

SimulationInterfaceMacro

5.1

Generated by Doxygen 1.8.14

Contents

1	Module Index	1
1.1	Modules	1
2	Namespace Index	3
2.1	Namespace List	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Data Structure Index	7
4.1	Data Structures	7
5	File Index	9
5.1	File List	9
6	Module Documentation	11
6.1	Models	11
6.1.1	Detailed Description	11
6.2	Utils	12
6.2.1	Detailed Description	12
6.3	SimInterface	13
6.3.1	Detailed Description	15
6.3.2	Macro Definition Documentation	15
6.3.2.1	CLASS	15
6.3.2.2	ER7_UTILS_ALWAYS_INLINE	16
6.3.2.3	ER7_UTILS_RESTRICT	16

6.3.2.4	ER7_UTILS_UNUSED	16
6.3.2.5	JEOD_ATTRIBUTES_POINTER_TYPE [1/2]	16
6.3.2.6	JEOD_ATTRIBUTES_POINTER_TYPE [2/2]	16
6.3.2.7	JEOD_ATTRIBUTES_SIM_ENGINE_HEADER	16
6.3.2.8	JEOD_ATTRIBUTES_TYPE [1/2]	17
6.3.2.9	JEOD_ATTRIBUTES_TYPE [2/2]	17
6.3.2.10	JEOD_CLASS_ESTABLISH_FRIENDS	17
6.3.2.11	JEOD_CLASS_ESTABLISH_FRIENDS1	17
6.3.2.12	JEOD_CLASS_ESTABLISH_FRIENDS2	17
6.3.2.13	JEOD_CLASS_ESTABLISH_FRIENDS3	18
6.3.2.14	JEOD_DECLARE_SIM_INTERFACES	18
6.3.2.15	JEOD_INTPTR_T	18
6.3.2.16	JEOD_MAKE_SIM_INTERFACES	18
6.3.2.17	JEOD_PTRDIFF_T	19
6.3.2.18	JEOD_SIM_INTEGRATOR_ENUM	19
6.3.2.19	JEOD_SIM_INTEGRATOR_FORWARD	19
6.3.2.20	JEOD_SIM_INTEGRATOR_POINTER_TYPE [1/2]	19
6.3.2.21	JEOD_SIM_INTEGRATOR_POINTER_TYPE [2/2]	20
6.3.2.22	JEOD_SIM_INTEGRATOR_SIM_ENGINE_HEADER	20
6.3.2.23	JEOD_SIZE_T	20
6.3.2.24	JEOD_UINTPTR_T	20
6.3.2.25	JEOD_UNUSED	20
6.3.2.26	JEODVMACRO_SEQ_COUNT	21
6.3.2.27	JEODVMACRO	22
6.3.2.28	JEODVMACRO_GET_COUNT	22
6.3.2.29	JEODVMACRO_REVSEQ_COUNT	22
6.3.2.30	JEODVMACRO_SELECT_FUNC	23
6.3.2.31	JEODVMACRO_SELECT_FUNC_CAT	23
6.3.2.32	JEODVMACRO_VA_SIZE	23
6.3.2.33	MAKE_MESSAGE_CODE	23
6.3.2.34	MAX_MSG_SIZE	23
6.3.2.35	PATH	24
6.3.3	Variable Documentation	24
6.3.3.1	trick_curr_integ	24
6.3.3.2	trick_MM [1/4]	24
6.3.3.3	trick_MM [2/4]	24
6.3.3.4	trick_MM [3/4]	24
6.3.3.5	trick_MM [4/4]	24

7	Namespace Documentation	25
7.1	er7_utils Namespace Reference	25
7.1.1	Detailed Description	25
7.2	jeod Namespace Reference	25
7.2.1	Detailed Description	26
7.3	Trick Namespace Reference	26
7.3.1	Detailed Description	26
8	Data Structure Documentation	27
8.1	jeod::JeodTrickMemoryInterface::AllocationMapEntry Struct Reference	27
8.1.1	Detailed Description	27
8.1.2	Constructor & Destructor Documentation	27
8.1.2.1	AllocationMapEntry()	27
8.1.3	Field Documentation	28
8.1.3.1	is_array	28
8.1.3.2	nelements	28
8.1.3.3	typeid_info	28
8.2	jeod::BasicJeodTrickSimInterface Class Reference	29
8.2.1	Detailed Description	30
8.2.2	Constructor & Destructor Documentation	31
8.2.2.1	BasicJeodTrickSimInterface() [1/2]	31
8.2.2.2	~BasicJeodTrickSimInterface()	31
8.2.2.3	BasicJeodTrickSimInterface() [2/2]	31
8.2.3	Member Function Documentation	31
8.2.3.1	checkpoint_allocations()	32
8.2.3.2	checkpoint_containers()	32
8.2.3.3	close_checkpoint_file()	32
8.2.3.4	close_restart_file()	32
8.2.3.5	create_integrator_internal()	33
8.2.3.6	get_checkpoint_file_name()	33
8.2.3.7	get_checkpoint_reader_internal()	33

