination\\_rank](#) () const

Get the destination rank for this [TransferInterface](#) instance.

- virtual bool [isRunning](#) ()

Determine whether the [TransferInterface](#) plugin is able to send/receive data.

- virtual void [flush\\_buffers](#) ()=0

Flush any in-flight data. This should be used by the receiver after the receive loop has ended.

## Static Public Member Functions

- static std::string [CopyStatusToString](#) ([CopyStatus](#) in)

Convert a [CopyStatus](#) variable to its string representation

## Protected Member Functions

- [TransferInterface](#) ([TransferInterface](#) &&)=delete

Move Constructor is deleted.

- [TransferInterface](#) & [operator=](#) ([TransferInterface](#) &&)=delete

Move Assignment Operator is deleted.

- [Role](#) [role](#) () const

Get the [TransferInterface::Role](#) of this [TransferInterface](#).

## Protected Attributes

- const [Role](#) [role\\_](#)

Whether this instance of [TransferInterface](#) is a sender or receiver.

- const int [source\\_rank\\_](#)

Rank of source.

- const int [destination\\_rank\\_](#)

Rank of destination.

- const std::string [unique\\_label\\_](#)

Unique label of transfer (ideally the same on sender and receiver)

- size\_t [buffer\\_count\\_](#)

The number of Fragment transfers the [TransferInterface](#) can handle simultaneously.

- const size\_t [max\\_fragment\\_size\\_words\\_](#)

The maximum size of the transferred Fragment objects, in [artdaq::Fragment::RawDataType](#) words.

### 6.246.1 Detailed Description

This interface defines the functions used to transfer data between artdaq applications.

Definition at line 32 of file [TransferInterface.hh](#).

## 6.246.2 Member Enumeration Documentation

### 6.246.2.1 enum artdaq::TransferInterface::CopyStatus [strong]

Returned from the send functions, this enumeration describes the possible return codes. If an exception occurs, it will be thrown and should be handled normally.

#### Enumerator

**kSuccess** The send operation completed successfully.

**kTimeout** The send operation timed out.

**kErrorNotRequiringException** Some error occurred, but no exception was thrown.

Definition at line 78 of file TransferInterface.hh.

### 6.246.2.2 enum artdaq::TransferInterface::ReceiveReturnCode : int

Return codes from receive operations

#### Enumerator

**DATA\_END** Value that is to be returned when a Transfer plugin determines that no more data will be arriving.

**RECV\_TIMEOUT** Value to be returned upon receive timeout.

**NO\_RANK\_INFO** Will be returned from a successful receive that does not know the source rank (Transfer to OM art process)

**RECV\_SUCCESS** For code clarity, things checking for successful receive should check retval >= NO\_RANK\_INFO.

Definition at line 57 of file TransferInterface.hh.

### 6.246.2.3 enum artdaq::TransferInterface::Role [strong]

Used to determine if a [TransferInterface](#) is a Sender or Receiver.

#### Enumerator

**kSend** This [TransferInterface](#) is a Sender.

**kReceive** This [TransferInterface](#) is a Receiver.

Definition at line 68 of file TransferInterface.hh.

## 6.246.3 Constructor & Destructor Documentation

### 6.246.3.1 artdaq::TransferInterface::TransferInterface ( const fhicl::ParameterSet & ps, Role role )

[TransferInterface](#) Constructor.

## Parameters

<i>ps</i>	ParameterSet used for configuring the <a href="#">TransferInterface</a> . See <a href="#">artdaq::TransferInterface::Config</a>
<i>role</i>	Role of the <a href="#">TransferInterface</a> (See <a href="#">TransferInterface::Role</a> )

Definition at line 12 of file TransferInterface.cc.

## 6.246.4 Member Function Documentation

6.246.4.1 `static std::string artdaq::TransferInterface::CopyStatusToString ( CopyStatus in )` `[inline]`, `[static]`

Convert a CopyStatus variable to its string representation

## Parameters

<i>in</i>	CopyStatus to convert
-----------	-----------------------

## Returns

String representation of CopyStatus

Definition at line 90 of file TransferInterface.hh.

6.246.4.2 `virtual int artdaq::TransferInterface::destination_rank ( ) const` `[inline]`, `[virtual]`

Get the destination rank for this [TransferInterface](#) instance.

## Returns

The destination rank for this [TransferInterface](#) instance

Definition at line 186 of file TransferInterface.hh.

6.246.4.3 `virtual bool artdaq::TransferInterface::isRunning ( )` `[inline]`, `[virtual]`

Determine whether the [TransferInterface](#) plugin is able to send/receive data.

## Returns

True if the [TransferInterface](#) plugin is currently able to send/receive data

Reimplemented in [artdaq::BundleTransfer](#), [artdaq::TCPSocketTransfer](#), [artdaq::MulticastTransfer](#), [artdaq::AutodetectTransfer](#), [artdaq::ShmemTransfer](#), [artdaq::NullTransfer](#), and [artdaq::RTIDTransfer](#).

Definition at line 192 of file TransferInterface.hh.

6.246.4.4 `TransferInterface& artdaq::TransferInterface::operator= ( const TransferInterface & )` `[delete]`

Copy Assignment operator is deleted.

## Returns

[TransferInterface](#) Copy

6.246.4.5 `int artdaq::TransferInterface::receiveFragment ( artdaq::Fragment & fragment, size_t receive_timeout )` [virtual]

Receive a Fragment from the transport mechanism.



## Parameters

out	fragment	Received Fragment
	receive_timeout	Timeout for receive

## Returns

The rank the Fragment was received from (should be source\_rank), or RECV\_TIMEOUT

Reimplemented in [artdaq::MulticastTransfer](#), [artdaq::RTIDDSTransfer](#), [artdaq::BundleTransfer](#), [artdaq::AutodetectTransfer](#), and [artdaq::NullTransfer](#).

Definition at line 24 of file TransferInterface.cc.

**6.246.4.6** virtual int artdaq::TransferInterface::receiveFragmentData ( RawDataType \* destination, size\_t wordCount ) [pure virtual]

Receive the body of a Fragment to the given destination pointer.

## Parameters

destination	Pointer to memory region where Fragment data should be stored
wordCount	Number of words of Fragment data to receive

## Returns

The rank the Fragment was received from (should be source\_rank), or RECV\_TIMEOUT

The precondition for calling this function is that you have received a valid header, therefore it does not have a , as the Fragment data should immediately be available.

Implemented in [artdaq::BundleTransfer](#), [artdaq::TCPSocketTransfer](#), [artdaq::MulticastTransfer](#), [artdaq::AutodetectTransfer](#), [artdaq::ShmemTransfer](#), and [artdaq::NullTransfer](#).

**6.246.4.7** virtual int artdaq::TransferInterface::receiveFragmentHeader ( detail::RawFragmentHeader & header, size\_t receiveTimeout ) [pure virtual]

Receive a Fragment Header from the transport mechanism.

## Parameters

out	header	Received Fragment Header
	receiveTimeout	Timeout for receive

## Returns

The rank the Fragment was received from (should be source\_rank), or RECV\_TIMEOUT

Implemented in [artdaq::TCPSocketTransfer](#), [artdaq::MulticastTransfer](#), [artdaq::BundleTransfer](#), [artdaq::ShmemTransfer](#), [artdaq::AutodetectTransfer](#), and [artdaq::NullTransfer](#).

**6.246.4.8** Role artdaq::TransferInterface::role ( ) const [inline], [protected]

Get the [TransferInterface::Role](#) of this [TransferInterface](#).

**Returns**

The Role of this [TransferInterface](#)

Definition at line 221 of file TransferInterface.hh.

6.246.4.9 `virtual int artdaq::TransferInterface::source_rank ( ) const [inline],[virtual]`

Get the source rank for this [TransferInterface](#) instance.

**Returns**

The source rank for this Transferinterface instance

Definition at line 181 of file TransferInterface.hh.

6.246.4.10 `virtual CopyStatus artdaq::TransferInterface::transfer_fragment_min_blocking_mode ( artdaq::Fragment const & fragment, size_t send_timeout_usec ) [pure virtual]`

Transfer a Fragment to the destination. May not necessarily be reliable, but will not block longer than send\_timeout\_usec.

**Parameters**

<i>fragment</i>	Fragment to transfer
<i>send_timeout_ - usec</i>	<a href="#">Timeout</a> for send, in microseconds

**Returns**

CopyStatus detailing result of transfer

Implemented in [artdaq::BundleTransfer](#), [artdaq::MulticastTransfer](#), [artdaq::AutodetectTransfer](#), [artdaq::NullTransfer](#), and [artdaq::RTIDDSTransfer](#).

6.246.4.11 `virtual CopyStatus artdaq::TransferInterface::transfer_fragment_reliable_mode ( artdaq::Fragment && fragment ) [pure virtual]`

Transfer a Fragment to the destination. This should be reliable, if the underlying transport mechanism supports reliable sending.

**Parameters**

<i>fragment</i>	Fragment to transfer
-----------------	----------------------

**Returns**

CopyStatus detailing result of copy

Implemented in [artdaq::BundleTransfer](#), [artdaq::MulticastTransfer](#), [artdaq::AutodetectTransfer](#), [artdaq::NullTransfer](#), and [artdaq::RTIDDSTransfer](#).

6.246.4.12 `std::string artdaq::TransferInterface::uniqueLabel ( ) const [inline]`

Get the unique label of this [TransferInterface](#) instance.

#### Returns

The unique label of this [TransferInterface](#) instance

Definition at line 175 of file TransferInterface.hh.

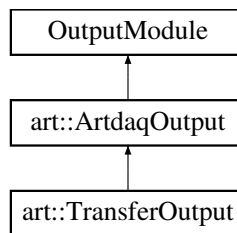
The documentation for this class was generated from the following files:

- artdaq/artdaq/TransferPlugins/TransferInterface.hh
- artdaq/artdaq/TransferPlugins/TransferInterface.cc

## 6.247 art::TransferOutput Class Reference

An `art::OutputModule` which sends events using `DataSenderManager`. This module is designed for transporting Fragment-wrapped `art::Events` after they have been read into art, for example between the `EventBuilder` and the `Aggregator`.

Inheritance diagram for `art::TransferOutput`:



### Public Member Functions

- [TransferOutput](#) (`fhicl::ParameterSet const &ps`)  
*TransferOutput Constructor.*
- [~TransferOutput](#) () override  
*TransferOutput Destructor.*

### Protected Member Functions

- void [SendMessage](#) (`artdaq::FragmentPtr &fragment`) override  
*Send a message using the Transfer Plugin*

### 6.247.1 Detailed Description

An `art::OutputModule` which sends events using `DataSenderManager`. This module is designed for transporting Fragment-wrapped `art::Events` after they have been read into art, for example between the `EventBuilder` and the `Aggregator`.

Definition at line 20 of file TransferOutput\_module.cc.

## 6.247.2 Constructor & Destructor Documentation

6.247.2.1 `art::TransferOutput::TransferOutput ( fhicl::ParameterSet const & ps )` `[explicit]`

[TransferOutput](#) Constructor.

Parameters

<i>ps</i>	ParameterSet used to configure <a href="#">TransferOutput</a>
-----------	---

[TransferOutput](#) accepts no Parameters beyond those which `art::OutputModule` takes. See the `art::OutputModule` documentation for more details on those Parameters.

Definition at line 55 of file `TransferOutput_module.cc`.

## 6.247.3 Member Function Documentation

6.247.3.1 `void art::TransferOutput::SendMessage ( artdaq::FragmentPtr & fragment )` `[override]`, `[protected]`, `[virtual]`

Send a message using the Transfer Plugin

Parameters

<i>fragment</i>	Fragment to send
-----------------	------------------

Implements [art::ArtdaqOutput](#).

Definition at line 76 of file `TransferOutput_module.cc`.

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

- `artdaq/artdaq/ArtModules/TransferOutput_module.cc`

## 6.248 artdaq::TransferTest Class Reference

Test a set of [TransferInterface](#) plugins.

```
#include <artdaq/DAQrate/TransferTest.hh>
```

### Public Member Functions

- [TransferTest](#) (fhicl::ParameterSet psi)  
*[TransferTest](#) Constructor.*
- `int runTest ()`  
*Run the test as configured.*
- `int returnCode ()`  
*Get the result of the test.*

### 6.248.1 Detailed Description

Test a set of [TransferInterface](#) plugins.

Definition at line 16 of file `TransferTest.hh`.

## 6.248.2 Constructor & Destructor Documentation

### 6.248.2.1 artdaq::TransferTest::TransferTest ( fhicl::ParameterSet *psi* ) [explicit]

[TransferTest](#) Constructor.

Parameters

<i>psi</i>	ParameterSet used to configure <a href="#">TransferTest</a>
------------	---

```
* TransferTest accepts the following Parameters:
* "num_senders" (REQUIRED): Number of sending TransferTest instances
* "num_receivers" (REQUIRED): Number of receiving TransferTest instances
* "sends_per_sender" (REQUIRED): Number of sends each sender will perform
* "sending_threads" (Default: 1): Number of TransferInterface instances to send fragments from for each source rank
* "buffer_count" (Default: 10): Buffer count for TransferInterfaces
* "fragment_size" (Default: 0x100000): Size of Fragments to transfer
* "metrics": FHiCL table used to configure MetricManager (see documentation)
* "transfer_plugin_type" (Default: Shmem): TransferInterface plugin to load
* "hostmap" (OPTIONAL): Host map to use for "host_map" parameter of TransferInterface plugins (i.e. TCPSocketTransferInterface)
*
```

Definition at line 12 of file TransferTest.cc.

## 6.248.3 Member Function Documentation

### 6.248.3.1 int artdaq::TransferTest::returnCode ( ) [inline]

Get the result of the test.

Returns

Test result (Unix-style, 0 is success)

Definition at line 48 of file TransferTest.hh.

### 6.248.3.2 int artdaq::TransferTest::runTest ( )

Run the test as configured.

Returns

0 upon success

Definition at line 105 of file TransferTest.cc.

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

- artdaq/artdaq/DAQrate/TransferTest.hh
- artdaq/artdaq/DAQrate/TransferTest.cc

## 6.249 artdaq::TransferWrapper Class Reference

[TransferWrapper](#) wraps a [TransferInterface](#) so that it can be used in the ArtdaqInput class to make an art::Source.

```
#include <artdaq/ArtModules/detail/TransferWrapper.hh>
```

## Public Member Functions

- [TransferWrapper](#) (const fhicl::ParameterSet &pset)  
*[TransferWrapper](#) Constructor.*
- virtual [~TransferWrapper](#) ()  
*[TransferWrapper](#) Destructor.*
- artdaq::FragmentPtrs [receiveMessage](#) ()  
*Receive a Fragment from the [TransferInterface](#), and send it to art.*
- std::unordered\_map  
< artdaq::Fragment::type\_t,  
std::unique\_ptr  
< artdaq::Fragments > > [receiveMessages](#) ()  
*Receive all messages for an event from [ArtdaqSharedMemoryService](#).*
- artdaq::FragmentPtrs [receiveInitMessage](#) ()  
*Receive the Init message from the [TransferInterface](#), and send it to art.*
- std::shared\_ptr  
< artdaq::detail::RawEventHeader > [getEventHeader](#) ()  
*Get a pointer to the last received RawEventHeader.*

### 6.249.1 Detailed Description

[TransferWrapper](#) wraps a [TransferInterface](#) so that it can be used in the ArtdaqInput class to make an art::Source.

JCF, May-27-2016

This is the class through which code that wants to access a transfer plugin (e.g., input sources, AggregatorCore, etc.) can do so. Its functionality is such that it satisfies the requirements needed to be a template in the ArtdaqInput class

Definition at line 31 of file TransferWrapper.hh.

### 6.249.2 Constructor & Destructor Documentation

#### 6.249.2.1 artdaq::TransferWrapper::TransferWrapper ( const fhicl::ParameterSet & pset ) [explicit]

[TransferWrapper](#) Constructor.

Parameters

<i>pset</i>	ParameterSet used to configure the <a href="#">TransferWrapper</a>
-------------	--

```
* TransferWrapper accepts the following Parameters:
* "timeoutInUsecs" (Default: 100000): The receive timeout
* "dispatcherHost" (REQUIRED): The hostname that the Dispatcher Aggregator is running on
* "dispatcherPort" (REQUIRED): The port that the Dispatcher Aggregator is running on
* "maxEventsBeforeInit" (Default: 5): How many non-Init events to receive before raising an error
* "allowedFragmentTypes" (Default: [226,227,229]): The Fragment type codes for expected Fragments
* "dispatcherConnectTimeout" (Default: 0): Maximum amount of time (in seconds) to wait for the Dispatcher to reach
* "dispatcherConnectRetryInterval_us" (Default 1,000,000): Amount of time to wait between polls of the Dispatcher
* "quitOnFragmentIntegrityProblem" (Default: true): If there is an inconsistency in the received Fragment, throw
* "allowMultipleRuns" (Default: false): If true, will ignore EndOfData message and reconnect to the Dispatcher on
* "debugLevel" (Default: 0): Enables some additional messages
* "transfer_plugin" (REQUIRED): Name of the TransferInterface plugin to load
*
* This parameter set is also passed to TransferInterface, so any necessary Parameters for TransferInterface or the
* should be included here.
*
```

Definition at line 37 of file TransferWrapper.cc.

### 6.249.3 Member Function Documentation

6.249.3.1 `std::shared_ptr<artdaq::detail::RawEventHeader> artdaq::TransferWrapper::getEventHeader ( )` `[inline]`

Get a pointer to the last received RawEventHeader.

Returns

a shared\_ptr to the last received RawEventHeader

Definition at line 87 of file TransferWrapper.hh.