8.2.3.8	get_checkpoint_writer_internal()	34
8.2.3.9	get_job_cycle_internal()	34
8.2.3.10	get_memory_interface_internal()	34
8.2.3.11	open_checkpoint_file()	35
8.2.3.12	open_restart_file()	35
8.2.3.13	operator=()	35
8.2.3.14	restore_allocations()	35
8.2.3.15	restore_containers()	36
8.2.3.16	set_checkpoint_file_name()	36
8.2.3.17	set_mode()	36
8.2.4	Friends And Related Function Documentation	36
8.2.4.1	init_attrjeod__BasicJeodTrickSimInterface	37
8.2.4.2	InputProcessor	37
8.2.5	Field Documentation	37
8.2.5.1	checkpoint_file_name	37
8.2.5.2	checkpoint_reader	37
8.2.5.3	checkpoint_writer	38
8.2.5.4	generic_message_handler	38
8.2.5.5	memory_manager	38
8.2.5.6	section_end	38
8.2.5.7	section_start	39
8.2.5.8	trick_memory_interface	39
8.3	jeod::CheckPointInputManager Class Reference	39
8.3.1	Detailed Description	41
8.3.2	Constructor & Destructor Documentation	41
8.3.2.1	CheckPointInputManager() [1/2]	41
8.3.2.2	CheckPointInputManager() [2/2]	41
8.3.3	Member Function Documentation	41
8.3.3.1	create_section_reader() [1/2]	42
8.3.3.2	create_section_reader() [2/2]	42

8.3.3.3	create_trick_section_reader()	43
8.3.3.4	deregister_reader()	43
8.3.3.5	have_active_reader()	44
8.3.3.6	initialize()	44
8.3.3.7	operator"()()	44
8.3.3.8	operator=()	44
8.3.3.9	register_reader()	44
8.3.4	Field Documentation	45
8.3.4.1	current_reader	45
8.3.4.2	filename	45
8.3.4.3	is_open	45
8.3.4.4	section_end	46
8.3.4.5	section_start	46
8.3.4.6	sections	46
8.3.4.7	stream	46
8.4	jeod::CheckPointOutputManager Class Reference	47
8.4.1	Detailed Description	48
8.4.2	Constructor & Destructor Documentation	48
8.4.2.1	CheckPointOutputManager() [1/2]	48
8.4.2.2	CheckPointOutputManager() [2/2]	48
8.4.3	Member Function Documentation	48
8.4.3.1	create_section_writer() [1/2]	49
8.4.3.2	create_section_writer() [2/2]	49
8.4.3.3	create_trick_section_writer()	50
8.4.3.4	deregister_writer()	50
8.4.3.5	have_active_writer()	50
8.4.3.6	operator"()()	51
8.4.3.7	operator=()	51
8.4.3.8	register_writer()	51
8.4.4	Friends And Related Function Documentation	52

8.4.4.1	MemoryManagerWrapper	52
8.4.5	Field Documentation	52
8.4.5.1	current_writer	52
8.4.5.2	filename	52
8.4.5.3	is_open	53
8.4.5.4	section_end	53
8.4.5.5	section_start	53
8.4.5.6	stream	53
8.5	jeod::JeodTrickMemoryInterface::ContainerListEntry Struct Reference	54
8.5.1	Detailed Description	54
8.5.2	Constructor & Destructor Documentation	54
8.5.2.1	ContainerListEntry()	54
8.5.3	Field Documentation	55
8.5.3.1	container	55
8.5.3.2	elem_name	55
8.5.3.3	owner	55
8.5.3.4	owner_type	56
8.6	jeod::JeodDynbodyIntegrationLoop Class Reference	56
8.6.1	Detailed Description	58
8.6.2	Constructor & Destructor Documentation	58
8.6.2.1	JeodDynbodyIntegrationLoop() [1/3]	58
8.6.2.2	JeodDynbodyIntegrationLoop() [2/3]	59
8.6.2.3	~JeodDynbodyIntegrationLoop()	59
8.6.2.4	JeodDynbodyIntegrationLoop() [3/3]	59
8.6.3	Member Function Documentation	60
8.6.3.1	add_integrable_object()	60
8.6.3.2	add_sim_object()	60
8.6.3.3	add_sim_object_bodies() [1/2]	61
8.6.3.4	add_sim_object_bodies() [2/2]	62
8.6.3.5	collect_derivatives()	62

8.6.3.6	find_containing_sim_object()	62
8.6.3.7	gravitation()	63
8.6.3.8	initialize_integ_loop()	63
8.6.3.9	integrate_dt()	63
8.6.3.10	operator=()	64
8.6.3.11	remove_integrable_object()	64
8.6.3.12	remove_sim_object()	64
8.6.3.13	remove_sim_object_bodies()	65
8.6.3.14	set_deriv_ephem_update()	65
8.6.3.15	set_time_to_loop_start()	65
8.6.3.16	update_integration_group()	66
8.6.4	Friends And Related Function Documentation	66
8.6.4.1	init_attrjeod__JeodDynbodyIntegrationLoop	66
8.6.4.2	InputProcessor	66
8.6.5	Field Documentation	67
8.6.5.1	deriv_ephem_update	67
8.6.5.2	dyn_manager	67
8.6.5.3	gravity_manager	67
8.6.5.4	integ_constructor	68
8.6.5.5	integ_group	68
8.6.5.6	integ_group_factory	68
8.6.5.7	integ_interface	68
8.6.5.8	loop_sim_object	69
8.6.5.9	time_manager	69
8.7	jeod::JeodIntegratorInterface Class Reference	69
8.7.1	Detailed Description	70
8.7.2	Constructor & Destructor Documentation	70
8.7.2.1	~JeodIntegratorInterface()	70
8.7.3	Member Function Documentation	70
8.7.3.1	get_integrator()	70