6.249.3.2 `artdaq::FragmentPtrs artdaq::TransferWrapper::receiveInitMessage ( )` `[inline]`

Receive the Init message from the [TransferInterface](#), and send it to art.

Returns

Received InitFragment

Definition at line 78 of file TransferWrapper.hh.

6.249.3.3 `artdaq::FragmentPtrs artdaq::TransferWrapper::receiveMessage ( )`

Receive a Fragment from the [TransferInterface](#), and send it to art.

Returns

Received Fragment

Definition at line 92 of file TransferWrapper.cc.

6.249.3.4 `std::unordered_map< artdaq::Fragment::type_t, std::unique_ptr< artdaq::Fragments > > artdaq::TransferWrapper::receiveMessages ( )`

Receive all messages for an event from [ArtdaqSharedMemoryService](#).

Returns

A map of Fragment::type\_t to a unique\_ptr to Fragments containing all Fragments in an event

Definition at line 220 of file TransferWrapper.cc.

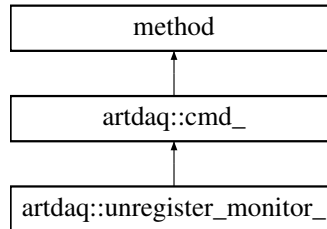
The documentation for this class was generated from the following files:

- artdaq/artdaq/ArtModules/detail/TransferWrapper.hh
- artdaq/artdaq/ArtModules/detail/TransferWrapper.cc

## 6.250 artdaq::unregister\_monitor\_ Class Reference

[unregister\\_monitor\\_](#) Command class

Inheritance diagram for artdaq::unregister\_monitor\_:



### Public Member Functions

- [unregister\\_monitor\\_](#) ([xmlrpc\\_commander](#) &c)  
[unregister\\_monitor\\_](#) Constructor

### Additional Inherited Members

#### 6.250.1 Detailed Description

[unregister\\_monitor\\_](#) Command class

Definition at line 1084 of file `xmlrpc_commander.cc`.

#### 6.250.2 Constructor & Destructor Documentation

6.250.2.1 `artdaq::unregister_monitor_::unregister_monitor_ ( xmlrpc_commander & c ) [inline]`

[unregister\\_monitor\\_](#) Constructor

Parameters

<code>c</code>	<a href="#">xmlrpc_commander</a> to send transition commands to
----------------	---

Definition at line 1091 of file `xmlrpc_commander.cc`.

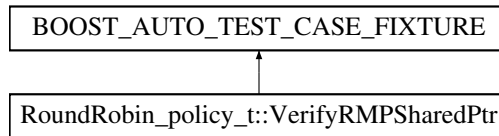
The documentation for this class was generated from the following file:

- `artdaq/artdaq/ExternalComms/xmlrpc_commander.cc`

## 6.251 RoundRobin\_policy\_t::VerifyRMPSHaredPtr Struct Reference

Inheritance diagram for RoundRobin\_policy\_t::VerifyRMPSHaredPtr:





### Public Member Functions

- void **test\_method** ()

#### 6.251.1 Detailed Description

Definition at line 10 of file RoundRobin\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/RoundRobin\_policy\_t.cc

## 6.252 RoundRobin\_policy\_t::VerifyRMPSHaredPtr\_id Struct Reference

#### 6.252.1 Detailed Description

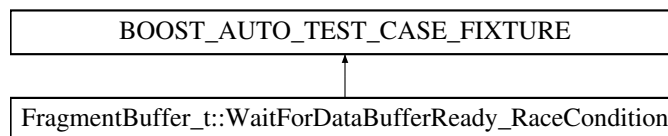
Definition at line 10 of file RoundRobin\_policy\_t.cc.

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

- artdaq/test/RoutingPolicies/RoundRobin\_policy\_t.cc

## 6.253 FragmentBuffer\_t::WaitForDataBufferReady\_RaceCondition Struct Reference

Inheritance diagram for FragmentBuffer\_t::WaitForDataBufferReady\_RaceCondition:



### Public Member Functions

- void **test\_method** ()

#### 6.253.1 Detailed Description

Definition at line 2403 of file FragmentBuffer\_t.cc.

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

- `artdaq/test/DAQrate/FragmentBuffer_t.cc`

## 6.254 `FragmentBuffer_t::WaitForDataBufferReady_RaceCondition_id` Struct Reference

### 6.254.1 Detailed Description

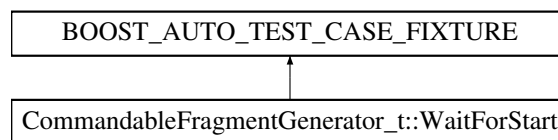
Definition at line 2403 of file `FragmentBuffer_t.cc`.

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

- `artdaq/test/DAQrate/FragmentBuffer_t.cc`

## 6.255 `CommandableFragmentGenerator_t::WaitForStart` Struct Reference

Inheritance diagram for `CommandableFragmentGenerator_t::WaitForStart`:



### Public Member Functions

- `void test_method ()`

### 6.255.1 Detailed Description

Definition at line 211 of file `CommandableFragmentGenerator_t.cc`.

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

- `artdaq/test/Generators/CommandableFragmentGenerator_t.cc`

## 6.256 `CommandableFragmentGenerator_t::WaitForStart_id` Struct Reference

### 6.256.1 Detailed Description

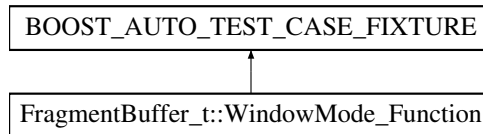
Definition at line 211 of file `CommandableFragmentGenerator_t.cc`.

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

- `artdaq/test/Generators/CommandableFragmentGenerator_t.cc`

## 6.257 FragmentBuffer\_t::WindowMode\_Function Struct Reference

Inheritance diagram for FragmentBuffer\_t::WindowMode\_Function:



### Public Member Functions

- void **test\_method** ()

#### 6.257.1 Detailed Description

Definition at line 521 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.258 FragmentBuffer\_t::WindowMode\_Function\_id Struct Reference

#### 6.258.1 Detailed Description

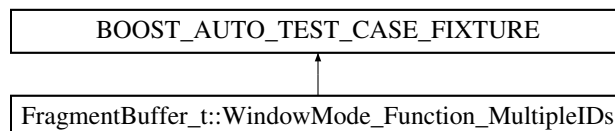
Definition at line 521 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.259 FragmentBuffer\_t::WindowMode\_Function\_MultipleIDs Struct Reference

Inheritance diagram for FragmentBuffer\_t::WindowMode\_Function\_MultipleIDs:



### Public Member Functions

- void **test\_method** ()

### 6.259.1 Detailed Description

Definition at line 1629 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.260 FragmentBuffer\_t::WindowMode\_Function\_MultipleIDs\_id Struct Reference

### 6.260.1 Detailed Description

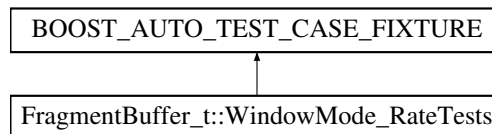
Definition at line 1629 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.261 FragmentBuffer\_t::WindowMode\_RateTests Struct Reference

Inheritance diagram for FragmentBuffer\_t::WindowMode\_RateTests:



### Public Member Functions

- void **test\_method** ()

### 6.261.1 Detailed Description

Definition at line 2169 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.262 FragmentBuffer\_t::WindowMode\_RateTests\_id Struct Reference

### 6.262.1 Detailed Description

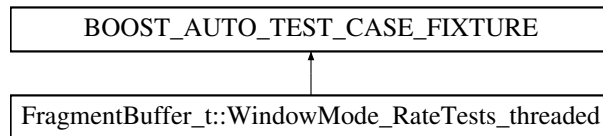
Definition at line 2169 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.263 FragmentBuffer\_t::WindowMode\_RateTests\_threaded Struct Reference

Inheritance diagram for FragmentBuffer\_t::WindowMode\_RateTests\_threaded:



### Public Member Functions

- void **test\_method** ()

#### 6.263.1 Detailed Description

Definition at line 2313 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.264 FragmentBuffer\_t::WindowMode\_RateTests\_threaded\_id Struct Reference

#### 6.264.1 Detailed Description

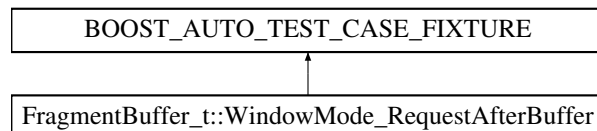
Definition at line 2313 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.265 FragmentBuffer\_t::WindowMode\_RequestAfterBuffer Struct Reference

Inheritance diagram for FragmentBuffer\_t::WindowMode\_RequestAfterBuffer:



### Public Member Functions

- void **test\_method** ()

### 6.265.1 Detailed Description

Definition at line 1011 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.266 FragmentBuffer\_t::WindowMode\_RequestAfterBuffer\_id Struct Reference

### 6.266.1 Detailed Description

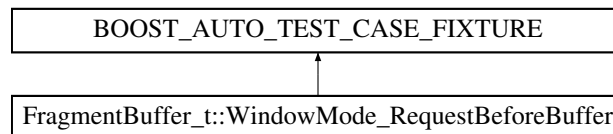
Definition at line 1011 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.267 FragmentBuffer\_t::WindowMode\_RequestBeforeBuffer Struct Reference

Inheritance diagram for FragmentBuffer\_t::WindowMode\_RequestBeforeBuffer:



### Public Member Functions

- void **test\_method** ()

### 6.267.1 Detailed Description

Definition at line 706 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.268 FragmentBuffer\_t::WindowMode\_RequestBeforeBuffer\_id Struct Reference

### 6.268.1 Detailed Description

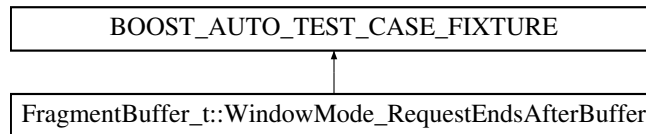
Definition at line 706 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.269 FragmentBuffer\_t::WindowMode\_RequestEndsAfterBuffer Struct Reference

Inheritance diagram for FragmentBuffer\_t::WindowMode\_RequestEndsAfterBuffer:



### Public Member Functions

- void **test\_method** ()

#### 6.269.1 Detailed Description

Definition at line 936 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.270 FragmentBuffer\_t::WindowMode\_RequestEndsAfterBuffer\_id Struct Reference

#### 6.270.1 Detailed Description

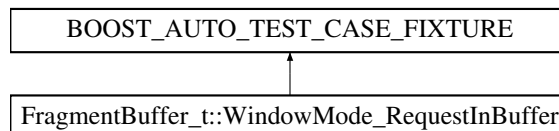
Definition at line 936 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.271 FragmentBuffer\_t::WindowMode\_RequestInBuffer Struct Reference

Inheritance diagram for FragmentBuffer\_t::WindowMode\_RequestInBuffer:



### Public Member Functions

- void **test\_method** ()

### 6.271.1 Detailed Description

Definition at line 891 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.272 FragmentBuffer\_t::WindowMode\_RequestInBuffer\_id Struct Reference

### 6.272.1 Detailed Description

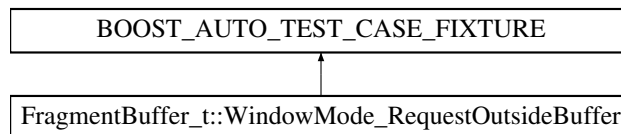
Definition at line 891 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.273 FragmentBuffer\_t::WindowMode\_RequestOutsideBuffer Struct Reference

Inheritance diagram for FragmentBuffer\_t::WindowMode\_RequestOutsideBuffer:



### Public Member Functions

- void **test\_method** ()

### 6.273.1 Detailed Description

Definition at line 798 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.274 FragmentBuffer\_t::WindowMode\_RequestOutsideBuffer\_id Struct Reference

### 6.274.1 Detailed Description

Definition at line 798 of file FragmentBuffer\_t.cc.

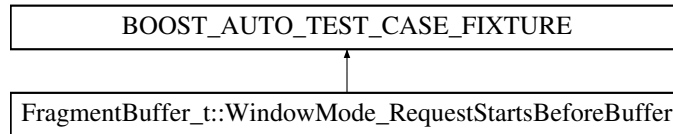
The documentation for this struct was generated from the following file:

- artdaq/test/DAQrate/FragmentBuffer\_t.cc



## 6.275 FragmentBuffer\_t::WindowMode\_RequestStartsBeforeBuffer Struct Reference

Inheritance diagram for FragmentBuffer\_t::WindowMode\_RequestStartsBeforeBuffer:



### Public Member Functions

- void **test\_method** ()

#### 6.275.1 Detailed Description

Definition at line 752 of file FragmentBuffer\_t.cc.

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

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.276 FragmentBuffer\_t::WindowMode\_RequestStartsBeforeBuffer\_id Struct Reference

#### 6.276.1 Detailed Description

Definition at line 752 of file FragmentBuffer\_t.cc.

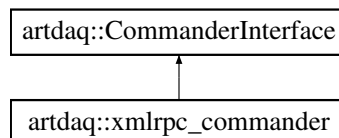
The documentation for this struct was generated from the following file:

- artdaq/test/DAQrate/FragmentBuffer\_t.cc

## 6.277 artdaq::xmlrpc\_commander Class Reference

The [xmlrpc\\_commander](#) class serves as the XMLRPC server run in each artdaq application.

Inheritance diagram for artdaq::xmlrpc\_commander:



### Public Member Functions

- [xmlrpc\\_commander](#) (const fhicl::ParameterSet &ps, [artdaq::Commandable](#) &commandable)

[xmlrpc\\_commander](#) Constructor

- void [run\\_server](#) () override  
*Run the XMLRPC server.*
- std::string [send\\_register\\_monitor](#) (std::string const &monitor\_fhicl) override  
*Send a register\_monitor command over XMLRPC*
- std::string [send\\_unregister\\_monitor](#) (std::string const &monitor\_label) override  
*Send an unregister\_monitor command over XMLRPC*
- std::string [send\\_init](#) (fhicl::ParameterSet const &ps, uint64\_t timeout, uint64\_t timestamp) override  
*Send an init command over XMLRPC*
- std::string [send\\_soft\\_init](#) (fhicl::ParameterSet const &ps, uint64\_t timeout, uint64\_t timestamp) override  
*Send a soft\_init command over XMLRPC*
- std::string [send\\_reinit](#) (fhicl::ParameterSet const &ps, uint64\_t timeout, uint64\_t timestamp) override  
*Send a reinit command over XMLRPC*
- std::string [send\\_start](#) (art::RunID runNumber, uint64\_t timeout, uint64\_t timestamp) override  
*Send a start command over XMLRPC*
- std::string [send\\_pause](#) (uint64\_t timeout, uint64\_t timestamp) override  
*Send a pause command over XMLRPC*
- std::string [send\\_resume](#) (uint64\_t timeout, uint64\_t timestamp) override  
*Send a resume command over XMLRPC*
- std::string [send\\_stop](#) (uint64\_t timeout, uint64\_t timestamp) override  
*Send a stop command over XMLRPC*
- std::string [send\\_shutdown](#) (uint64\_t timeout) override  
*Send a shutdown command over XMLRPC*
- std::string [send\\_status](#) () override  
*Send a status command over XMLRPC*
- std::string [send\\_report](#) (std::string const &what) override  
*Send a report command over XMLRPC*
- std::string [send\\_legal\\_commands](#) () override  
*Send a legal\_commands command over XMLRPC*
- std::string [send\\_trace\\_get](#) (std::string const &name) override  
*Send an send\_trace\_get command over XMLRPC*
- std::string [send\\_trace\\_set](#) (std::string const &name, std::string const &type, std::string const &mask) override  
*Send an send\_trace\_msgfacility\_set command over XMLRPC*
- std::string [send\\_meta\\_command](#) (std::string const &command, std::string const &argument) override  
*Send an send\_meta\_command command over XMLRPC*
- std::string [send\\_rollover\\_subrun](#) (uint64\_t when, uint32\_t sr) override  
*Send a send\_rollover\_subrun command over XMLRPC*

## Public Attributes

- std::timed\_mutex [mutex\\_](#)  
*XMLRPC mutex.*
- std::unique\_ptr  
< xmlrpc\_c::serverAbyss > [server](#)  
*XMLRPC server.*

## Additional Inherited Members

### 6.277.1 Detailed Description

The [xmlrpc\\_commander](#) class serves as the XMLRPC server run in each artdaq application.

Definition at line 43 of file `xmlrpc_commander.cc`.

### 6.277.2 Constructor & Destructor Documentation

6.277.2.1 `artdaq::xmlrpc_commander::xmlrpc_commander ( const fhicl::ParameterSet & ps, artdaq::Commandable & commandable )`

[xmlrpc\\_commander](#) Constructor

Parameters

<i>ps</i>	ParameterSet used for configuring <a href="#">xmlrpc_commander</a>
<i>commandable</i>	<a href="#">artdaq::Commandable</a> object to send transition commands to

`xmlrpc_commander` accepts the following Parameters:  
*id*: For XMLRPC, the ID should be the port to listen on  
*server\_url*: When sending, location of XMLRPC server  
 \*

Definition at line 1311 of file `xmlrpc_commander.cc`.

### 6.277.3 Member Function Documentation

6.277.3.1 `std::string artdaq::xmlrpc_commander::send_init ( fhicl::ParameterSet const & ps, uint64_t timeout, uint64_t timestamp )`  
`[override], [virtual]`

Send an init command over XMLRPC

The init command is accepted by all artdaq processes that are in the booted state. It expects a ParameterSet for configuration, a timeout, and a timestamp.

Parameters

<i>ps</i>	ParameterSet received with the init command
<i>timeout</i>	<a href="#">Timeout</a> for the command
<i>timestamp</i>	Timestamp of the command

Returns

Command result: "SUCCESS" if succeeded

Reimplemented from [artdaq::CommanderInterface](#).

Definition at line 1747 of file `xmlrpc_commander.cc`.

6.277.3.2 `std::string artdaq::xmlrpc_commander::send_legal_commands ( )` `[override], [virtual]`

Send a `legal_commands` command over XMLRPC

This will query the artdaq process, and it will return the list of allowed transition commands from its current state.

**Returns**

Command result: a list of allowed transition commands from its current state

Reimplemented from [artdaq::CommanderInterface](#).

Definition at line 1787 of file xmlrpc\_commander.cc.

**6.277.3.3** `std::string artdaq::xmlrpc_commander::send_meta_command ( std::string const & command, std::string const & argument )` `[override], [virtual]`

Send an send\_meta\_command command over XMLRPC

This will cause the receiver to run the given command with the given argument in user code

**Parameters**

<i>command</i>	Command name to send
<i>argument</i>	Argument for command

**Returns**

Command result: "SUCCESS" if succeeded

Reimplemented from [artdaq::CommanderInterface](#).

Definition at line 1799 of file xmlrpc\_commander.cc.

**6.277.3.4** `std::string artdaq::xmlrpc_commander::send_pause ( uint64_t timeout, uint64_t timestamp )` `[override], [virtual]`

Send a pause command over XMLRPC

The pause command pauses a Run. When the run resumes, the subrun number will be incremented. This command accepts a timeout parameter and a timestamp parameter.

**Parameters**

<i>timeout</i>	<a href="#">Timeout</a> for the command
<i>timestamp</i>	Timestamp of the command

**Returns**

Command result: "SUCCESS" if succeeded

Reimplemented from [artdaq::CommanderInterface](#).

Definition at line 1763 of file xmlrpc\_commander.cc.

**6.277.3.5** `std::string artdaq::xmlrpc_commander::send_register_monitor ( std::string const & monitor_fhicl )` `[override], [virtual]`

Send a register\_monitor command over XMLRPC

## Parameters

<i>monitor_thicl</i>	FHiCL string containing monitor configuration
----------------------	---

## Returns

Return status from XMLRPC

Reimplemented from [artdaq::CommanderInterface](#).

Definition at line 1739 of file xmlrpc\_commander.cc.

**6.277.3.6** `std::string artdaq::xmlrpc_commander::send_reinit ( fhicl::ParameterSet const & ps, uint64_t timeout, uint64_t timestamp ) [override],[virtual]`

Send a reinit command over XMLRPC

The reinit command is accepted by all artdaq processes. It expects a ParameterSet for configuration, a timeout, and a timestamp.

## Parameters

<i>ps</i>	ParameterSet received with the reinit command
<i>timeout</i>	<a href="#">Timeout</a> for the command
<i>timestamp</i>	Timestamp of the command

## Returns

Command result: "SUCCESS" if succeeded

Reimplemented from [artdaq::CommanderInterface](#).

Definition at line 1755 of file xmlrpc\_commander.cc.

**6.277.3.7** `std::string artdaq::xmlrpc_commander::send_report ( std::string const & what ) [override],[virtual]`

Send a report command over XMLRPC

The report command returns the current value of the requested reportable quantity.

## Parameters

<i>what</i>	Reportable quantity to request
-------------	--------------------------------

## Returns

Command result: current value of the requested reportable quantity

Reimplemented from [artdaq::CommanderInterface](#).

Definition at line 1783 of file xmlrpc\_commander.cc.

**6.277.3.8** `std::string artdaq::xmlrpc_commander::send_resume ( uint64_t timeout, uint64_t timestamp ) [override],[virtual]`

Send a resume command over XMLRPC

The resume command resumes a paused Run. When the run resumes, the subrun number will be incremented. This command accepts a timeout parameter and a timestamp parameter.

## Parameters

<i>timeout</i>	<a href="#">Timeout</a> for the command
<i>timestamp</i>	Timestamp of the command

## Returns

Command result: "SUCCESS" if succeeded

Reimplemented from [artdaq::CommanderInterface](#).

Definition at line 1767 of file xmlrpc\_commander.cc.

**6.277.3.9** `std::string artdaq::xmlrpc_commander::send_rollover_subrun ( uint64_t when, uint32_t sr ) [override], [virtual]`

Send a send\_rollover\_subrun command over XMLRPC

This will cause the receiver to rollover the subrun number at the given event. (Event with seqID == boundary will be in new subrun.) Should be sent to all EventBuilders before the given event is processed.

## Parameters

<i>when</i>	Sequence ID of new subrun
<i>sr</i>	Subrun number of the new subrun

## Returns

Command result: "SUCCESS" if succeeded

Reimplemented from [artdaq::CommanderInterface](#).

Definition at line 1803 of file xmlrpc\_commander.cc.

**6.277.3.10** `std::string artdaq::xmlrpc_commander::send_shutdown ( uint64_t timeout ) [override], [virtual]`

Send a shutdown command over XMLRPC

The shutdown command shuts down the artdaq process. This command accepts a timeout parameter.

## Parameters

<i>timeout</i>	<a href="#">Timeout</a> for the command
----------------	---

## Returns

Command result: "SUCCESS" if succeeded

Reimplemented from [artdaq::CommanderInterface](#).

Definition at line 1775 of file xmlrpc\_commander.cc.

**6.277.3.11** `std::string artdaq::xmlrpc_commander::send_soft_init ( fhicl::ParameterSet const & ps, uint64_t timeout, uint64_t timestamp ) [override], [virtual]`

Send a soft\_init command over XMLRPC

The `soft_init` command is accepted by all artdaq processes that are in the booted state. It expects a `ParameterSet` for configuration, a timeout, and a timestamp.



## Parameters

<i>ps</i>	ParameterSet received with the soft_init command
<i>timeout</i>	<a href="#">Timeout</a> for the command
<i>timestamp</i>	Timestamp of the command

## Returns

Command result: "SUCCESS" if succeeded

Reimplemented from [artdaq::CommanderInterface](#).

Definition at line 1751 of file xmlrpc\_commander.cc.

**6.277.3.12** `std::string artdaq::xmlrpc_commander::send_start ( art::RunID runNumber, uint64_t timeout, uint64_t timestamp ) [override], [virtual]`

Send a start command over XMLRPC

The start command starts a Run using the given run number. This command also accepts a timeout parameter and a timestamp parameter.

## Parameters

<i>runNumber</i>	Run number of the new run
<i>timeout</i>	<a href="#">Timeout</a> for the command
<i>timestamp</i>	Timestamp of the command

## Returns

Command result: "SUCCESS" if succeeded

Reimplemented from [artdaq::CommanderInterface](#).

Definition at line 1759 of file xmlrpc\_commander.cc.

**6.277.3.13** `std::string artdaq::xmlrpc_commander::send_status ( ) [override], [virtual]`

Send a status command over XMLRPC

The status command returns the current status of the artdaq process.

## Returns

Command result: current status of the artdaq process

Reimplemented from [artdaq::CommanderInterface](#).

Definition at line 1779 of file xmlrpc\_commander.cc.

**6.277.3.14** `std::string artdaq::xmlrpc_commander::send_stop ( uint64_t timeout, uint64_t timestamp ) [override], [virtual]`

Send a stop command over XMLRPC

The stop command stops the current Run. This command accepts a timeout parameter and a timestamp parameter.

## Parameters

<i>timeout</i>	Timeout for the command
<i>timestamp</i>	Timestamp of the command

## Returns

Command result: "SUCCESS" if succeeded

Reimplemented from [artdaq::CommanderInterface](#).

Definition at line 1771 of file xmlrpc\_commander.cc.

6.277.3.15 `std::string artdaq::xmlrpc_commander::send_trace_get ( std::string const & name ) [override],[virtual]`

Send an `send_trace_get` command over XMLRPC

This will cause the receiver to get the TRACE level masks for the given name Use name == "ALL" to get ALL names

## Parameters

<i>name</i>	TRACE name to get the mask for ("ALL" to get all names)
-------------	---

## Returns

Command result: "SUCCESS" if succeeded

Reimplemented from [artdaq::CommanderInterface](#).

Definition at line 1791 of file xmlrpc\_commander.cc.

6.277.3.16 `std::string artdaq::xmlrpc_commander::send_trace_set ( std::string const & name, std::string const & type, std::string const & mask ) [override],[virtual]`

Send an `send_trace_msgfacility_set` command over XMLRPC

This will cause the receiver to set the given TRACE level mask for the given name to the given mask. Only the first character of the mask selection will be parsed, dial 'M' for Memory, or 'S' for Slow. Use name == "ALL" to set ALL names

EXAMPLE: xmlrpc `http://localhost:5235/RPC2` daq.trace\_msgfacility\_set TraceLock i8/\$(0x1234) # Use Bash to convert hex to dec

## Parameters

<i>name</i>	TRACE name to set ("ALL" for all TRACE names)
<i>type</i>	Type of mask to set ('M', 'S', or 'T')
<i>mask</i>	64-bit mask, in string form

## Returns

Command result: "SUCCESS" if succeeded

Reimplemented from [artdaq::CommanderInterface](#).

Definition at line 1795 of file xmlrpc\_commander.cc.

6.277.3.17 `std::string artdaq::xmlrpc_commander::send_unregister_monitor ( std::string const & monitor_label )`  
[`override`], [`virtual`]

Send an unregister\_monitor command over XMLRPC

#### Parameters

<i>monitor_label</i>	Label of the monitor to unregister
----------------------	------------------------------------

#### Returns

Return status from XMLRPC

Reimplemented from [artdaq::CommanderInterface](#).

Definition at line 1743 of file xmlrpc\_commander.cc.

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

- artdaq/artdaq/ExternalComms/xmlrpc\_commander.cc

## Chapter 7

# File Documentation

### 7.1 artdaq/artdaq/DAQdata/TCP\_listen\_fd.hh File Reference

#### Functions

- int [TCP\\_listen\\_fd](#) (int port, int rcvbuf)  
*Create a TCP listening socket on the given port and INADDR\_ANY, with the given receive buffer.*

#### 7.1.1 Detailed Description

Defines a generator function for a TCP listen socket

Definition in file [TCP\\_listen\\_fd.hh](#).

#### 7.1.2 Function Documentation

##### 7.1.2.1 int TCP\_listen\_fd ( int port, int rcvbuf )

Create a TCP listening socket on the given port and INADDR\_ANY, with the given receive buffer.

#### Parameters

<i>port</i>	Port to listen on
<i>rcvbuf</i>	Receive buffer for socket. Set to 0 for TCP automatic buffer size

#### Returns

fd of new socket

Definition at line 24 of file TCP\_listen\_fd.cc.

### 7.2 artdaq/artdaq/DAQdata/TCPConnect.hh File Reference

```
#include <netinet/in.h>
#include <string>
```

## Functions

- int [ResolveHost](#) (char const \*host\_in, in\_addr &addr)  
*Convert a string hostname to a in\_addr suitable for socket communication.*
- int [GetIPOfInterface](#) (const std::string &interface\_name, in\_addr &addr)  
*Get the IP address associated with a given interface name.*
- int [AutodetectPrivateInterface](#) (in\_addr &addr)  
*Pick a private IP address on this host.*
- int [GetInterfaceForNetwork](#) (char const \*host\_in, in\_addr &addr)  
*Convert an IP address to the network address of the interface sharing the subnet mask.*
- int [ResolveHost](#) (char const \*host\_in, int dflt\_port, sockaddr\_in &sin)  
*Convert a string hostname and port to a sockaddr\_in suitable for socket communication.*
- int [TCPConnect](#) (char const \*host\_in, int dflt\_port, int64\_t flags=0, int sndbufsiz=0)  
*Connect to a host on a given port.*

### 7.2.1 Detailed Description

Provides utility functions for connecting TCP sockets

Definition in file [TCPConnect.hh](#).

### 7.2.2 Function Documentation

#### 7.2.2.1 int AutodetectPrivateInterface ( in\_addr & addr )

Pick a private IP address on this host.

##### Parameters

out	addr	in_addr object populated with resolved host
-----	------	---

##### Returns

0 if success, -1 if gethostbyname fails, 2 if defaulted to 0.0.0.0 (No matching interfaces)

The following preference order is used:

1. 192.168.0.0/16
2. 172.16.0.0/12
3. 10.0.0.0/8
4. 131.225.0.0/16
5. 0.0.0.0 (returns 2)

Definition at line 132 of file TCPConnect.cc.

#### 7.2.2.2 int GetInterfaceForNetwork ( char const \* host\_in, in\_addr & addr )

Convert an IP address to the network address of the interface sharing the subnet mask.

## Parameters

	<i>host_in</i>	IP to resolve
out	<i>addr</i>	in_addr object populated with resolved host

## Returns

0 if success, -1 if gethostbyname fails, 2 if defaulted to 0.0.0.0 (No matching interfaces)

Definition at line 224 of file TCPConnect.cc.

### 7.2.2.3 int GetIPOfInterface ( const std::string & *interface\_name*, in\_addr & *addr* )

Get the IP address associated with a given interface name.

## Parameters

	<i>interface_name</i>	Name of the interface to resolve
out	<i>addr</i>	in_addr object populated with interface IP

## Returns

0 if success, -1 if gethostbyname fails, 2 if defaulted to 0.0.0.0 (No matching interfaces)

Definition at line 78 of file TCPConnect.cc.

### 7.2.2.4 int ResolveHost ( char const \* *host\_in*, in\_addr & *addr* )

Convert a string hostname to a in\_addr suitable for socket communication.

## Parameters

	<i>host_in</i>	Name or IP of host to resolve
out	<i>addr</i>	in_addr object populated with resolved host

## Returns

0 if success, -1 if gethostbyname fails

Definition at line 34 of file TCPConnect.cc.

### 7.2.2.5 int ResolveHost ( char const \* *host\_in*, int *dflt\_port*, sockaddr\_in & *sin* )

Convert a string hostname and port to a sockaddr\_in suitable for socket communication.

## Parameters

	<i>host_in</i>	Name or IP of host to resolve
	<i>dflt_port</i>	Port to populate in output

out	sin	sockaddr_in object populated with resolved host and port
-----	-----	--

**Returns**

0 if success, -1 if gethostbyname fails

Definition at line 317 of file TCPConnect.cc.

**7.2.2.6** `int TCPConnect ( char const * host_in, int dflt_port, int64_t flags = 0, int sndbufsiz = 0 )`

Connect to a host on a given port.

**Parameters**

<i>host_in</i>	Name or IP of the host to connect to
<i>dflt_port</i>	Port to connect to
<i>flags</i>	TCP flags to use for the socket
<i>sndbufsiz</i>	Size of the send buffer. Set to 0 for automatic send buffer management

**Returns**

File descriptor of connected socket.

Definition at line 377 of file TCPConnect.cc.

## 7.3 artdaq/tools/swig\_artdaq.h File Reference

```
#include "TRACE/tracemf.h"
```

**Classes**

- class [swig\\_artdaq](#)

*Simple class exposing methods for TRACEing and sending metrics that can be wrapped with SWIG.*

### 7.3.1 Detailed Description

This file serves as an entry point for a SWIG module which can be used to send MessageFacility messages or metrics

Definition in file [swig\\_artdaq.h](#).