8.7.3.2	interpret_integration_type()	71
8.7.4	Friends And Related Function Documentation	71
8.7.4.1	init_attrjeod__JeodIntegratorInterface	71
8.7.4.2	InputProcessor	71
8.8	jeod::JeodMemoryInterface Class Reference	71
8.8.1	Detailed Description	72
8.8.2	Constructor & Destructor Documentation	73
8.8.2.1	JeodMemoryInterface() [1/2]	73
8.8.2.2	~JeodMemoryInterface()	73
8.8.2.3	JeodMemoryInterface() [2/2]	73
8.8.3	Member Function Documentation	73
8.8.3.1	deregister_allocation()	73
8.8.3.2	deregister_container()	74
8.8.3.3	find_attributes() [1/2]	74
8.8.3.4	find_attributes() [2/2]	74
8.8.3.5	get_address_at_name()	75
8.8.3.6	get_name_at_address()	75
8.8.3.7	is_checkpoint_restart_supported()	76
8.8.3.8	operator=()	76
8.8.3.9	pointer_attributes()	76
8.8.3.10	primitive_attributes()	77
8.8.3.11	register_allocation()	77
8.8.3.12	register_container()	78
8.8.3.13	structure_attributes()	78
8.8.3.14	void_pointer_attributes()	78
8.8.4	Friends And Related Function Documentation	79
8.8.4.1	init_attrjeod__JeodMemoryInterface	79
8.8.4.2	InputProcessor	79
8.9	jeod::JeodSimulationInterface Class Reference	79
8.9.1	Detailed Description	81

8.9.2	Member Enumeration Documentation	81
8.9.2.1	Mode	81
8.9.3	Constructor & Destructor Documentation	82
8.9.3.1	JeodSimulationInterface() [1/2]	82
8.9.3.2	~JeodSimulationInterface()	82
8.9.3.3	JeodSimulationInterface() [2/2]	82
8.9.4	Member Function Documentation	82
8.9.4.1	configure()	82
8.9.4.2	create_integrator_interface()	83
8.9.4.3	create_integrator_internal()	83
8.9.4.4	get_address_at_name()	83
8.9.4.5	get_checkpoint_reader()	84
8.9.4.6	get_checkpoint_reader_internal()	84
8.9.4.7	get_checkpoint_writer()	85
8.9.4.8	get_checkpoint_writer_internal()	85
8.9.4.9	get_job_cycle()	85
8.9.4.10	get_job_cycle_internal()	86
8.9.4.11	get_memory_interface()	86
8.9.4.12	get_memory_interface_internal()	86
8.9.4.13	get_mode()	87
8.9.4.14	get_name_at_address()	87
8.9.4.15	operator=()	87
8.9.4.16	set_mode()	88
8.9.5	Friends And Related Function Documentation	88
8.9.5.1	init_attrjeod__JeodSimulationInterface	88
8.9.5.2	InputProcessor	88
8.9.6	Field Documentation	88
8.9.6.1	mode	89
8.9.6.2	saved_mode	89
8.9.6.3	sim_interface	89

8.10	jeod::JeodSimulationInterfaceInit Class Reference	90
8.10.1	Detailed Description	90
8.10.2	Constructor & Destructor Documentation	90
8.10.2.1	JeodSimulationInterfaceInit()	90
8.10.3	Field Documentation	90
8.10.3.1	memory_debug_level	91
8.10.3.2	message_suppress_id	91
8.10.3.3	message_suppress_location	91
8.10.3.4	message_suppression_level	91
8.11	jeod::JeodTrick10MemoryInterface Class Reference	92
8.11.1	Detailed Description	93
8.11.2	Constructor & Destructor Documentation	93
8.11.2.1	JeodTrick10MemoryInterface() [1/2]	93
8.11.2.2	~JeodTrick10MemoryInterface()	94
8.11.2.3	JeodTrick10MemoryInterface() [2/2]	94
8.11.3	Member Function Documentation	94
8.11.3.1	checkpoint_allocations()	94
8.11.3.2	checkpoint_containers()	94
8.11.3.3	deregister_container()	95
8.11.3.4	get_address_at_name()	95
8.11.3.5	get_container_id()	96
8.11.3.6	get_name_at_address()	96
8.11.3.7	get_trick_checkpoint_file()	97
8.11.3.8	is_checkpoint_restart_supported()	97
8.11.3.9	operator=()	98
8.11.3.10	register_container()	98
8.11.3.11	restore_allocations()	98
8.11.3.12	restore_containers()	99
8.11.3.13	translate_addr_to_name()	99
8.11.3.14	translate_name_to_addr()	100