# Index

- ~ArtdaqOutput
  - art::ArtdaqOutput, [61](#)
- ~CommandableFragmentGenerator
  - artdaq::CommandableFragmentGenerator, [122](#)
- ~DataLoggerCore
  - artdaq::DataLoggerCore, [182](#)
- ~DataReceiverCore
  - artdaq::DataReceiverCore, [185](#)
- ~DispatcherCore
  - artdaq::DispatcherCore, [201](#)
- ~EventBuilderCore
  - artdaq::EventBuilderCore, [213](#)
- ~FragmentBuffer
  - artdaq::FragmentBuffer, [220](#)
- ~RoutingManagerCore
  - artdaq::RoutingManagerCore, [328](#)
- \_commandable
  - artdaq::CommanderInterface, [147](#)
- add\_config\_archive\_entry
  - artdaq::CommanderInterface, [136](#)
  - artdaq::DataReceiverCore, [185](#)
- add\_config\_archive\_entry\_
  - artdaq::add\_config\_archive\_entry\_, [45](#)
- add\_periodic
  - Timeout, [399](#)
- add\_relative
  - Timeout, [401](#)
- AddFragment
  - artdaq::SharedMemoryEventManager, [352](#)
- AddFragmentsToBuffer
  - artdaq::FragmentBuffer, [221](#)
- addMonitoredQuantityName
  - artdaq::StatisticsHelper, [378](#)
- AddReceiverToken
  - artdaq::RoutingManagerPolicy, [334](#)
- AddRequest
  - artdaq::RequestSender, [308](#)
- addRequest
  - artdaq::detail::RequestMessage, [300](#)
- addSample
  - artdaq::StatisticsHelper, [378](#)
- analyze
  - artdaq::EventDump, [215](#)
  - artdaq::FragmentSniffer, [234](#)
  - artdaq::FragmentWatcher, [237](#)
  - artdaq::MissingDataCheck, [263](#)
  - artdaq::RequestSenderModule, [311](#)
- anonymous\_namespace{ListenTransferWrapper.cc}, [24](#)
- anonymous\_namespace{RootDAQOut\_module.cc}, [24](#)
- anonymous\_namespace{RootDAQOutFile.cc}, [24](#)
- anonymous\_namespace{TransferWrapper.cc}, [25](#)
- anonymous\_namespace{genToArt.cc}, [23](#)
  - process\_cmd\_line, [23](#)
  - process\_data, [23](#)
- anonymous\_namespace{genToArt.cc}::ThrottledGenerator,
  - [395](#)
  - getNext, [396](#)
  - numFragIDs, [396](#)
  - start, [396](#)
  - stop, [396](#)
  - ThrottledGenerator, [396](#)
- anonymous\_namespace{xmlrpc\_commander.cc}, [25](#)
- anonymous\_namespace{xmlrpc\_commander.cc}::env\_
  - wrap, [207](#)
- applyRequests
  - artdaq::FragmentBuffer, [221](#)
- applyRequestsBufferMode
  - artdaq::FragmentBuffer, [221](#)
- applyRequestsIgnoredMode
  - artdaq::FragmentBuffer, [221](#)
- applyRequestsSequenceIDMode
  - artdaq::FragmentBuffer, [221](#)
- applyRequestsSingleMode
  - artdaq::FragmentBuffer, [223](#)
- applyRequestsWindowMode
  - artdaq::FragmentBuffer, [223](#)
- applyRequestsWindowMode\_CheckAndFillDataBuffer
  - artdaq::FragmentBuffer, [223](#)
- art, [25](#)
  - printProcessHistoryID, [27](#)
  - printProcessMap, [27](#)
  - ReadObjectAny, [28](#)
  - TransferInput, [27](#)
- art::AnalyzersConfig, [46](#)
- art::ArtdaqFragmentNamingServiceInterfaceConfig, [52](#)
- art::ArtdaqInputHelper
  - ArtdaqInputHelper, [56](#)
  - hasMoreData, [56](#)
  - operator=, [56](#)
  - readFile, [56](#)

- readNext, 58
- art::ArtdaqInputHelper< U >, 55
- art::ArtdaqOutput, 58
  - ~ArtdaqOutput, 61
  - ArtdaqOutput, 60
  - beginRun, 61
  - beginRun\_, 61
  - beginSubRun, 61
  - beginSubRun\_, 61
  - closeFile, 61
  - endJob, 61
  - event, 62
  - event\_, 62
  - extractProducts\_, 62
  - openFile, 62
  - respondToCloseInputFile, 62
  - respondToCloseOutputFiles, 62
  - send\_init\_message, 62
  - SendMessage, 63
  - write, 63
  - writeDataProducts, 63
  - writeRun, 63
  - writeSubRun, 63
- art::ArtdaqSharedMemoryServiceInterfaceConfig, 68
- art::BinaryFileOutput, 72
  - BinaryFileOutput, 73
- art::BinaryNetOutput, 73
  - BinaryNetOutput, 74
- art::Config, 155
  - source, 155
- art::FiltersConfig, 215
- art::OutputItem, 275
- art::OutputsConfig, 275
- art::PhysicsConfig, 277
  - filters, 277
- art::ProducersConfig, 286
- art::RootDAQOut, 314
- art::RootDAQOut::Config, 155
  - dropMetaDataForDroppedData, 156
  - fileProperties, 156
  - saveMemoryObjectThreshold, 157
- art::RootDAQOut::Config::FileNameSubstitution, 215
- art::RootDAQOut::Config::KeysToIgnore, 253
- art::RootDAQOut::Config::NewSubStringForApp, 268
- art::RootDAQOutFile, 315
- art::RootNetOutput, 316
  - dataReceiverCount, 317
  - RootNetOutput, 317
  - SendMessage, 317
- art::RootOutputConfig, 318
  - Comment, 319
- art::RootOutputConfig::KeysToIgnore, 252
  - operator(), 252
- art::ServicesConfig, 348
- art::ShmemWrapper, 363
  - getEventHeader, 364
  - receiveInitMessage, 364
  - receiveMessage, 364
  - receiveMessages, 365
  - ShmemWrapper, 364
- art::Source\_generator< ArtdaqInputHelper< artdaq::ListenTransferWrapper > >, 373
- art::Source\_generator< ArtdaqInputHelper< artdaq::TransferWrapper > >, 374
- art::Source\_generator< ArtdaqInputHelper< ShmemWrapper > >, 374
- art::SourceConfig, 375
- art::TransferOutput, 418
  - SendMessage, 419
  - TransferOutput, 419
- art\_config\_file
  - artdaq::art\_config\_file, 48
- artdaq, 28
  - cmd\_::getParam< art::RunID >, 34
  - cmd\_::getParam< fhicl::ParameterSet >, 34
  - cmd\_::getParam< std::string >, 36
  - exception\_msg, 36, 37
  - makeCommandableFragmentGenerator, 39
  - MakeCommanderPlugin, 39
  - makeFunc\_t, 34
  - MakeHostMap, 39
  - MakeHostMapPset, 40
  - makeRoutingManagerPolicy, 40
  - MakeTransferPlugin, 40
- artdaq/artdaq/DAQdata/TCP\_listen\_fd.hh, 445
- artdaq/artdaq/DAQdata/TCPConnect.hh, 445
- artdaq/tools/swig\_artdaq.h, 448
- artdaq::GenericFragmentSimulator
  - DEAD\_BEEF, 239
  - EMPTY, 239
  - FRAG\_ID, 239
  - RANDOM, 239
- artdaq::TransferInterface
  - DATA\_END, 413
  - kErrorNotRequiringException, 413
  - kReceive, 413
  - kSend, 413
  - kSuccess, 413
  - kTimeout, 413
  - NO\_RANK\_INFO, 413
  - RECV\_SUCCESS, 413
  - RECV\_TIMEOUT, 413
- artdaq::detail
  - DataFlow, 42
  - EndOfRun, 42
  - EventBuilding, 42
  - Normal, 42
  - RequestBasedEventBuilding, 42

- artdaq::AutodetectTransfer, 69
  - AutodetectTransfer, 70
  - isRunning, 70
  - receiveFragment, 71
  - receiveFragmentData, 71
  - receiveFragmentHeader, 71
  - transfer\_fragment\_min\_blocking\_mode, 72
  - transfer\_fragment\_reliable\_mode, 72
- artdaq::BoardReaderApp, 74
  - BootedEnter, 76
  - do\_initialize, 76
  - do\_meta\_command, 76
  - do\_pause, 76
  - do\_reinitialize, 78
  - do\_resume, 78
  - do\_shutdown, 78
  - do\_soft\_initialize, 79
  - do\_start, 79
  - do\_stop, 79
  - operator=, 80
  - report, 80
- artdaq::BoardReaderCore, 80
  - BoardReaderCore, 82
  - GetDataSenderManagerPtr, 82
  - GetFragmentsProcessed, 83
  - GetReceiverThreadActive, 83
  - GetSenderThreadActive, 83
  - initialize, 83
  - metaCommand, 84
  - operator=, 84
  - pause, 84
  - receive\_fragments, 84
  - reinitialize, 85
  - report, 85
  - resume, 85
  - send\_fragments, 85
  - SetStartTransitionTimeout, 86
  - shutdown, 86
  - soft\_initialize, 86
  - start, 86
  - stop, 87
- artdaq::BuildInfo
  - beginRun, 92
  - BuildInfo, 92
  - produce, 92
- artdaq::BuildInfo< instanceName, Pkgs >, 91
- artdaq::BundleTransfer, 93
  - BundleTransfer, 93
  - isRunning, 94
  - receiveFragment, 94
  - receiveFragmentData, 94
  - receiveFragmentHeader, 94
  - transfer\_fragment\_min\_blocking\_mode, 96
  - transfer\_fragment\_reliable\_mode, 96
- artdaq::CapacityTestPolicy, 96
  - CapacityTestPolicy, 97
  - CreateRouteForSequenceID, 97
  - CreateRoutingTable, 99
- artdaq::Commandable, 107
  - badTransition, 110
  - BootedEnter, 110
  - Commandable, 110
  - current\_state, 110
  - do\_add\_config\_archive\_entry, 110
  - do\_clear\_config\_archive, 111
  - do\_initialize, 111
  - do\_meta\_command, 111
  - do\_pause, 111
  - do\_reinitialize, 112
  - do\_resume, 112
  - do\_rollover\_subrun, 112
  - do\_shutdown, 113
  - do\_soft\_initialize, 113
  - do\_start, 113
  - do\_stop, 113
  - do\_trace\_get, 114
  - do\_trace\_set, 114
  - in\_run\_failure, 114
  - InRunExit, 115
  - initialize, 115
  - legal\_commands, 115
  - operator=, 115
  - pause, 115
  - register\_monitor, 116
  - reinitialize, 116
  - report, 116
  - resume, 116
  - shutdown, 117
  - soft\_initialize, 117
  - start, 117
  - status, 118
  - stop, 118
  - unregister\_monitor, 118
- artdaq::CommandableFragmentGenerator, 119
  - ~CommandableFragmentGenerator, 122
  - check\_stop, 122
  - checkHWStatus\_, 122
  - CommandableFragmentGenerator, 122
  - ev\_counter, 122
  - ev\_counter\_inc, 123
  - exception, 123
  - fragment\_id, 123
  - fragmentIDs, 123
  - getNext, 124
  - getNext\_, 124
  - GetRequestBuffer, 124
  - joinThreads, 124
  - metaCommand, 124

- metricsReportingInstanceName, 126
- pause, 126
- PauseCmd, 126
- pauseNoMutex, 127
- report, 127
- ReportCmd, 127
- reportSpecific, 127
- resume, 128
- ResumeCmd, 128
- run\_number, 128
- set\_exception, 128
- SetRequestBuffer, 128
- should\_stop, 129
- start, 129
- StartCmd, 129
- stop, 129
- StopCmd, 129
- stopNoMutex, 131
- subrun\_number, 131
- timeout, 131
- timestamp, 131
- artdaq::CommandableFragmentGenerator::Config, 149
  - fragment\_id, 149
  - fragment\_ids, 150
- artdaq::CommanderInterface, 134
  - \_commandable, 147
  - add\_config\_archive\_entry, 136
  - clear\_config\_archive, 137
  - CommanderInterface, 136
  - GetStatus, 137
  - operator=, 137
  - run\_server, 137
  - send\_init, 138
  - send\_legal\_commands, 138
  - send\_meta\_command, 138
  - send\_pause, 139
  - send\_register\_monitor, 139
  - send\_reinit, 139
  - send\_report, 141
  - send\_resume, 141
  - send\_rollover\_subrun, 141
  - send\_shutdown, 143
  - send\_soft\_init, 143
  - send\_start, 143
  - send\_status, 145
  - send\_stop, 145
  - send\_trace\_get, 145
  - send\_trace\_set, 146
  - send\_unregister\_monitor, 146
- artdaq::CommanderInterface::Config, 172
- artdaq::CompositeDriver, 147
  - CompositeDriver, 148
- artdaq::DataLoggerApp, 177
  - do\_add\_config\_archive\_entry, 178
  - do\_clear\_config\_archive, 178
  - do\_initialize, 178
  - do\_pause, 178
  - do\_reinitialize, 179
  - do\_resume, 179
  - do\_shutdown, 179
  - do\_soft\_initialize, 179
  - do\_start, 180
  - do\_stop, 180
  - operator=, 180
  - report, 180
- artdaq::DataLoggerCore, 181
  - ~DataLoggerCore, 182
  - initialize, 182
  - operator=, 182
- artdaq::DataReceiverCore, 183
  - ~DataReceiverCore, 185
  - add\_config\_archive\_entry, 185
  - clear\_config\_archive, 185
  - initialize, 185
  - initializeDataReceiver, 186
  - operator=, 186
  - pause, 186
  - reinitialize, 186
  - report, 188
  - resume, 188
  - rollover\_subrun, 188
  - shutdown, 188
  - soft\_initialize, 189
  - start, 189
  - stop, 189
- artdaq::DataReceiverManager, 190
  - byteCount, 191
  - count, 191
  - DataReceiverManager, 190
  - enabled\_sources, 191
  - getSharedMemoryEventManager, 191
  - running\_sources, 192
  - slotCount, 192
- artdaq::DataSenderManager, 192
  - count, 194
  - DataSenderManager, 193
  - destinationCount, 194
  - enabled\_destinations, 194
  - GetRemainingRoutingTableEntries, 194
  - GetRoutingTableEntryCount, 194
  - GetSentSequenceIDCount, 194
  - RemoveRoutingTableEntry, 195
  - sendFragment, 195
  - slotCount, 195
- artdaq::DataSenderManager::Config, 161
  - destinations, 162
  - routing\_table\_config, 162
- artdaq::DataSenderManager::DestinationsConfig, 196

- artdaq::DispatcherApp, 196
  - do\_initialize, 197
  - do\_pause, 197
  - do\_reinitialize, 198
  - do\_resume, 198
  - do\_shutdown, 198
  - do\_soft\_initialize, 198
  - do\_start, 199
  - do\_stop, 199
  - operator=, 199
  - register\_monitor, 199
  - report, 200
  - unregister\_monitor, 200
- artdaq::DispatcherCore, 200
  - ~DispatcherCore, 201
  - initialize, 202
  - operator=, 203
  - register\_monitor, 203
  - unregister\_monitor, 203
- artdaq::EventBuilderApp, 207
  - BootedEnter, 208
  - do\_add\_config\_archive\_entry, 208
  - do\_clear\_config\_archive, 209
  - do\_initialize, 209
  - do\_pause, 209
  - do\_reinitialize, 209
  - do\_resume, 210
  - do\_rollover\_subrun, 210
  - do\_shutdown, 210
  - do\_soft\_initialize, 210
  - do\_start, 211
  - do\_stop, 211
  - operator=, 211
  - report, 211
- artdaq::EventBuilderCore, 212
  - ~EventBuilderCore, 213
  - initialize, 213
  - operator=, 213
- artdaq::EventDump, 214
  - analyze, 215
  - EventDump, 214
- artdaq::FragmentBuffer, 218
  - ~FragmentBuffer, 220
  - AddFragmentsToBuffer, 221
  - applyRequests, 221
  - applyRequestsBufferMode, 221
  - applyRequestsIgnoredMode, 221
  - applyRequestsSequenceIDMode, 221
  - applyRequestsSingleMode, 223
  - applyRequestsWindowMode, 223
  - applyRequestsWindowMode\_CheckAndFillData-Buffer, 223
  - check\_stop, 223
  - checkDataBuffer, 223
  - checkSentWindows, 224
  - dataBufferFragmentCount\_, 224
  - dataBufferIsTooLarge, 224
  - fragment\_id, 224
  - FragmentBuffer, 220
  - fragmentIDs, 224
  - getDataBufferStats, 225
  - GetNextSequenceID, 225
  - GetSentWindowList, 225
  - printMode\_, 225
  - request\_mode, 225
  - Reset, 226
  - sendEmptyFragment, 226
  - sendEmptyFragments, 226
  - SetRequestBuffer, 226
  - waitForDataBufferReady, 227
- artdaq::FragmentBuffer::Config, 165
  - fragment\_id, 167
  - fragment\_ids, 168
  - request\_mode, 168
- artdaq::FragmentReceiverManager, 229
  - byteCount, 230
  - count, 230
  - enabled\_sources, 231
  - FragmentReceiverManager, 230
  - recvFragment, 231
  - running\_sources, 231
  - slotCount, 231
- artdaq::FragmentSniffer, 232
  - analyze, 234
  - FragmentSniffer, 233
- artdaq::FragmentStoreElement, 234
  - emplace\_back, 235
  - emplace\_front, 235
  - empty, 235
  - front, 235
  - GetEndOfData, 235
  - SetEndOfData, 236
  - size, 236
- artdaq::FragmentWatcher, 236
  - analyze, 237
  - FragmentWatcher, 237
- artdaq::GenToBufferTest, 241
  - GenToBufferTest, 241
  - GetRequestBuffer, 242
  - start, 242
- artdaq::GenericFragmentSimulator, 237
  - content\_selector\_t, 238
  - fragmentIDs, 239
  - GenericFragmentSimulator, 239
  - getNext, 239
- artdaq::GenericFragmentSimulator::Config, 157
  - content\_selection, 158
  - starting\_fragment\_id, 158

- artdaq::GetPackageBuildInfo, 242
- getPackageBuildInfo, 242
- artdaq::Globals, 243
  - GetMFIteration\_, 244
  - GetMFModuleName\_, 244
  - getPartitionNumber\_, 244
  - seedAndRandom\_, 244
  - SetMFIteration\_, 244
  - SetMFModuleName\_, 246
- artdaq::HostMap, 248
- artdaq::HostMap::Config, 158
  - host\_map, 159
- artdaq::HostMap::HostConfig, 247
- artdaq::ListenTransferWrapper, 255
  - getEventHeader, 256
  - ListenTransferWrapper, 256
  - receiveInitMessage, 257
  - receiveMessage, 257
  - receiveMessages, 257
- artdaq::MissingDataCheck, 262
  - analyze, 263
  - MissingDataCheck, 263
- artdaq::MulticastTransfer, 263
  - isRunning, 265
  - MulticastTransfer, 264
  - receiveFragment, 265
  - receiveFragmentData, 265
  - receiveFragmentHeader, 265
  - transfer\_fragment\_min\_blocking\_mode, 266
  - transfer\_fragment\_reliable\_mode, 266
- artdaq::NetMonHeader, 267
- artdaq::NoOpPolicy, 268
  - CreateRouteForSequenceID, 269
  - CreateRoutingTable, 269
  - NoOpPolicy, 269
- artdaq::NullTransfer, 270
  - isRunning, 271
  - NullTransfer, 271
  - receiveFragment, 271
  - receiveFragmentData, 272
  - receiveFragmentHeader, 272
  - transfer\_fragment\_min\_blocking\_mode, 272
  - transfer\_fragment\_reliable\_mode, 272
- artdaq::OffsetPrescale, 274
- artdaq::PortManager, 277
  - GetMulticastOutputAddress, 279
  - GetMulticastTransferGroupAddress, 280
  - GetMulticastTransferPort, 280
  - GetRequestMessageGroupAddress, 280
  - GetRequestMessagePort, 280
  - GetRoutingAckPort, 281
  - GetRoutingTableGroupAddress, 281
  - GetRoutingTablePort, 281
  - GetRoutingTokenPort, 281
  - GetTCPSocketTransferPort, 282
  - GetXMLRPCPort, 282
  - UpdateConfiguration, 282
- artdaq::PortManager::Config, 159
- artdaq::PreferSameHostPolicy, 283
  - CreateRouteForSequenceID, 284
  - CreateRoutingTable, 285
  - PreferSameHostPolicy, 283
- artdaq::PrintBuildInfo, 285
  - beginRun, 286
  - PrintBuildInfo, 286
- artdaq::RTIDDS, 343
  - RTIDDS, 344
- artdaq::RTIDDS::OctetsListener, 273
  - on\_data\_available, 273
  - receiveFragmentFromDDS, 274
- artdaq::RTIDDSTransfer, 344
  - isRunning, 346
  - RTIDDSTransfer, 345
  - receiveFragment, 346
  - transfer\_fragment\_min\_blocking\_mode, 346
  - transfer\_fragment\_reliable\_mode, 346
- artdaq::RandomDelayFilter, 287
  - filter, 288
  - operator=, 288
  - RandomDelayFilter, 287
- artdaq::RequestBuffer, 294
  - ClearRequests, 296
  - GetAndClearRequests, 296
  - GetNextRequest, 296
  - GetRequestTime, 296
  - GetRequests, 296
  - isRunning, 297
  - push, 297
  - RemoveRequest, 297
  - RequestBuffer, 295
  - setRunning, 297
  - size, 297
  - WaitForRequests, 298
- artdaq::RequestReceiver, 304
  - isRunning, 306
  - RequestReceiver, 305
  - SetRunNumber, 306
  - stopRequestReception, 306
- artdaq::RequestReceiver::Config, 163
- artdaq::RequestSender, 307
  - AddRequest, 308
  - GetRequestMode, 309
  - GetSentMessageCount, 309
  - operator=, 309
  - RemoveRequest, 309
  - RequestSender, 308
  - RequestsInFlight, 309
  - SendRequest, 309

- SetRequestMode, 310
- SetRunNumber, 310
- artdaq::RequestSender::Config, 163
- artdaq::RequestSenderModule, 310
  - analyze, 311
  - beginRun, 311
  - endRun, 311
  - RequestSenderModule, 311
- artdaq::RoundRobinPolicy, 319
  - CreateRouteForSequenceID, 320
  - CreateRoutingTable, 320
  - RoundRobinPolicy, 320
- artdaq::RoutingManagerApp, 322
  - BootedEnter, 323
  - do\_initialize, 323
  - do\_pause, 324
  - do\_reinitialize, 324
  - do\_resume, 324
  - do\_shutdown, 325
  - do\_soft\_initialize, 325
  - do\_start, 325
  - do\_stop, 326
  - operator=, 326
  - report, 326
  - RoutingManagerApp, 323
- artdaq::RoutingManagerCore, 327
  - ~RoutingManagerCore, 328
  - get\_update\_count, 328
  - initialize, 328
  - operator=, 329
  - pause, 329
  - reinitialize, 329
  - report, 329
  - resume, 330
  - send\_event\_table, 330
  - shutdown, 330
  - soft\_initialize, 330
  - start, 331
  - stop, 331
- artdaq::RoutingManagerPolicy, 332
  - AddReceiverToken, 334
  - CacheHasRoute, 334
  - CreateRouteForSequenceID, 335
  - CreateRoutingTable, 335
  - GetCacheSize, 335
  - GetCurrentTable, 335
  - GetHeldTokenCount, 336
  - GetMaxNumberOfTokens, 336
  - GetNextSequenceID, 336
  - GetReceiverCount, 336
  - GetRouteForSequenceID, 336
  - GetRoutingMode, 337
  - GetTokensUsedSinceLastUpdate, 337
  - RoutingManagerPolicy, 334
- artdaq::RoutingReceiverConfig, 339
- artdaq::SharedMemoryEventManager, 349
  - AddFragment, 352
  - DoneWritingFragment, 353
  - endOfData, 353
  - endRun, 353
  - GetArtEventCount, 353
  - GetBroadcastKey, 353
  - GetCurrentSubrun, 354
  - GetDroppedDataAddress, 354
  - GetFragmentCount, 354
  - GetFragmentCountInBuffer, 354
  - GetLockedBufferCount, 355
  - GetOpenEventCount, 355
  - GetPendingEventCount, 355
  - GetSubrunForSequenceID, 355
  - ReconfigureArt, 356
  - rolloverSubrun, 356
  - runID, 356
  - setOverwrite, 356
  - setRequestMode, 356
  - SharedMemoryEventManager, 351
  - ShutdownArtProcesses, 358
  - StartArtProcess, 358
  - startRun, 358
  - UpdateArtConfiguration, 358
  - WriteFragmentHeader, 358
- artdaq::SharedMemoryEventManager::Config, 169
  - expected\_art\_event\_processing\_time\_us, 171
  - fragment\_broadcast\_timeout\_ms, 171
  - max\_event\_size\_bytes, 171
  - max\_fragment\_size\_bytes, 172
- artdaq::ShmemTransfer, 360
  - isRunning, 361
  - receiveFragment, 361
  - receiveFragmentData, 362
  - receiveFragmentHeader, 362
  - ShmemTransfer, 361
  - transfer\_fragment\_min\_blocking\_mode, 362
  - transfer\_fragment\_reliable\_mode, 363
- artdaq::StatisticsHelper, 377
  - addMonitoredQuantityName, 378
  - addSample, 378
  - createCollectors, 379
  - operator=, 379
  - readyToReport, 379
  - statsRollingWindowHasMoved, 379
- artdaq::TCPSocketTransfer, 392
  - isRunning, 393
  - receiveFragmentData, 394
  - receiveFragmentHeader, 394
  - TCPSocketTransfer, 393
  - transfer\_fragment\_min\_blocking\_mode, 394
  - transfer\_fragment\_reliable\_mode, 395



- artdaq::TableReceiver, 390
  - GetRemainingRoutingTableEntries, 391
  - GetRoutingTableEntry, 391
  - GetRoutingTableEntryCount, 392
  - RemoveRoutingTableEntry, 392
  - TableReceiver, 391
- artdaq::TableReceiver::Config, 164
- artdaq::TokenReceiver, 403
  - getReceivedTokenCount, 405
  - setRunNumber, 405
  - setStatsHelper, 405
  - stopTokenReception, 405
  - TokenReceiver, 404
- artdaq::TokenReceiver::Config, 165
- artdaq::TokenSender, 406
  - GetSentTokenCount, 408
  - operator=, 408
  - RoutingTokenSendsEnabled, 408
  - SendRoutingToken, 408
  - SetRunNumber, 408
  - TokenSender, 407
- artdaq::TokenSender::Config, 168
- artdaq::TransferInterface, 410
  - CopyStatus, 413
  - CopyStatusToString, 414
  - destination\_rank, 414
  - isRunning, 414
  - operator=, 414
  - receiveFragment, 414
  - receiveFragmentData, 416
  - receiveFragmentHeader, 416
  - ReceiveReturnCode, 413
  - Role, 413
  - role, 416
  - source\_rank, 417
  - transfer\_fragment\_min\_blocking\_mode, 417
  - transfer\_fragment\_reliable\_mode, 417
  - TransferInterface, 413
  - uniqueLabel, 417
- artdaq::TransferInterface::Config, 150
- artdaq::TransferTest, 419
  - returnCode, 420
  - runTest, 420
  - TransferTest, 420
- artdaq::TransferWrapper, 420
  - getEventHeader, 422
  - receiveInitMessage, 422
  - receiveMessage, 422
  - receiveMessages, 422
  - TransferWrapper, 421
- artdaq::add\_config\_archive\_entry\_, 45
  - add\_config\_archive\_entry\_, 45
- artdaq::art\_config\_file, 47
  - art\_config\_file, 48
  - getFileName, 48
- artdaq::artdaqapp, 48
  - runArtdaqApp, 49
- artdaq::artdaqapp::Config, 151
  - commanderPluginConfig, 152
- artdaq::clear\_config\_archive\_, 101
  - clear\_config\_archive\_, 102
- artdaq::cmd\_, 102
  - cmd\_, 104
  - execute, 104
  - execute\_, 104
  - getParam, 106, 107
- artdaq::detail, 41
  - IntToTaskType, 43
  - operator<=, 44
  - RequestMessageMode, 42
  - RoutingManagerMode, 42
  - StringToTaskType, 44
  - TaskTypeToString, 44
- artdaq::detail::FragCounter, 215
  - count, 216
  - incSlot, 216
  - minCount, 217
  - nSlots, 217
  - setSlot, 217
  - slotCount, 217
- artdaq::detail::RequestHeader, 298
  - header, 299
  - isValid, 299
- artdaq::detail::RequestMessage, 299
  - addRequest, 300
  - GetMessage, 300
  - max\_request\_count, 300
  - setMode, 300
  - setRank, 302
  - setRunNumber, 302
  - size, 302
- artdaq::detail::RequestPacket, 302
  - header, 303
  - isValid, 303
  - RequestPacket, 303
- artdaq::detail::RoutingManagerModeConverter, 331
  - routingManagerModeToString, 332
  - stringToRoutingManagerMode, 332
- artdaq::detail::RoutingPacketEntry, 337
  - RoutingPacketEntry, 338
- artdaq::detail::RoutingPacketHeader, 338
  - RoutingPacketHeader, 339
- artdaq::detail::RoutingRequest, 340
  - RequestModeToString, 342
  - RoutingRequest, 341, 342
- artdaq::detail::RoutingToken, 342
- artdaq::init\_, 251
  - defaultTimeout, 252



- defaultTimestamp, [252](#)
- init\_, [251](#)
- artdaq::legal\_commands\_, [254](#)
  - legal\_commands\_, [255](#)
- artdaq::meta\_command\_, [260](#)
  - meta\_command\_, [260](#)
- artdaq::pause\_, [276](#)
  - defaultTimeout, [276](#)
  - defaultTimestamp, [276](#)
  - pause\_, [276](#)
- artdaq::register\_monitor\_, [289](#)
  - register\_monitor\_, [289](#)
- artdaq::reinit\_, [289](#)
  - defaultTimeout, [290](#)
  - defaultTimestamp, [290](#)
  - reinit\_, [290](#)
- artdaq::report\_, [291](#)
  - report\_, [291](#)
- artdaq::resume\_, [312](#)
  - defaultTimeout, [313](#)
  - defaultTimestamp, [313](#)
  - resume\_, [312](#)
- artdaq::rollover\_subrun\_, [313](#)
  - rollover\_subrun\_, [314](#)
- artdaq::shutdown\_, [365](#)
  - defaultTimeout, [366](#)
  - shutdown\_, [366](#)
- artdaq::soft\_init\_, [372](#)
  - defaultTimeout, [373](#)
  - defaultTimestamp, [373](#)
  - soft\_init\_, [373](#)
- artdaq::start\_, [375](#)
  - defaultTimeout, [376](#)
  - defaultTimestamp, [376](#)
  - start\_, [376](#)
- artdaq::status\_, [380](#)
  - status\_, [380](#)
- artdaq::stop\_, [381](#)
  - defaultTimeout, [382](#)
  - defaultTimestamp, [382](#)
  - stop\_, [381](#)
- artdaq::trace\_get\_, [409](#)
  - trace\_get\_, [409](#)
- artdaq::trace\_set\_, [410](#)
  - trace\_set\_, [410](#)
- artdaq::unregister\_monitor\_, [423](#)
  - unregister\_monitor\_, [423](#)
- artdaq::xmlrpc\_commander, [432](#)
  - send\_init, [434](#)
  - send\_legal\_commands, [434](#)
  - send\_meta\_command, [435](#)
  - send\_pause, [435](#)
  - send\_register\_monitor, [435](#)
  - send\_reinit, [436](#)
  - send\_report, [436](#)
  - send\_resume, [436](#)
  - send\_rollover\_subrun, [438](#)
  - send\_shutdown, [438](#)
  - send\_soft\_init, [438](#)
  - send\_start, [440](#)
  - send\_status, [440](#)
  - send\_stop, [440](#)
  - send\_trace\_get, [441](#)
  - send\_trace\_set, [441](#)
  - send\_unregister\_monitor, [441](#)
  - xmlrpc\_commander, [434](#)
- ArtdaqFragmentNamingService, [49](#)
  - ArtdaqFragmentNamingService, [50](#)
  - ArtdaqFragmentNamingService, [50](#)
- ArtdaqFragmentNamingServiceInterface, [50](#)
  - ArtdaqFragmentNamingServiceInterface, [51](#)
  - ArtdaqFragmentNamingServiceInterface, [51](#)
  - GetUnidentifiedInstanceName, [51](#)
- ArtdaqGlobalsService, [52](#)
  - ArtdaqGlobalsService, [54](#)
  - ArtdaqGlobalsService, [54](#)
  - GetEventHeader, [54](#)
  - GetMyId, [54](#)
  - GetQueueCapacity, [54](#)
  - GetQueueSize, [54](#)
  - ReceiveEvent, [55](#)
- ArtdaqGlobalsService::Config, [152](#)
- ArtdaqInputHelper
  - art::ArtdaqInputHelper, [56](#)
- ArtdaqOutput
  - art::ArtdaqOutput, [60](#)
- ArtdaqSharedMemoryService, [64](#)
  - ArtdaqSharedMemoryService, [65](#)
  - ArtdaqSharedMemoryService, [65](#)
  - GetEventHeader, [65](#)
  - GetMyId, [65](#)
  - GetQueueCapacity, [65](#)
  - GetQueueSize, [66](#)
  - ReceiveEvent, [66](#)
- ArtdaqSharedMemoryService::Config, [154](#)
- ArtdaqSharedMemoryServiceInterface, [66](#)
  - GetEventHeader, [67](#)
  - GetMyId, [67](#)
  - GetQueueCapacity, [68](#)
  - GetQueueSize, [68](#)
  - ReceiveEvent, [68](#)
- artdaqtest::BrokenTransferTest, [87](#)
  - BrokenTransferTest, [88](#)
  - TestReceiverPause, [88](#)
  - TestReceiverReconnect, [88](#)
  - TestSenderPause, [88](#)
  - TestSenderReconnect, [88](#)
- artdaqtest::BrokenTransferTest::Config, [160](#)

- artdaqtest::CommandableFragmentGeneratorTest, 132
  - checkHWStatus\_, 133
  - getNext\_, 133
  - getTimestamp, 133
  - setEnabledIds, 133
  - setFireCount, 134
  - setTimestamp, 134
- artdaqtest::FragmentBufferTestGenerator, 227
  - Generate, 228
  - getTimestamp, 229
  - setTimestamp, 229
- AutodetectPrivateInterface
  - TCPConnect.hh, 446
- AutodetectTransfer
  - artdaq::AutodetectTransfer, 70
- badTransition
  - artdaq::Commandable, 110
- beginRun
  - art::ArtdaqOutput, 61
  - artdaq::BuildInfo, 92
  - artdaq::PrintBuildInfo, 286
  - artdaq::RequestSenderModule, 311
- beginRun\_
  - art::ArtdaqOutput, 61
- beginSubRun
  - art::ArtdaqOutput, 61
- beginSubRun\_
  - art::ArtdaqOutput, 61
- BinaryFileOutput
  - art::BinaryFileOutput, 73
- BinaryNetOutput
  - art::BinaryNetOutput, 74
- BoardReaderCore
  - artdaq::BoardReaderCore, 82
- BootedEnter
  - artdaq::BoardReaderApp, 76
  - artdaq::Commandable, 110
  - artdaq::EventBuilderApp, 208
  - artdaq::RoutingManagerApp, 323
- BrokenTransferTest
  - artdaqtest::BrokenTransferTest, 88
- BuildInfo
  - artdaq::BuildInfo, 92
- BundleTransfer
  - artdaq::BundleTransfer, 93
- byteCount
  - artdaq::DataReceiverManager, 191
  - artdaq::FragmentReceiverManager, 230
- CacheHasRoute
  - artdaq::RoutingManagerPolicy, 334
- cancel\_timeout
  - Timeout, 401
- CapacityTest\_policy\_t::DataFlowMode\_id, 174
- CapacityTest\_policy\_t::RequestBasedEventBuilding, 291
- CapacityTest\_policy\_t::RequestBasedEventBuilding\_id, 293
- CapacityTest\_policy\_t::Simple, 367
- CapacityTest\_policy\_t::Simple\_id, 370
- CapacityTestPolicy
  - artdaq::CapacityTestPolicy, 97
- check\_stop
  - artdaq::CommandableFragmentGenerator, 122
  - artdaq::FragmentBuffer, 223
- checkDataBuffer
  - artdaq::FragmentBuffer, 223
- checkHWStatus\_
  - artdaq::CommandableFragmentGenerator, 122
  - artdaqtest::CommandableFragmentGeneratorTest, 133
- checkSentWindows
  - artdaq::FragmentBuffer, 224
- clear\_config\_archive
  - artdaq::CommanderInterface, 137
  - artdaq::DataReceiverCore, 185
- clear\_config\_archive\_
  - artdaq::clear\_config\_archive\_, 102
- ClearRequests
  - artdaq::RequestBuffer, 296
- closeFile
  - art::ArtdaqOutput, 61
- cmd\_
  - artdaq::cmd\_, 104
- cmd\_::getParam< art::RunID >
  - artdaq, 34
- cmd\_::getParam< fhicl::ParameterSet >
  - artdaq, 34
- cmd\_::getParam< std::string >
  - artdaq, 36
- Commandable
  - artdaq::Commandable, 110
- CommandableFragmentGenerator
  - artdaq::CommandableFragmentGenerator, 122
- CommandableFragmentGenerator\_t::HardwareFailure\_NonThreaded, 246
- CommandableFragmentGenerator\_t::HardwareFailure\_NonThreaded\_id, 246
- CommandableFragmentGenerator\_t::HardwareFailure\_Threated, 247
- CommandableFragmentGenerator\_t::HardwareFailure\_Threated\_id, 247
- CommandableFragmentGenerator\_t::MultipleIDs, 266
- CommandableFragmentGenerator\_t::MultipleIDs\_id, 267
- CommandableFragmentGenerator\_t::Simple, 367
- CommandableFragmentGenerator\_t::Simple\_id, 369
- CommandableFragmentGenerator\_t::StateMachine, 377