8.11.4	Friends And Related Function Documentation	100
8.11.4.1	init_attrjeod__JeodTrick10MemoryInterface	100
8.11.4.2	InputProcessor	101
8.11.5	Field Documentation	101
8.11.5.1	trick_checkpoint_agent	101
8.12	jeod::JeodTrickIntegrator Class Reference	101
8.12.1	Detailed Description	102
8.12.2	Constructor & Destructor Documentation	102
8.12.2.1	JeodTrickIntegrator() [1/2]	103
8.12.2.2	~JeodTrickIntegrator()	103
8.12.2.3	JeodTrickIntegrator() [2/2]	103
8.12.3	Member Function Documentation	103
8.12.3.1	get_dt()	103
8.12.3.2	get_first_step_derivs_flag()	103
8.12.3.3	get_integrator()	104
8.12.3.4	interpret_integration_type()	104
8.12.3.5	operator=()	104
8.12.3.6	reset_first_step_derivs_flag()	104
8.12.3.7	restore_first_step_derivs_flag()	104
8.12.3.8	set_first_step_derivs_flag()	104
8.12.3.9	set_step_number()	105
8.12.3.10	set_time()	105
8.12.4	Friends And Related Function Documentation	105
8.12.4.1	init_attrjeod__JeodTrickIntegrator	105
8.12.4.2	InputProcessor	106
8.12.5	Field Documentation	106
8.12.5.1	default_first_step_deriv	106
8.12.5.2	trick_integrator	106
8.13	jeod::JeodTrickMemoryInterface Class Reference	106
8.13.1	Detailed Description	108

8.13.2	Member Typedef Documentation	109
8.13.2.1	AllocationMap	109
8.13.2.2	ContainerList	109
8.13.3	Constructor & Destructor Documentation	109
8.13.3.1	JeodTrickMemoryInterface() [1/2]	109
8.13.3.2	~JeodTrickMemoryInterface()	109
8.13.3.3	JeodTrickMemoryInterface() [2/2]	110
8.13.4	Member Function Documentation	110
8.13.4.1	checkpoint_allocations()	110
8.13.4.2	checkpoint_containers()	110
8.13.4.3	construct_identifier()	110
8.13.4.4	deregister_allocation()	111
8.13.4.5	deregister_container()	111
8.13.4.6	find_attributes() [1/2]	112
8.13.4.7	find_attributes() [2/2]	112
8.13.4.8	get_address_at_name()	113
8.13.4.9	get_name_at_address()	113
8.13.4.10	get_trick_checkpoint_file()	114
8.13.4.11	is_checkpoint_restart_supported()	114
8.13.4.12	operator=()	115
8.13.4.13	pointer_attributes()	115
8.13.4.14	primitive_attributes()	115
8.13.4.15	register_allocation()	116
8.13.4.16	register_container()	116
8.13.4.17	restore_allocations()	117
8.13.4.18	restore_containers()	117
8.13.4.19	set_mode()	117
8.13.4.20	structure_attributes()	118
8.13.4.21	void_pointer_attributes()	118
8.13.5	Friends And Related Function Documentation	119

8.13.5.1	init_attrjeod__JeodTrickMemoryInterface	119
8.13.5.2	InputProcessor	119
8.13.6	Field Documentation	119
8.13.6.1	allocation_map	119
8.13.6.2	container_list	119
8.13.6.3	dlhandle	120
8.13.6.4	id_length	120
8.13.6.5	id_prefix	120
8.13.6.6	mode	120
8.14	jeod::JeodTrickSimInterface Class Reference	121
8.14.1	Detailed Description	121
8.14.2	Constructor & Destructor Documentation	121
8.14.2.1	JeodTrickSimInterface() [1/2]	122
8.14.2.2	~JeodTrickSimInterface()	122
8.14.2.3	JeodTrickSimInterface() [2/2]	122
8.14.3	Member Function Documentation	122
8.14.3.1	operator=()	122
8.14.4	Friends And Related Function Documentation	122
8.14.4.1	init_attrjeod__JeodTrickSimInterface	122
8.14.4.2	InputProcessor	123
8.15	jeod::SectionedInputBuffer Class Reference	123
8.15.1	Detailed Description	124
8.15.2	Constructor & Destructor Documentation	124
8.15.2.1	~SectionedInputBuffer()	124
8.15.2.2	SectionedInputBuffer() [1/2]	125
8.15.2.3	SectionedInputBuffer() [2/2]	125
8.15.3	Member Function Documentation	125
8.15.3.1	activate()	125
8.15.3.2	deactivate()	126
8.15.3.3	operator"()()	126

8.15.3.4	operator=()	126
8.15.3.5	underflow()	127
8.15.4	Friends And Related Function Documentation	127
8.15.4.1	SectionedInputStream	127
8.15.5	Field Documentation	127
8.15.5.1	at_eof	127
8.15.5.2	buf	128
8.15.5.3	curr_pos	128
8.15.5.4	end_pos	128
8.15.5.5	file_buf	128
8.15.5.6	start_pos	129
8.16	jeod::SectionedInputStream Class Reference	129
8.16.1	Detailed Description	130
8.16.2	Constructor & Destructor Documentation	132
8.16.2.1	SectionedInputStream() [1/3]	132
8.16.2.2	SectionedInputStream() [2/3]	132
8.16.2.3	~SectionedInputStream()	133
8.16.2.4	SectionedInputStream() [3/3]	133
8.16.3	Member Function Documentation	133
8.16.3.1	activate()	133
8.16.3.2	deactivate()	134
8.16.3.3	is_activatable()	134
8.16.3.4	operator void *()	135
8.16.3.5	operator"!()	135
8.16.3.6	operator=()	135
8.16.4	Friends And Related Function Documentation	135
8.16.4.1	CheckPointInputManager	135
8.16.5	Field Documentation	136
8.16.5.1	end_pos	136
8.16.5.2	is_active	136

8.16.5.3	is_copy	136
8.16.5.4	manager	137
8.16.5.5	sectbuf	137
8.16.5.6	start_pos	137
8.16.5.7	stream	137
8.17	jeod::SectionedOutputBuffer Class Reference	138
8.17.1	Detailed Description	139
8.17.2	Constructor & Destructor Documentation	139
8.17.2.1	~SectionedOutputBuffer()	139
8.17.2.2	SectionedOutputBuffer() [1/3]	139
8.17.2.3	SectionedOutputBuffer() [2/3]	139
8.17.2.4	SectionedOutputBuffer() [3/3]	140
8.17.3	Member Function Documentation	140
8.17.3.1	activate()	140
8.17.3.2	deactivate()	140
8.17.3.3	operator"!"()	141
8.17.3.4	operator=()	141
8.17.3.5	overflow()	141
8.17.4	Friends And Related Function Documentation	141
8.17.4.1	SectionedOutputStream	142
8.17.5	Field Documentation	142
8.17.5.1	file_buf	142
8.18	jeod::SectionedOutputStream Class Reference	142
8.18.1	Detailed Description	144
8.18.2	Constructor & Destructor Documentation	144
8.18.2.1	SectionedOutputStream() [1/3]	144
8.18.2.2	SectionedOutputStream() [2/3]	144
8.18.2.3	~SectionedOutputStream()	145
8.18.2.4	SectionedOutputStream() [3/3]	145
8.18.3	Member Function Documentation	145

8.18.3.1	activate()	145
8.18.3.2	deactivate()	146
8.18.3.3	is_activatable()	146
8.18.3.4	operator void *()	147
8.18.3.5	operator"!()	147
8.18.3.6	operator=()	147
8.18.4	Friends And Related Function Documentation	147
8.18.4.1	CheckPointOutputManager	147
8.18.5	Field Documentation	148
8.18.5.1	is_active	148
8.18.5.2	is_copy	148
8.18.5.3	manager	148
8.18.5.4	sectbuf	149
8.18.5.5	section_end	149
8.18.5.6	section_start	149
8.18.5.7	stream	149
8.18.5.8	tag	150
8.19	jeod::CheckPointInputManager::SectionInfo Struct Reference	150
8.19.1	Detailed Description	150
8.19.2	Constructor & Destructor Documentation	150
8.19.2.1	SectionInfo()	150
8.19.3	Field Documentation	151
8.19.3.1	end_pos	151
8.19.3.2	start_pos	151
8.20	jeod::SimInterfaceMessages Class Reference	151
8.20.1	Detailed Description	152
8.20.2	Constructor & Destructor Documentation	152
8.20.2.1	SimInterfaceMessages() [1/2]	152
8.20.2.2	SimInterfaceMessages() [2/2]	152
8.20.3	Member Function Documentation	152

8.20.3.1	operator=()	153
8.20.4	Field Documentation	153
8.20.4.1	implementation_error	153
8.20.4.2	integration_error	153
8.20.4.3	interface_error	154
8.20.4.4	phasing_error	154
8.20.4.5	singleton_error	154
8.21	jeod::TrickJeodIntegrator Class Reference	155
8.21.1	Detailed Description	155
8.21.2	Constructor & Destructor Documentation	155
8.21.2.1	~TrickJeodIntegrator()	155
8.21.3	Member Function Documentation	155
8.21.3.1	initialize()	156
8.21.3.2	integrate()	156
8.22	jeod::TrickMessageHandler Class Reference	156
8.22.1	Detailed Description	157
8.22.2	Constructor & Destructor Documentation	157
8.22.2.1	TrickMessageHandler() [1/2]	157
8.22.2.2	~TrickMessageHandler()	157
8.22.2.3	TrickMessageHandler() [2/2]	157
8.22.3	Member Function Documentation	157
8.22.3.1	operator=()	158
8.22.3.2	process_message()	158
8.22.3.3	register_contents()	158
8.22.4	Friends And Related Function Documentation	159
8.22.4.1	init_attrjeod__TrickMessageHandler	159
8.22.4.2	InputProcessor	159
8.23	jeod::TrickMessageHandlerMixin Class Reference	159
8.23.1	Detailed Description	160
8.23.2	Constructor & Destructor Documentation	160
8.23.2.1	TrickMessageHandlerMixin() [1/2]	160
8.23.2.2	~TrickMessageHandlerMixin()	160
8.23.2.3	TrickMessageHandlerMixin() [2/2]	160
8.23.3	Member Function Documentation	161
8.23.3.1	operator=()	161
8.23.4	Friends And Related Function Documentation	161
8.23.4.1	init_attrjeod__TrickMessageHandlerMixin	161
8.23.4.2	InputProcessor	161
8.23.5	Field Documentation	161
8.23.5.1	message_handler	161