- CommandableFragmentGenerator\_t::StateMachine\_id, [377](#)
- CommandableFragmentGenerator\_t::WaitForStart, [425](#)
- CommandableFragmentGenerator\_t::WaitForStart\_id, [425](#)
- CommanderInterface
  - artdaq::CommanderInterface, [136](#)
- commanderPluginConfig
  - artdaq::artdaqapp::Config, [152](#)
- Comment
  - art::RootOutputConfig, [319](#)
- CompositeDriver
  - artdaq::CompositeDriver, [148](#)
- Config, [152](#)
- content\_selection
  - artdaq::GenericFragmentSimulator::Config, [158](#)
- content\_selector\_t
  - artdaq::GenericFragmentSimulator, [238](#)
- copy\_in\_timeout
  - Timeout, [401](#)
- CopyStatus
  - artdaq::TransferInterface, [413](#)
- CopyStatusToString
  - artdaq::TransferInterface, [414](#)
- count
  - artdaq::DataReceiverManager, [191](#)
  - artdaq::DataSenderManager, [194](#)
  - artdaq::detail::FragCounter, [216](#)
  - artdaq::FragmentReceiverManager, [230](#)
- createCollectors
  - artdaq::StatisticsHelper, [379](#)
- CreateRouteForSequenceID
  - artdaq::CapacityTestPolicy, [97](#)
  - artdaq::NoOpPolicy, [269](#)
  - artdaq::PreferSameHostPolicy, [284](#)
  - artdaq::RoundRobinPolicy, [320](#)
  - artdaq::RoutingManagerPolicy, [335](#)
- CreateRoutingTable
  - artdaq::CapacityTestPolicy, [99](#)
  - artdaq::NoOpPolicy, [269](#)
  - artdaq::PreferSameHostPolicy, [285](#)
  - artdaq::RoundRobinPolicy, [320](#)
  - artdaq::RoutingManagerPolicy, [335](#)
- current\_state
  - artdaq::Commandable, [110](#)
- DATA\_END
  - artdaq::TransferInterface, [413](#)
- DEAD\_BEEF
  - artdaq::GenericFragmentSimulator, [239](#)
- DataFlow
  - artdaq::detail, [42](#)
- dataBufferFragmentCount\_
  - artdaq::FragmentBuffer, [224](#)
- dataBufferIsTooLarge
  - artdaq::FragmentBuffer, [224](#)
- dataReceiverCount
  - art::RootNetOutput, [317](#)
- DataReceiverManager
  - artdaq::DataReceiverManager, [190](#)
- DataSenderManager
  - artdaq::DataSenderManager, [193](#)
- defaultTimeout
  - artdaq::init\_, [252](#)
  - artdaq::pause\_, [276](#)
  - artdaq::reinit\_, [290](#)
  - artdaq::resume\_, [313](#)
  - artdaq::shutdown\_, [366](#)
  - artdaq::soft\_init\_, [373](#)
  - artdaq::start\_, [376](#)
  - artdaq::stop\_, [382](#)
- defaultTimestamp
  - artdaq::init\_, [252](#)
  - artdaq::pause\_, [276](#)
  - artdaq::reinit\_, [290](#)
  - artdaq::resume\_, [313](#)
  - artdaq::soft\_init\_, [373](#)
  - artdaq::start\_, [376](#)
  - artdaq::stop\_, [382](#)
- destination\_rank
  - artdaq::TransferInterface, [414](#)
- destinationCount
  - artdaq::DataSenderManager, [194](#)
- destinations
  - artdaq::DataSenderManager::Config, [162](#)
- do\_add\_config\_archive\_entry
  - artdaq::Commandable, [110](#)
  - artdaq::DataLoggerApp, [178](#)
  - artdaq::EventBuilderApp, [208](#)
- do\_clear\_config\_archive
  - artdaq::Commandable, [111](#)
  - artdaq::DataLoggerApp, [178](#)
  - artdaq::EventBuilderApp, [209](#)
- do\_initialize
  - artdaq::BoardReaderApp, [76](#)
  - artdaq::Commandable, [111](#)
  - artdaq::DataLoggerApp, [178](#)
  - artdaq::DispatcherApp, [197](#)
  - artdaq::EventBuilderApp, [209](#)
  - artdaq::RoutingManagerApp, [323](#)
- do\_meta\_command
  - artdaq::BoardReaderApp, [76](#)
  - artdaq::Commandable, [111](#)
- do\_pause
  - artdaq::BoardReaderApp, [76](#)
  - artdaq::Commandable, [111](#)
  - artdaq::DataLoggerApp, [178](#)
  - artdaq::DispatcherApp, [197](#)

- artdaq::EventBuilderApp, [209](#)
- artdaq::RoutingManagerApp, [324](#)
- do\_reinitialize
  - artdaq::BoardReaderApp, [78](#)
  - artdaq::Commandable, [112](#)
  - artdaq::DataLoggerApp, [179](#)
  - artdaq::DispatcherApp, [198](#)
  - artdaq::EventBuilderApp, [209](#)
  - artdaq::RoutingManagerApp, [324](#)
- do\_resume
  - artdaq::BoardReaderApp, [78](#)
  - artdaq::Commandable, [112](#)
  - artdaq::DataLoggerApp, [179](#)
  - artdaq::DispatcherApp, [198](#)
  - artdaq::EventBuilderApp, [210](#)
  - artdaq::RoutingManagerApp, [324](#)
- do\_rollover\_subrun
  - artdaq::Commandable, [112](#)
  - artdaq::EventBuilderApp, [210](#)
- do\_shutdown
  - artdaq::BoardReaderApp, [78](#)
  - artdaq::Commandable, [113](#)
  - artdaq::DataLoggerApp, [179](#)
  - artdaq::DispatcherApp, [198](#)
  - artdaq::EventBuilderApp, [210](#)
  - artdaq::RoutingManagerApp, [325](#)
- do\_soft\_initialize
  - artdaq::BoardReaderApp, [79](#)
  - artdaq::Commandable, [113](#)
  - artdaq::DataLoggerApp, [179](#)
  - artdaq::DispatcherApp, [198](#)
  - artdaq::EventBuilderApp, [210](#)
  - artdaq::RoutingManagerApp, [325](#)
- do\_start
  - artdaq::BoardReaderApp, [79](#)
  - artdaq::Commandable, [113](#)
  - artdaq::DataLoggerApp, [180](#)
  - artdaq::DispatcherApp, [199](#)
  - artdaq::EventBuilderApp, [211](#)
  - artdaq::RoutingManagerApp, [325](#)
- do\_stop
  - artdaq::BoardReaderApp, [79](#)
  - artdaq::Commandable, [113](#)
  - artdaq::DataLoggerApp, [180](#)
  - artdaq::DispatcherApp, [199](#)
  - artdaq::EventBuilderApp, [211](#)
  - artdaq::RoutingManagerApp, [326](#)
- do\_trace\_get
  - artdaq::Commandable, [114](#)
- do\_trace\_set
  - artdaq::Commandable, [114](#)
- DoneWritingFragment
  - artdaq::SharedMemoryEventManager, [353](#)
- dropMetaDataForDroppedData
- art::RootDAQOut::Config, [156](#)
- EMPTY
  - artdaq::GenericFragmentSimulator, [239](#)
- ELArtdaqMetric
  - mplugins::ELArtdaqMetric, [206](#)
- emplace\_back
  - artdaq::FragmentStoreElement, [235](#)
- emplace\_front
  - artdaq::FragmentStoreElement, [235](#)
- empty
  - artdaq::FragmentStoreElement, [235](#)
- enabled\_destinations
  - artdaq::DataSenderManager, [194](#)
- enabled\_sources
  - artdaq::DataReceiverManager, [191](#)
  - artdaq::FragmentReceiverManager, [231](#)
- EndOfRun
  - artdaq::detail, [42](#)
- endJob
  - art::ArtdaqOutput, [61](#)
- endOfData
  - artdaq::SharedMemoryEventManager, [353](#)
- endRun
  - artdaq::RequestSenderModule, [311](#)
  - artdaq::SharedMemoryEventManager, [353](#)
- ev\_counter
  - artdaq::CommandableFragmentGenerator, [122](#)
- ev\_counter\_inc
  - artdaq::CommandableFragmentGenerator, [123](#)
- event
  - art::ArtdaqOutput, [62](#)
- EventBuilding
  - artdaq::detail, [42](#)
- event\_
  - art::ArtdaqOutput, [62](#)
- EventDump
  - artdaq::EventDump, [214](#)
- exception
  - artdaq::CommandableFragmentGenerator, [123](#)
- exception\_msg
  - artdaq, [36](#), [37](#)
- execute
  - artdaq::cmd\_, [104](#)
- execute\_
  - artdaq::cmd\_, [104](#)
- expected\_art\_event\_processing\_time\_us
  - artdaq::SharedMemoryEventManager::Config, [171](#)
- extractProducts\_
  - art::ArtdaqOutput, [62](#)
- FRAG\_ID
  - artdaq::GenericFragmentSimulator, [239](#)
- fileProperties
  - art::RootDAQOut::Config, [156](#)

- fillPrefix
  - mfplugins::ELArtdaqMetric, 206
- fillUsrMsg
  - mfplugins::ELArtdaqMetric, 206
- filter
  - artdaq::RandomDelayFilter, 288
- filters
  - art::PhysicsConfig, 277
- FragCounter\_test::Apply, 46
- FragCounter\_test::Apply\_id, 47
- FragCounter\_test::ApplyWithOffset, 47
- FragCounter\_test::ApplyWithOffset\_id, 47
- FragCounter\_test::Construct, 172
- FragCounter\_test::Construct\_id, 173
- FragCounter\_test::nSlots, 269
- FragCounter\_test::nSlots\_id, 270
- fragment\_broadcast\_timeout\_ms
  - artdaq::SharedMemoryEventManager::Config, 171
- fragment\_id
  - artdaq::CommandableFragmentGenerator, 123
  - artdaq::CommandableFragmentGenerator::Config, 149
  - artdaq::FragmentBuffer, 224
  - artdaq::FragmentBuffer::Config, 167
- fragment\_ids
  - artdaq::CommandableFragmentGenerator::Config, 150
  - artdaq::FragmentBuffer::Config, 168
- FragmentBuffer
  - artdaq::FragmentBuffer, 220
- FragmentBuffer\_t::BufferMode, 89
- FragmentBuffer\_t::BufferMode\_KeepLatest, 89
- FragmentBuffer\_t::BufferMode\_KeepLatest\_id, 90
- FragmentBuffer\_t::BufferMode\_MultipleIDs, 90
- FragmentBuffer\_t::BufferMode\_MultipleIDs\_id, 91
- FragmentBuffer\_t::BufferMode\_id, 89
- FragmentBuffer\_t::CircularBufferMode, 99
- FragmentBuffer\_t::CircularBufferMode\_MultipleIDs, 100
- FragmentBuffer\_t::CircularBufferMode\_MultipleIDs\_id, 100
- FragmentBuffer\_t::CircularBufferMode\_RateTests, 101
- FragmentBuffer\_t::CircularBufferMode\_RateTests\_id, 101
- FragmentBuffer\_t::CircularBufferMode\_id, 100
- FragmentBuffer\_t::IgnoreRequests, 248
- FragmentBuffer\_t::IgnoreRequests\_MultipleIDs, 249
- FragmentBuffer\_t::IgnoreRequests\_MultipleIDs\_id, 249
- FragmentBuffer\_t::IgnoreRequests\_StateMachine, 249
- FragmentBuffer\_t::IgnoreRequests\_StateMachine\_id, 250
- FragmentBuffer\_t::IgnoreRequests\_id, 249
- FragmentBuffer\_t::ImproperConfiguration, 250
- FragmentBuffer\_t::ImproperConfiguration\_id, 251
- FragmentBuffer\_t::SequenceIDMode, 347
- FragmentBuffer\_t::SequenceIDMode\_MultipleIDs, 348
- FragmentBuffer\_t::SequenceIDMode\_MultipleIDs\_id, 348
- FragmentBuffer\_t::SequenceIDMode\_id, 347
- FragmentBuffer\_t::SingleMode, 370
- FragmentBuffer\_t::SingleMode\_MultipleIDs, 371
- FragmentBuffer\_t::SingleMode\_MultipleIDs\_id, 371
- FragmentBuffer\_t::SingleMode\_StateMachine, 372
- FragmentBuffer\_t::SingleMode\_StateMachine\_id, 372
- FragmentBuffer\_t::SingleMode\_id, 371
- FragmentBuffer\_t::WaitForDataBufferReady\_Race-Condition, 424
- FragmentBuffer\_t::WaitForDataBufferReady\_Race-Condition\_id, 425
- FragmentBuffer\_t::WindowMode\_Function, 426
- FragmentBuffer\_t::WindowMode\_Function\_MultipleIDs, 426
- FragmentBuffer\_t::WindowMode\_Function\_MultipleIDs\_id, 427
- FragmentBuffer\_t::WindowMode\_Function\_id, 426
- FragmentBuffer\_t::WindowMode\_RateTests, 427
- FragmentBuffer\_t::WindowMode\_RateTests\_id, 427
- FragmentBuffer\_t::WindowMode\_RateTests\_threaded, 428
- FragmentBuffer\_t::WindowMode\_RateTests\_threaded\_id, 428
- FragmentBuffer\_t::WindowMode\_RequestAfterBuffer, 428
- FragmentBuffer\_t::WindowMode\_RequestAfterBuffer\_id, 429
- FragmentBuffer\_t::WindowMode\_RequestBeforeBuffer, 429
- FragmentBuffer\_t::WindowMode\_RequestBeforeBuffer\_id, 429
- FragmentBuffer\_t::WindowMode\_RequestEndsAfter-Buffer, 430
- FragmentBuffer\_t::WindowMode\_RequestEndsAfter-Buffer\_id, 430
- FragmentBuffer\_t::WindowMode\_RequestInBuffer, 430
- FragmentBuffer\_t::WindowMode\_RequestInBuffer\_id, 431
- FragmentBuffer\_t::WindowMode\_RequestOutsideBuffer, 431
- FragmentBuffer\_t::WindowMode\_RequestOutsideBuffer\_id, 431
- FragmentBuffer\_t::WindowMode\_RequestStartsBefore-Buffer, 432
- FragmentBuffer\_t::WindowMode\_RequestStartsBefore-Buffer\_id, 432
- fragmentIDs
  - artdaq::CommandableFragmentGenerator, 123
  - artdaq::FragmentBuffer, 224
  - artdaq::GenericFragmentSimulator, 239
- FragmentReceiverManager
  - artdaq::FragmentReceiverManager, 230
- FragmentSniffer
  - artdaq::FragmentSniffer, 233
- FragmentWatcher
  - artdaq::FragmentWatcher, 237