9 File Documentation	163
9.1 checkpoint_input_manager.cc File Reference	163
9.1.1 Detailed Description	163
9.2 checkpoint_input_manager.hh File Reference	163
9.2.1 Detailed Description	164
9.3 checkpoint_output_manager.cc File Reference	164
9.3.1 Detailed Description	164
9.4 checkpoint_output_manager.hh File Reference	165
9.4.1 Detailed Description	165
9.5 class_declarations.hh File Reference	165
9.5.1 Detailed Description	165
9.6 config.hh File Reference	166
9.6.1 Detailed Description	166
9.7 config_test_harness.hh File Reference	166
9.7.1 Detailed Description	166
9.8 config_trick10.hh File Reference	166
9.8.1 Detailed Description	167
9.9 jeod_class.hh File Reference	167
9.9.1 Detailed Description	167
9.10 jeod_integrator_interface.hh File Reference	168
9.10.1 Detailed Description	168
9.11 jeod_trick_integrator.hh File Reference	168
9.11.1 Detailed Description	169
9.12 jeod_va_macro_utility.hh File Reference	169
9.12.1 Detailed Description	169
9.13 memory_attributes.hh File Reference	169
9.13.1 Detailed Description	170
9.13.2 Macro Definition Documentation	170
9.13.2.1 JEOD_ATTRIBUTES	170
9.13.2.2 JEOD_DECLARE_ATTRIBUTES	171

9.14	memory_interface.hh File Reference	171
9.14.1	Detailed Description	171
9.15	sim_interface_messages.cc File Reference	171
9.15.1	Detailed Description	172
9.16	sim_interface_messages.hh File Reference	172
9.16.1	Detailed Description	172
9.17	simulation_interface.cc File Reference	172
9.17.1	Detailed Description	173
9.18	simulation_interface.hh File Reference	173
9.18.1	Detailed Description	173
9.19	trick10_memory_interface.cc File Reference	173
9.19.1	Detailed Description	174
9.20	trick10_memory_interface.hh File Reference	174
9.20.1	Detailed Description	174
9.21	trick_dynbody_integ_loop.cc File Reference	175
9.21.1	Detailed Description	175
9.22	trick_dynbody_integ_loop.hh File Reference	175
9.22.1	Detailed Description	176
9.23	trick_memory_interface.cc File Reference	176
9.23.1	Detailed Description	176
9.24	trick_memory_interface.hh File Reference	176
9.24.1	Detailed Description	177
9.25	trick_memory_interface_alloc.cc File Reference	177
9.25.1	Detailed Description	178
9.26	trick_memory_interface_attrib.cc File Reference	178
9.26.1	Detailed Description	178
9.27	trick_memory_interface_chkpnt.cc File Reference	178
9.27.1	Detailed Description	179
9.28	trick_memory_interface_xlate.cc File Reference	179
9.28.1	Detailed Description	179
9.29	trick_message_handler.cc File Reference	180
9.29.1	Detailed Description	180
9.30	trick_message_handler.hh File Reference	180
9.30.1	Detailed Description	181
9.31	trick_sim_interface.cc File Reference	181
9.31.1	Detailed Description	181
9.32	trick_sim_interface.hh File Reference	181
9.32.1	Detailed Description	182

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Models	11
Utils	12
SimInterface	13

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

er7_utils	Namespace er7_utils contains the state integration models used by JEOD	25
jeod	Namespace jeod	25
Trick	Namespace Trick furnishes several standard functions for use in the Trick environment	26

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

jeod::JeodTrickMemoryInterface::AllocationMapEntry	27
jeod::CheckPointInputManager	39
jeod::CheckPointOutputManager	47
jeod::JeodTrickMemoryInterface::ContainerListEntry	54
IntegLoopScheduler	
jeod::JeodDynbodyIntegrationLoop	56
Integrator	
jeod::TrickJeodIntegrator	155
IntegratorInterface	
jeod::JeodIntegratorInterface	69
jeod::JeodTrickIntegrator	101
std::ios_base	
std::basic_ios	
std::basic_istream	
std::istream	
jeod::SectionedInputStream	129
std::basic_ostream	
std::ostream	
jeod::SectionedOutputStream	142
JeodIntegrationGroupOwner	
jeod::JeodDynbodyIntegrationLoop	56
jeod::JeodMemoryInterface	71
jeod::JeodTrickMemoryInterface	106
jeod::JeodTrick10MemoryInterface	92
jeod::JeodSimulationInterface	79
jeod::BasicJeodTrickSimInterface	29
jeod::JeodTrickSimInterface	121
jeod::JeodSimulationInterfaceInit	90
jeod::CheckPointInputManager::SectionInfo	150
jeod::SimInterfaceMessages	151
streambuf	
jeod::SectionedInputBuffer	123
jeod::SectionedOutputBuffer	138
SuppressedCodeMessageHandler	
jeod::TrickMessageHandler	156
jeod::TrickMessageHandlerMixin	159
jeod::JeodTrickSimInterface	121

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

jeod::JeodTrickMemoryInterface::AllocationMapEntry	27
Describes a chunk of JEOD-allocated memory	
jeod::BasicJeodTrickSimInterface	
The BasicJeodTrickSimInterface implements the required capabilities of the generic JeodSimulationInterface in a Trick simulation environment	29
jeod::CheckPointInputManager	
A CheckPointInputManager provides tools for reading a checkpoint file	39
jeod::CheckPointOutputManager	
A CheckPointOutputManager provides the basic tools for writing a checkpoint file	47
jeod::JeodTrickMemoryInterface::ContainerListEntry	
Describes a Checkpointable object	54
jeod::JeodDynbodyIntegrationLoop	
A Trick::IntegLoopScheduler that provides the ability to integrate a collection of Trick::SimObject instances over time, with the sim objects capable of being moved from one integration loop to another during run time	56
jeod::JeodIntegratorInterface	
A JeodIntegratorInterface extends the ER7 IntegratorInterface with the concept of a pointer to the simulation engine's integration object	69
jeod::JeodMemoryInterface	
Abstract interface between the JEOD memory manager and the simulation engine	71
jeod::JeodSimulationInterface	
This abstract class defines the basis for the interface between JEOD and a simulation engine	79
jeod::JeodSimulationInterfaceInit	
Define configuration data needed to configure the dynamically-created message handler and memory manager	90
jeod::JeodTrick10MemoryInterface	
A TrickMemoryInterface implements the two required methods needed to register and deregister memory with the simulation engine, Trick in this case	92
jeod::JeodTrickIntegrator	
A JeodTrickIntegrator specializes the JeodIntegratorInterface for use with Trick as the simulation engine	101
jeod::JeodTrickMemoryInterface	
A TrickMemoryInterface implements the two required methods needed to register and deregister memory with the simulation engine, Trick in this case	106

jeod::JeodTrickSimInterface	
A JEODTrickSimInterface implements the required capabilities of the generic JeodSimulationInterface in a Trick simulation environment	121
jeod::SectionedInputBuffer	
A SectionedInputBuffer is a <code>std::streambuf</code> that reads from a section in a checkpoint file	123
jeod::SectionedInputStream	
A SectionedInputStream is a <code>std::istream</code> that reads from a section in a checkpoint file	129
jeod::SectionedOutputBuffer	
A SectionedOutputBuffer is a <code>std::streambuf</code> that writes a section of a checkpoint file	138
jeod::SectionedOutputStream	
A SectionedOutputStream is a <code>std::ostream</code> that writes a section of a checkpoint file	142
jeod::CheckPointInputManager::SectionInfo	
A SectionInfo contains the start and end positions of a checkpoint file section	150
jeod::SimInterfaceMessages	
Specifies the message IDs used in the <code>sim_interface</code> model	151
jeod::TrickJeodIntegrator	
A TrickJeodIntegrator specializes the <code>Trick::Integrator</code> to provide the Trick side of the integration interface between Trick and JEOD	155
jeod::TrickMessageHandler	
The <code>MessageHandler</code> class for designed for use in Trick -based simulations	156
jeod::TrickMessageHandlerMixin	
The TrickMessageHandlerMixin implements the required capabilities of the generic JeodSimulationInterface in a Trick simulation environment	159

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

checkpoint_input_manager.cc	Define CheckPointInputManager member functions and of related classes	163
checkpoint_input_manager.hh	Define class CheckPointInputManager and related classes	163
checkpoint_output_manager.cc	Define CheckPointOutputManager member functions and of related classes	164
checkpoint_output_manager.hh	Define class CheckPointOutputManager and related classes	165
class_declarations.hh	Forward declarations of classes defined in the utils/sim_interface model	165
config.hh	Configure JEOD for use by some simulation engine	166
config_test_harness.hh	Configure JEOD for use in standalone test mode	166
config_trick10.hh	Configure JEOD for use in a Trick10 environment	166
jeod_class.hh	Define the JEOD class declaration macros JEOD_MAKE_SIM_INTERFACES and and JEOD↔ _DECLARE_SIM_INTERFACES	167
jeod_integrator_interface.hh	Define the interface for accessing / updating elements of a simulation engine's integrator object	168
jeod_trick_integrator.hh	Define the interface for accessing / updating elements of a Trick simulation integrator object . .	168
jeod_va_macro_utility.hh	Support header for variable argument macro functions	169
memory_attributes.hh	Define JEOD memory interface macros	169
memory_interface.hh	Define the MemoryInterface class, which abstractly defines the interface between the memory manager and the simulation engine	171
sim_interface_messages.cc	Implement the class SimInterfaceMessages	171
sim_interface_messages.hh	Define the class SimInterfaceMessages, the class that specifies the message IDs used in the sim interface model	172