- front
  - artdaq::FragmentStoreElement, [235](#)
- GenToBufferTest
  - artdaq::GenToBufferTest, [241](#)
- Generate
  - artdaqtest::FragmentBufferTestGenerator, [228](#)
- GenericFragmentSimulator
  - artdaq::GenericFragmentSimulator, [239](#)
- GenericFragmentSimulator\_t::Simple, [366](#)
- GenericFragmentSimulator\_t::Simple\_id, [369](#)
- get\_next\_expired\_timeout
  - Timeout, [402](#)
- get\_next\_timeout\_delay
  - Timeout, [402](#)
- get\_next\_timeout\_msdy
  - Timeout, [402](#)
- get\_update\_count
  - artdaq::RoutingManagerCore, [328](#)
- GetAndClearRequests
  - artdaq::RequestBuffer, [296](#)
- GetArtEventCount
  - artdaq::SharedMemoryEventManager, [353](#)
- GetBroadcastKey
  - artdaq::SharedMemoryEventManager, [353](#)
- GetCacheSize
  - artdaq::RoutingManagerPolicy, [335](#)
- GetCurrentSubrun
  - artdaq::SharedMemoryEventManager, [354](#)
- GetCurrentTable
  - artdaq::RoutingManagerPolicy, [335](#)
- getDataBufferStats
  - artdaq::FragmentBuffer, [225](#)
- GetDataSenderManagerPtr
  - artdaq::BoardReaderCore, [82](#)
- GetDroppedDataAddress
  - artdaq::SharedMemoryEventManager, [354](#)
- GetEndOfData
  - artdaq::FragmentStoreElement, [235](#)
- GetEventHeader
  - ArtdaqGlobalsService, [54](#)
  - ArtdaqSharedMemoryService, [65](#)
  - ArtdaqSharedMemoryServiceInterface, [67](#)
- getEventHeader
  - art::ShmemWrapper, [364](#)
  - artdaq::ListenTransferWrapper, [256](#)
  - artdaq::TransferWrapper, [422](#)
- getFileName
  - artdaq::art\_config\_file, [48](#)
- GetFragmentCount
  - artdaq::SharedMemoryEventManager, [354](#)
- GetFragmentCountInBuffer
  - artdaq::SharedMemoryEventManager, [354](#)
- GetFragmentsProcessed
  - artdaq::BoardReaderCore, [83](#)
- GetHeldTokenCount
  - artdaq::RoutingManagerPolicy, [336](#)
- GetIPOfInterface
  - TCPConnect.hh, [447](#)
- GetInterfaceForNetwork
  - TCPConnect.hh, [446](#)
- GetLockedBufferCount
  - artdaq::SharedMemoryEventManager, [355](#)
- GetMFilteration\_
  - artdaq::Globals, [244](#)
- GetMFModuleName\_
  - artdaq::Globals, [244](#)
- GetMaxNumberOfTokens
  - artdaq::RoutingManagerPolicy, [336](#)
- GetMessage
  - artdaq::detail::RequestMessage, [300](#)
- GetMulticastOutputAddress
  - artdaq::PortManager, [279](#)
- GetMulticastTransferGroupAddress
  - artdaq::PortManager, [280](#)
- GetMulticastTransferPort
  - artdaq::PortManager, [280](#)
- GetMyId
  - ArtdaqGlobalsService, [54](#)
  - ArtdaqSharedMemoryService, [65](#)
  - ArtdaqSharedMemoryServiceInterface, [67](#)
- getNext
  - anonymous\_namespace{genToArt.cc}::Throttled-Generator, [396](#)
  - artdaq::CommandableFragmentGenerator, [124](#)
  - artdaq::GenericFragmentSimulator, [239](#)
- getNext\_
  - artdaq::CommandableFragmentGenerator, [124](#)
  - artdaqtest::CommandableFragmentGeneratorTest, [133](#)
- GetNextRequest
  - artdaq::RequestBuffer, [296](#)
- GetNextSequenceID
  - artdaq::FragmentBuffer, [225](#)
  - artdaq::RoutingManagerPolicy, [336](#)
- GetOpenEventCount
  - artdaq::SharedMemoryEventManager, [355](#)
- getPackageBuildInfo
  - artdaq::GetPackageBuildInfo, [242](#)
- getParam
  - artdaq::cmd\_, [106](#), [107](#)
- getPartitionNumber\_
  - artdaq::Globals, [244](#)
- GetPendingEventCount
  - artdaq::SharedMemoryEventManager, [355](#)
- GetQueueCapacity
  - ArtdaqGlobalsService, [54](#)
  - ArtdaqSharedMemoryService, [65](#)



- ArtdaqSharedMemoryServiceInterface, 68
- GetQueueSize
  - ArtdaqGlobalsService, 54
  - ArtdaqSharedMemoryService, 66
  - ArtdaqSharedMemoryServiceInterface, 68
- getReceivedTokenCount
  - artdaq::TokenReceiver, 405
- GetReceiverCount
  - artdaq::RoutingManagerPolicy, 336
- GetReceiverThreadActive
  - artdaq::BoardReaderCore, 83
- GetRemainingRoutingTableEntries
  - artdaq::DataSenderManager, 194
  - artdaq::TableReceiver, 391
- GetRequestBuffer
  - artdaq::CommandableFragmentGenerator, 124
  - artdaq::GenToBufferTest, 242
- GetRequestMessageGroupAddress
  - artdaq::PortManager, 280
- GetRequestMessagePort
  - artdaq::PortManager, 280
- GetRequestMode
  - artdaq::RequestSender, 309
- GetRequestTime
  - artdaq::RequestBuffer, 296
- GetRequests
  - artdaq::RequestBuffer, 296
- GetRouteForSequenceID
  - artdaq::RoutingManagerPolicy, 336
- GetRoutingAckPort
  - artdaq::PortManager, 281
- GetRoutingMode
  - artdaq::RoutingManagerPolicy, 337
- GetRoutingTableEntry
  - artdaq::TableReceiver, 391
- GetRoutingTableEntryCount
  - artdaq::DataSenderManager, 194
  - artdaq::TableReceiver, 392
- GetRoutingTableGroupAddress
  - artdaq::PortManager, 281
- GetRoutingTablePort
  - artdaq::PortManager, 281
- GetRoutingTokenPort
  - artdaq::PortManager, 281
- GetSenderThreadActive
  - artdaq::BoardReaderCore, 83
- GetSentMessageCount
  - artdaq::RequestSender, 309
- GetSentSequenceIDCount
  - artdaq::DataSenderManager, 194
- GetSentTokenCount
  - artdaq::TokenSender, 408
- GetSentWindowList
  - artdaq::FragmentBuffer, 225
- getSharedMemoryEventManager
  - artdaq::DataReceiverManager, 191
- GetStatus
  - artdaq::CommanderInterface, 137
- GetSubrunForSequenceID
  - artdaq::SharedMemoryEventManager, 355
- GetTCPSocketTransferPort
  - artdaq::PortManager, 282
- getTimestamp
  - artdaqtest::CommandableFragmentGeneratorTest, 133
  - artdaqtest::FragmentBufferTestGenerator, 229
- GetTokensUsedSinceLastUpdate
  - artdaq::RoutingManagerPolicy, 337
- GetUnidentifiedInstanceName
  - ArtdaqFragmentNamingServiceInterface, 51
- GetXMLRPCPort
  - artdaq::PortManager, 282
- hasMoreData
  - art::ArtdaqInputHelper, 56
- header
  - artdaq::detail::RequestHeader, 299
  - artdaq::detail::RequestPacket, 303
- host\_map
  - artdaq::HostMap::Config, 159
- in\_run\_failure
  - artdaq::Commandable, 114
- InRunExit
  - artdaq::Commandable, 115
- incSlot
  - artdaq::detail::FragCounter, 216
- init\_
  - artdaq::init\_, 251
- initialize
  - artdaq::BoardReaderCore, 83
  - artdaq::Commandable, 115
  - artdaq::DataLoggerCore, 182
  - artdaq::DataReceiverCore, 185
  - artdaq::DispatcherCore, 202
  - artdaq::EventBuilderCore, 213
  - artdaq::RoutingManagerCore, 328
- initializeDataReceiver
  - artdaq::DataReceiverCore, 186
- IntToTaskType
  - artdaq::detail, 43
- is\_consistent
  - Timeout, 402
- isRunning
  - artdaq::AutodetectTransfer, 70
  - artdaq::BundleTransfer, 94
  - artdaq::MulticastTransfer, 265
  - artdaq::NullTransfer, 271
  - artdaq::RequestBuffer, 297

- artdaq::RequestReceiver, [306](#)
- artdaq::RTIDDSTransfer, [346](#)
- artdaq::ShmemTransfer, [361](#)
- artdaq::TCPSocketTransfer, [393](#)
- artdaq::TransferInterface, [414](#)
- isValid
  - artdaq::detail::RequestHeader, [299](#)
  - artdaq::detail::RequestPacket, [303](#)
- joinThreads
  - artdaq::CommandableFragmentGenerator, [124](#)
- kErrorNotRequiringException
  - artdaq::TransferInterface, [413](#)
- kReceive
  - artdaq::TransferInterface, [413](#)
- kSend
  - artdaq::TransferInterface, [413](#)
- kSuccess
  - artdaq::TransferInterface, [413](#)
- kTimeout
  - artdaq::TransferInterface, [413](#)
- legal\_commands
  - artdaq::Commandable, [115](#)
- legal\_commands\_
  - artdaq::legal\_commands\_, [255](#)
- ListenTransferWrapper
  - artdaq::ListenTransferWrapper, [256](#)
- makeCommandableFragmentGenerator
  - artdaq, [39](#)
- MakeCommanderPlugin
  - artdaq, [39](#)
- makeFunc\_t
  - artdaq, [34](#)
- MakeHostMap
  - artdaq, [39](#)
- MakeHostMapPset
  - artdaq, [40](#)
- makeRoutingManagerPolicy
  - artdaq, [40](#)
- MakeTransferPlugin
  - artdaq, [40](#)
- max\_event\_size\_bytes
  - artdaq::SharedMemoryEventManager::Config, [171](#)
- max\_fragment\_size\_bytes
  - artdaq::SharedMemoryEventManager::Config, [172](#)
- max\_request\_count
  - artdaq::detail::RequestMessage, [300](#)
- MessHead, [259](#)
  - MessType, [259](#)
- MessType
  - MessHead, [259](#)
- meta\_command\_
  - artdaq::meta\_command\_, [260](#)
- metaCommand
  - artdaq::BoardReaderCore, [84](#)
  - artdaq::CommandableFragmentGenerator, [124](#)
- metricsReportingInstanceName
  - artdaq::CommandableFragmentGenerator, [126](#)
- mfplugins::ELArtdaqMetric, [205](#)
  - ELArtdaqMetric, [206](#)
  - fillPrefix, [206](#)
  - fillUsrMsg, [206](#)
  - routePayload, [206](#)
- mfplugins::ELArtdaqMetric::Config, [152](#)
  - showDebug, [153](#)
  - showError, [153](#)
  - showInfo, [153](#)
  - showWarning, [154](#)
- minCount
  - artdaq::detail::FragCounter, [217](#)
- MissingDataCheck
  - artdaq::MissingDataCheck, [263](#)
- MulticastTransfer
  - artdaq::MulticastTransfer, [264](#)
- NO\_RANK\_INFO
  - artdaq::TransferInterface, [413](#)
- nSlots
  - artdaq::detail::FragCounter, [217](#)
- NoOp\_policy\_t::DataFlowMode, [174](#)
- NoOp\_policy\_t::DataFlowMode\_id, [176](#)
- NoOp\_policy\_t::RequestBasedEventBuilding, [292](#)
- NoOp\_policy\_t::RequestBasedEventBuilding\_id, [294](#)
- NoOp\_policy\_t::Simple, [368](#)
- NoOp\_policy\_t::Simple\_id, [370](#)
- NoOpPolicy
  - artdaq::NoOpPolicy, [269](#)
- Normal
  - artdaq::detail, [42](#)
- NullTransfer
  - artdaq::NullTransfer, [271](#)
- numFragIDs
  - anonymous\_namespace{genToArt.cc}::Throttled-Generator, [396](#)
- on\_data\_available
  - artdaq::RTIDDS::OctetsListener, [273](#)
- openFile
  - art::ArtdaqOutput, [62](#)
- operator<<
  - artdaq::detail, [44](#)
- operator()
  - art::RootOutputConfig::KeysToIgnore, [252](#)
- operator=
  - art::ArtdaqInputHelper, [56](#)
  - artdaq::BoardReaderApp, [80](#)
  - artdaq::BoardReaderCore, [84](#)



- artdaq::Commandable, [115](#)
- artdaq::CommanderInterface, [137](#)
- artdaq::DataLoggerApp, [180](#)
- artdaq::DataLoggerCore, [182](#)
- artdaq::DataReceiverCore, [186](#)
- artdaq::DispatcherApp, [199](#)
- artdaq::DispatcherCore, [203](#)
- artdaq::EventBuilderApp, [211](#)
- artdaq::EventBuilderCore, [213](#)
- artdaq::RandomDelayFilter, [288](#)
- artdaq::RequestSender, [309](#)
- artdaq::RoutingManagerApp, [326](#)
- artdaq::RoutingManagerCore, [329](#)
- artdaq::StatisticsHelper, [379](#)
- artdaq::TokenSender, [408](#)
- artdaq::TransferInterface, [414](#)
- pause
  - artdaq::BoardReaderCore, [84](#)
  - artdaq::Commandable, [115](#)
  - artdaq::CommandableFragmentGenerator, [126](#)
  - artdaq::DataReceiverCore, [186](#)
  - artdaq::RoutingManagerCore, [329](#)
- pause\_
  - artdaq::pause\_, [276](#)
- PauseCmd
  - artdaq::CommandableFragmentGenerator, [126](#)
- pauseNoMutex
  - artdaq::CommandableFragmentGenerator, [127](#)
- PreferSameHost\_policy\_t::DataFlowMode, [175](#)
- PreferSameHost\_policy\_t::DataFlowMode\_id, [176](#)
- PreferSameHost\_policy\_t::LargeMinimumParticipants, [254](#)
- PreferSameHost\_policy\_t::LargeMinimumParticipants\_id, [254](#)
- PreferSameHost\_policy\_t::ManyMissingParticipants, [257](#)
- PreferSameHost\_policy\_t::ManyMissingParticipants\_id, [258](#)
- PreferSameHost\_policy\_t::MinimumParticipants, [261](#)
- PreferSameHost\_policy\_t::MinimumParticipants\_id, [262](#)
- PreferSameHost\_policy\_t::RequestBasedEventBuilding, [292](#)
- PreferSameHost\_policy\_t::RequestBasedEventBuilding\_id, [294](#)
- PreferSameHost\_policy\_t::Simple, [368](#)
- PreferSameHost\_policy\_t::Simple\_id, [370](#)
- PreferSameHostPolicy
  - artdaq::PreferSameHostPolicy, [283](#)
- PrintBuildInfo
  - artdaq::PrintBuildInfo, [286](#)
- printMode\_
  - artdaq::FragmentBuffer, [225](#)
- printProcessHistoryID
  - art, [27](#)
- printProcessMap
  - art, [27](#)
- process\_cmd\_line
  - anonymous\_namespace{genToArt.cc}, [23](#)
- process\_data
  - anonymous\_namespace{genToArt.cc}, [23](#)
- produce
  - artdaq::BuildInfo, [92](#)
- push
  - artdaq::RequestBuffer, [297](#)
- RANDOM
  - artdaq::GenericFragmentSimulator, [239](#)
- RECV\_SUCCESS
  - artdaq::TransferInterface, [413](#)
- RECV\_TIMEOUT
  - artdaq::TransferInterface, [413](#)
- RTIDDS
  - artdaq::RTIDDS, [344](#)
- RTIDDSTransfer
  - artdaq::RTIDDSTransfer, [345](#)
- RandomDelayFilter
  - artdaq::RandomDelayFilter, [287](#)
- readFile
  - art::ArtdaqInputHelper, [56](#)
- readNext
  - art::ArtdaqInputHelper, [58](#)
- ReadObjectAny
  - art, [28](#)
- readyToReport
  - artdaq::StatisticsHelper, [379](#)
- receive\_fragments
  - artdaq::BoardReaderCore, [84](#)
- ReceiveEvent
  - ArtdaqGlobalsService, [55](#)
  - ArtdaqSharedMemoryService, [66](#)
  - ArtdaqSharedMemoryServiceInterface, [68](#)
- receiveFragment
  - artdaq::AutodetectTransfer, [71](#)
  - artdaq::BundleTransfer, [94](#)
  - artdaq::MulticastTransfer, [265](#)
  - artdaq::NullTransfer, [271](#)
  - artdaq::RTIDDSTransfer, [346](#)
  - artdaq::ShmemTransfer, [361](#)
  - artdaq::TransferInterface, [414](#)
- receiveFragmentData
  - artdaq::AutodetectTransfer, [71](#)
  - artdaq::BundleTransfer, [94](#)
  - artdaq::MulticastTransfer, [265](#)
  - artdaq::NullTransfer, [272](#)
  - artdaq::ShmemTransfer, [362](#)
  - artdaq::TCPSocketTransfer, [394](#)
  - artdaq::TransferInterface, [416](#)
- receiveFragmentFromDDS

- artdaq::RTIDDS::OctetsListener, 274
- receiveFragmentHeader
  - artdaq::AutodetectTransfer, 71
  - artdaq::BundleTransfer, 94
  - artdaq::MulticastTransfer, 265
  - artdaq::NullTransfer, 272
  - artdaq::ShmemTransfer, 362
  - artdaq::TCPSocketTransfer, 394
  - artdaq::TransferInterface, 416
- receiveInitMessage
  - art::ShmemWrapper, 364
  - artdaq::ListenTransferWrapper, 257
  - artdaq::TransferWrapper, 422
- receiveMessage
  - art::ShmemWrapper, 364
  - artdaq::ListenTransferWrapper, 257
  - artdaq::TransferWrapper, 422
- receiveMessages
  - art::ShmemWrapper, 365
  - artdaq::ListenTransferWrapper, 257
  - artdaq::TransferWrapper, 422
- ReceiveReturnCode
  - artdaq::TransferInterface, 413
- ReconfigureArt
  - artdaq::SharedMemoryEventManager, 356
- recvFragment
  - artdaq::FragmentReceiverManager, 231
- register\_monitor
  - artdaq::Commandable, 116
  - artdaq::DispatcherApp, 199
  - artdaq::DispatcherCore, 203
- register\_monitor\_
  - artdaq::register\_monitor\_, 289
- reinit\_
  - artdaq::reinit\_, 290
- reinitialize
  - artdaq::BoardReaderCore, 85
  - artdaq::Commandable, 116
  - artdaq::DataReceiverCore, 186
  - artdaq::RoutingManagerCore, 329
- RemoveRequest
  - artdaq::RequestBuffer, 297
  - artdaq::RequestSender, 309
- RemoveRoutingTableEntry
  - artdaq::DataSenderManager, 195
  - artdaq::TableReceiver, 392
- report
  - artdaq::BoardReaderApp, 80
  - artdaq::BoardReaderCore, 85
  - artdaq::Commandable, 116
  - artdaq::CommandableFragmentGenerator, 127
  - artdaq::DataLoggerApp, 180
  - artdaq::DataReceiverCore, 188
  - artdaq::DispatcherApp, 200
  - artdaq::EventBuilderApp, 211
  - artdaq::RoutingManagerApp, 326
  - artdaq::RoutingManagerCore, 329
- report\_
  - artdaq::report\_, 291
- ReportCmd
  - artdaq::CommandableFragmentGenerator, 127
- reportSpecific
  - artdaq::CommandableFragmentGenerator, 127
- RequestBasedEventBuilding
  - artdaq::detail, 42
- request\_mode
  - artdaq::FragmentBuffer, 225
  - artdaq::FragmentBuffer::Config, 168
- RequestBuffer
  - artdaq::RequestBuffer, 295
- RequestMessageMode
  - artdaq::detail, 42
- RequestModeToString
  - artdaq::detail::RoutingRequest, 342
- RequestPacket
  - artdaq::detail::RequestPacket, 303
- RequestReceiver
  - artdaq::RequestReceiver, 305
- RequestSender
  - artdaq::RequestSender, 308
- RequestSender\_test::Construct, 173
- RequestSender\_test::Construct\_id, 174
- RequestSender\_test::Requests, 306
- RequestSender\_test::Requests\_id, 307
- RequestSenderModule
  - artdaq::RequestSenderModule, 311
- RequestsInFlight
  - artdaq::RequestSender, 309
- Reset
  - artdaq::FragmentBuffer, 226
- ResolveHost
  - TCPConnect.hh, 447
- respondToCloseInputFile
  - art::ArtdaqOutput, 62
- respondToCloseOutputFiles
  - art::ArtdaqOutput, 62
- resume
  - artdaq::BoardReaderCore, 85
  - artdaq::Commandable, 116
  - artdaq::CommandableFragmentGenerator, 128
  - artdaq::DataReceiverCore, 188
  - artdaq::RoutingManagerCore, 330
- resume\_
  - artdaq::resume\_, 312
- ResumeCmd
  - artdaq::CommandableFragmentGenerator, 128
- returnCode
  - artdaq::TransferTest, 420

- Role
  - artdaq::TransferInterface, [413](#)
- role
  - artdaq::TransferInterface, [416](#)
- rollover\_subrun
  - artdaq::DataReceiverCore, [188](#)
- rollover\_subrun\_
  - artdaq::rollover\_subrun\_, [314](#)
- rolloverSubrun
  - artdaq::SharedMemoryEventManager, [356](#)
- RootNetOutput
  - art::RootNetOutput, [317](#)
- RoundRobin\_policy\_t::DataFlowMode, [175](#)
- RoundRobin\_policy\_t::DataFlowMode\_id, [176](#)
- RoundRobin\_policy\_t::LargeMinimumParticipants, [253](#)
- RoundRobin\_policy\_t::LargeMinimumParticipants\_id, [254](#)
- RoundRobin\_policy\_t::ManyMissingParticipants, [258](#)
- RoundRobin\_policy\_t::ManyMissingParticipants\_id, [258](#)
- RoundRobin\_policy\_t::MinimumParticipants, [261](#)
- RoundRobin\_policy\_t::MinimumParticipants\_id, [262](#)
- RoundRobin\_policy\_t::RequestBasedEventBuilding, [293](#)
- RoundRobin\_policy\_t::RequestBasedEventBuilding\_id, [294](#)
- RoundRobin\_policy\_t::Simple, [369](#)
- RoundRobin\_policy\_t::Simple\_id, [369](#)
- RoundRobin\_policy\_t::VerifyRMPSHaredPtr, [423](#)
- RoundRobin\_policy\_t::VerifyRMPSHaredPtr\_id, [424](#)
- RoundRobinPolicy
  - artdaq::RoundRobinPolicy, [320](#)
- routePayload
  - mfplugins::ELArtdaqMetric, [206](#)
- routing\_table\_config
  - artdaq::DataSenderManager::Config, [162](#)
- RoutingManagerApp
  - artdaq::RoutingManagerApp, [323](#)
- RoutingManagerMode
  - artdaq::detail, [42](#)
- routingManagerModeToString
  - artdaq::detail::RoutingManagerModeConverter, [332](#)
- RoutingManagerPolicy
  - artdaq::RoutingManagerPolicy, [334](#)
- RoutingPacketEntry
  - artdaq::detail::RoutingPacketEntry, [338](#)
- RoutingPacketHeader
  - artdaq::detail::RoutingPacketHeader, [339](#)
- RoutingRequest
  - artdaq::detail::RoutingRequest, [341](#), [342](#)
- RoutingTokenSendsEnabled
  - artdaq::TokenSender, [408](#)
- run\_number
  - artdaq::CommandableFragmentGenerator, [128](#)
- run\_server
  - artdaq::CommanderInterface, [137](#)
- runArtdaqApp
  - artdaq::artdaqapp, [49](#)
- runID
  - artdaq::SharedMemoryEventManager, [356](#)
- runTest
  - artdaq::TransferTest, [420](#)
- running\_sources
  - artdaq::DataReceiverManager, [192](#)
  - artdaq::FragmentReceiverManager, [231](#)
- saveMemoryObjectThreshold
  - art::RootDAQOut::Config, [157](#)
- seedAndRandom\_
  - artdaq::Globals, [244](#)
- send\_event\_table
  - artdaq::RoutingManagerCore, [330](#)
- send\_fragments
  - artdaq::BoardReaderCore, [85](#)
- send\_init
  - artdaq::CommanderInterface, [138](#)
  - artdaq::xmlrpc\_commander, [434](#)
- send\_init\_message
  - art::ArtdaqOutput, [62](#)
- send\_legal\_commands
  - artdaq::CommanderInterface, [138](#)
  - artdaq::xmlrpc\_commander, [434](#)
- send\_meta\_command
  - artdaq::CommanderInterface, [138](#)
  - artdaq::xmlrpc\_commander, [435](#)
- send\_metric
  - swig\_artdaq, [383](#), [384](#)
- send\_pause
  - artdaq::CommanderInterface, [139](#)
  - artdaq::xmlrpc\_commander, [435](#)
- send\_rate\_metric
  - swig\_artdaq, [384](#), [386](#)
- send\_register\_monitor
  - artdaq::CommanderInterface, [139](#)
  - artdaq::xmlrpc\_commander, [435](#)
- send\_reinit
  - artdaq::CommanderInterface, [139](#)
  - artdaq::xmlrpc\_commander, [436](#)
- send\_report
  - artdaq::CommanderInterface, [141](#)
  - artdaq::xmlrpc\_commander, [436](#)
- send\_resume
  - artdaq::CommanderInterface, [141](#)
  - artdaq::xmlrpc\_commander, [436](#)
- send\_rollover\_subrun
  - artdaq::CommanderInterface, [141](#)
  - artdaq::xmlrpc\_commander, [438](#)
- send\_shutdown
  - artdaq::CommanderInterface, [143](#)
  - artdaq::xmlrpc\_commander, [438](#)
- send\_soft\_init

- artdaq::CommanderInterface, 143
- artdaq::xmlrpc\_commander, 438
- send\_start
  - artdaq::CommanderInterface, 143
  - artdaq::xmlrpc\_commander, 440
- send\_status
  - artdaq::CommanderInterface, 145
  - artdaq::xmlrpc\_commander, 440
- send\_stop
  - artdaq::CommanderInterface, 145
  - artdaq::xmlrpc\_commander, 440
- send\_sum\_metric
  - swig\_artdaq, 386
- send\_trace\_get
  - artdaq::CommanderInterface, 145
  - artdaq::xmlrpc\_commander, 441
- send\_trace\_set
  - artdaq::CommanderInterface, 146
  - artdaq::xmlrpc\_commander, 441
- send\_unregister\_monitor
  - artdaq::CommanderInterface, 146
  - artdaq::xmlrpc\_commander, 441
- sendEmptyFragment
  - artdaq::FragmentBuffer, 226
- sendEmptyFragments
  - artdaq::FragmentBuffer, 226
- sendFragment
  - artdaq::DataSenderManager, 195
- SendMessage
  - art::ArtdaqOutput, 63
  - art::RootNetOutput, 317
  - art::TransferOutput, 419
- SendRequest
  - artdaq::RequestSender, 309
- SendRoutingToken
  - artdaq::TokenSender, 408
- set\_exception
  - artdaq::CommandableFragmentGenerator, 128
- setEnabledIds
  - artdaqtest::CommandableFragmentGeneratorTest, 133
- SetEndOfData
  - artdaq::FragmentStoreElement, 236
- setFireCount
  - artdaqtest::CommandableFragmentGeneratorTest, 134
- SetMFIteration\_
  - artdaq::Globals, 244
- SetMFModuleName\_
  - artdaq::Globals, 246
- setMode
  - artdaq::detail::RequestMessage, 300
- setOverwrite
  - artdaq::SharedMemoryEventManager, 356
- setRank
  - artdaq::detail::RequestMessage, 302
- SetRequestBuffer
  - artdaq::CommandableFragmentGenerator, 128
  - artdaq::FragmentBuffer, 226
- SetRequestMode
  - artdaq::RequestSender, 310
- setRequestMode
  - artdaq::SharedMemoryEventManager, 356
- SetRunNumber
  - artdaq::RequestReceiver, 306
  - artdaq::RequestSender, 310
  - artdaq::TokenSender, 408
- setRunNumber
  - artdaq::detail::RequestMessage, 302
  - artdaq::TokenReceiver, 405
- setRunning
  - artdaq::RequestBuffer, 297
- setSlot
  - artdaq::detail::FragCounter, 217
- SetStartTransitionTimeout
  - artdaq::BoardReaderCore, 86
- setStatsHelper
  - artdaq::TokenReceiver, 405
- setTimestamp
  - artdaqtest::CommandableFragmentGeneratorTest, 134
  - artdaqtest::FragmentBufferTestGenerator, 229
- SharedMemoryEventManager
  - artdaq::SharedMemoryEventManager, 351
- ShmemTransfer
  - artdaq::ShmemTransfer, 361
- ShmemWrapper
  - art::ShmemWrapper, 364
- should\_stop
  - artdaq::CommandableFragmentGenerator, 129
- showDebug
  - mfplugins::ELArtdaqMetric::Config, 153
- showError
  - mfplugins::ELArtdaqMetric::Config, 153
- showInfo
  - mfplugins::ELArtdaqMetric::Config, 153
- showWarning
  - mfplugins::ELArtdaqMetric::Config, 154
- shutdown
  - artdaq::BoardReaderCore, 86
  - artdaq::Commandable, 117
  - artdaq::DataReceiverCore, 188
  - artdaq::RoutingManagerCore, 330
- shutdown\_
  - artdaq::shutdown\_, 366
- ShutdownArtProcesses
  - artdaq::SharedMemoryEventManager, 358
- size

- artdaq::detail::RequestMessage, 302
  - artdaq::FragmentStoreElement, 236
  - artdaq::RequestBuffer, 297
- slotCount
  - artdaq::DataReceiverManager, 192
  - artdaq::DataSenderManager, 195
  - artdaq::detail::FragCounter, 217
  - artdaq::FragmentReceiverManager, 231
- soft\_init\_
  - artdaq::soft\_init\_, 373
- soft\_initialize
  - artdaq::BoardReaderCore, 86
  - artdaq::Commandable, 117
  - artdaq::DataReceiverCore, 189
  - artdaq::RoutingManagerCore, 330
- source
  - art::Config, 155
- source\_rank
  - artdaq::TransferInterface, 417
- start
  - anonymous\_namespace{genToArt.cc}::Throttled-Generator, 396
  - artdaq::BoardReaderCore, 86
  - artdaq::Commandable, 117
  - artdaq::CommandableFragmentGenerator, 129
  - artdaq::DataReceiverCore, 189
  - artdaq::GenToBufferTest, 242
  - artdaq::RoutingManagerCore, 331
- start\_
  - artdaq::start\_, 376
- StartArtProcess
  - artdaq::SharedMemoryEventManager, 358
- StartCmd
  - artdaq::CommandableFragmentGenerator, 129
- startRun
  - artdaq::SharedMemoryEventManager, 358
- starting\_fragment\_id
  - artdaq::GenericFragmentSimulator::Config, 158
- statsRollingWindowHasMoved
  - artdaq::StatisticsHelper, 379
- status
  - artdaq::Commandable, 118
- status\_
  - artdaq::status\_, 380
- stop
  - anonymous\_namespace{genToArt.cc}::Throttled-Generator, 396
  - artdaq::BoardReaderCore, 87
  - artdaq::Commandable, 118
  - artdaq::CommandableFragmentGenerator, 129
  - artdaq::DataReceiverCore, 189
  - artdaq::RoutingManagerCore, 331
- stop\_
  - artdaq::stop\_, 381
- StopCmd
  - artdaq::CommandableFragmentGenerator, 129
- stopNoMutex
  - artdaq::CommandableFragmentGenerator, 131
- stopRequestReception
  - artdaq::RequestReceiver, 306
- stopTokenReception
  - artdaq::TokenReceiver, 405
- stringToRoutingManagerMode
  - artdaq::detail::RoutingManagerModeConverter, 332
- StringToTaskType
  - artdaq::detail, 44
- subrun\_number
  - artdaq::CommandableFragmentGenerator, 131
- swig\_artdaq, 382
  - send\_metric, 383, 384
  - send\_rate\_metric, 384, 386
  - send\_sum\_metric, 386
  - swig\_artdaq, 383
  - swig\_artdaq, 383
  - write\_debug, 388
  - write\_error, 388
  - write\_info, 388
  - write\_trace, 388
  - write\_warning, 388
- TCP\_listen\_fd
  - TCP\_listen\_fd.hh, 445
- TCP\_listen\_fd.hh
  - TCP\_listen\_fd, 445
- TCPConnect
  - TCPConnect.hh, 448
- TCPConnect.hh
  - AutodetectPrivateInterface, 446
  - GetIPOfInterface, 447
  - GetInterfaceForNetwork, 446
  - ResolveHost, 447
  - TCPConnect, 448
- TCPSocketTransfer
  - artdaq::TCPSocketTransfer, 393
- TableReceiver
  - artdaq::TableReceiver, 391
- TaskTypeToString
  - artdaq::detail, 44
- TestReceiverPause
  - artdaqtest::BrokenTransferTest, 88
- TestReceiverReconnect
  - artdaqtest::BrokenTransferTest, 88
- TestSenderPause
  - artdaqtest::BrokenTransferTest, 88
- TestSenderReconnect
  - artdaqtest::BrokenTransferTest, 88
- ThrottledGenerator

- anonymous\_namespace{genToArt.cc}::Throttled-Generator, 396
- Timeout, 398
  - add\_periodic, 399
  - add\_relative, 401
  - cancel\_timeout, 401
  - copy\_in\_timeout, 401
  - get\_next\_expired\_timeout, 402
  - get\_next\_timeout\_delay, 402
  - get\_next\_timeout\_msdlly, 402
  - is\_consistent, 402
  - Timeout, 399
- timeout
  - artdaq::CommandableFragmentGenerator, 131
- Timeout::timeoutspec, 403
- timestamp
  - artdaq::CommandableFragmentGenerator, 131
- TokenReceiver
  - artdaq::TokenReceiver, 404
- TokenSender
  - artdaq::TokenSender, 407
- trace\_get\_
  - artdaq::trace\_get\_, 409
- trace\_set\_
  - artdaq::trace\_set\_, 410
- transfer\_fragment\_min\_blocking\_mode
  - artdaq::AutodetectTransfer, 72
  - artdaq::BundleTransfer, 96
  - artdaq::MulticastTransfer, 266
  - artdaq::NullTransfer, 272
  - artdaq::RTIDDSSTransfer, 346
  - artdaq::ShmemTransfer, 362
  - artdaq::TCPSocketTransfer, 394
  - artdaq::TransferInterface, 417
- transfer\_fragment\_reliable\_mode
  - artdaq::AutodetectTransfer, 72
  - artdaq::BundleTransfer, 96
  - artdaq::MulticastTransfer, 266
  - artdaq::NullTransfer, 272
  - artdaq::RTIDDSSTransfer, 346
  - artdaq::ShmemTransfer, 363
  - artdaq::TCPSocketTransfer, 395
  - artdaq::TransferInterface, 417
- transfer\_fragment\_reliable\_mode\_via\_DDS\_
  - artdaq::RTIDDS, 344
- TransferInput
  - art, 27
- TransferInterface
  - artdaq::TransferInterface, 413
- TransferOutput
  - art::TransferOutput, 419
- TransferTest
  - artdaq::TransferTest, 420
- TransferWrapper
  - artdaq::TransferWrapper, 421
- uniqueLabel
  - artdaq::TransferInterface, 417
- unregister\_monitor
  - artdaq::Commandable, 118
  - artdaq::DispatcherApp, 200
  - artdaq::DispatcherCore, 203
- unregister\_monitor\_
  - artdaq::unregister\_monitor\_, 423
- UpdateArtConfiguration
  - artdaq::SharedMemoryEventManager, 358
- UpdateConfiguration
  - artdaq::PortManager, 282
- waitForDataBufferReady
  - artdaq::FragmentBuffer, 227
- WaitForRequests
  - artdaq::RequestBuffer, 298
- write
  - art::ArtdaqOutput, 63
- write\_debug
  - swig\_artdaq, 388
- write\_error
  - swig\_artdaq, 388
- write\_info
  - swig\_artdaq, 388
- write\_trace
  - swig\_artdaq, 388
- write\_warning
  - swig\_artdaq, 388
- writeDataProducts
  - art::ArtdaqOutput, 63
- WriteFragmentHeader
  - artdaq::SharedMemoryEventManager, 358
- writeRun
  - art::ArtdaqOutput, 63
- writeSubRun
  - art::ArtdaqOutput, 63
- xmlrpc\_commander
  - artdaq::xmlrpc\_commander, 434