simulation_interface.cc	Implement SimulationInterface methods	172
simulation_interface.hh	Define the abstract class JeodSimulationInterface	173
trick10_memory_interface.cc	Define JeodTrickMemoryInterface methods	173
trick10_memory_interface.hh	Define the interface for registering / deregistering memory with Trick	174
trick_dynbody_integ_loop.cc	Define JeodDynbodyIntegrationLoop methods	175
trick_dynbody_integ_loop.hh	Define the class IntegrationGroupIntegLoopScheduler, which replaces the base Trick integration loop for multi-rate JEOD-based simulations	175
trick_memory_interface.cc	Define JeodTrickMemoryInterface methods	176
trick_memory_interface.hh	Define the interface for registering / deregistering memory with Trick	176
trick_memory_interface_alloc.cc	Define JeodTrickMemoryInterface methods related to allocation/deallocation	177
trick_memory_interface_attrib.cc	Define JeodTrickMemoryInterface methods related to attributes	178
trick_memory_interface_chkpt.cc	Define JeodTrick10MemoryInterface methods related to checkpoint/restart	178
trick_memory_interface_xlate.cc	Define JeodTrickMemoryInterface methods related to name translation	179
trick_message_handler.cc	Define member functions for the class TrickMessageHandler	180
trick_message_handler.hh	Define the class TrickMessageHandler, the message handler designed for use in Trick-based simulations	180
trick_sim_interface.cc	Implement TrickSimInterface methods	181
trick_sim_interface.hh	Define the class JeodTrickSimInterface	181

Chapter 6

Module Documentation

6.1 Models

Modules

- [Utils](#)

6.1.1 Detailed Description

6.2 Utils

Modules

- [SimInterface](#)

6.2.1 Detailed Description

6.3 SimInterface

Files

- file [checkpoint_input_manager.hh](#)
Define class CheckPointInputManager and related classes.
- file [checkpoint_output_manager.hh](#)
Define class CheckPointOutputManager and related classes.
- file [class_declarations.hh](#)
Forward declarations of classes defined in the utils/sim_interface model.
- file [config.hh](#)
Configure JEOD for use by some simulation engine.
- file [config_test_harness.hh](#)
Configure JEOD for use in standalone test mode.
- file [config_trick10.hh](#)
Configure JEOD for use in a Trick10 environment.
- file [jeod_class.hh](#)
*Define the JEOD class declaration macros JEOD_MAKE_SIM_INTERFACES and and JEOD_DECLARE_SIM_IN-
TERFACES.*
- file [jeod_integrator_interface.hh](#)
Define the interface for accessing / updating elements of a simulation engine's integrator object.
- file [jeod_trick_integrator.hh](#)
Define the interface for accessing / updating elements of a [Trick](#) simulation integrator object.
- file [jeod_va_macro_utility.hh](#)
Support header for variable argument macro functions.
- file [memory_attributes.hh](#)
Define JEOD memory interface macros.
- file [memory_interface.hh](#)
Define the MemoryInterface class, which abstractly defines the interface between the memory manager and the simulation engine.
- file [sim_interface_messages.hh](#)
Define the class SimInterfaceMessages, the class that specifies the message IDs used in the sim interface model.
- file [simulation_interface.hh](#)
Define the abstract class JeodSimulationInterface.
- file [trick10_memory_interface.hh](#)
Define the interface for registering / deregistering memory with [Trick](#).
- file [trick_dynbody_integ_loop.hh](#)
Define the class IntegrationGroupIntegLoopScheduler, which replaces the base [Trick](#) integration loop for multi-rate JEOD-based simulations.
- file [trick_memory_interface.hh](#)
Define the interface for registering / deregistering memory with [Trick](#).
- file [trick_message_handler.hh](#)
Define the class TrickMessageHandler, the message handler designed for use in Trick-based simulations.
- file [trick_sim_interface.hh](#)
Define the class JeodTrickSimInterface.
- file [checkpoint_input_manager.cc](#)
Define CheckPointInputManager member functions and of related classes.
- file [checkpoint_output_manager.cc](#)
Define CheckPointOutputManager member functions and of related classes.
- file [sim_interface_messages.cc](#)
Implement the class SimInterfaceMessages.

- file [simulation_interface.cc](#)
Implement SimulationInterface methods.
- file [trick10_memory_interface.cc](#)
Define JeodTrickMemoryInterface methods.
- file [trick_dynbody_integ_loop.cc](#)
Define JeodDynbodyIntegrationLoop methods.
- file [trick_memory_interface.cc](#)
Define JeodTrickMemoryInterface methods.
- file [trick_memory_interface_alloc.cc](#)
Define JeodTrickMemoryInterface methods related to allocation/deallocation.
- file [trick_memory_interface_attrib.cc](#)
Define JeodTrickMemoryInterface methods related to attributes.
- file [trick_memory_interface_chkpnt.cc](#)
Define JeodTrick10MemoryInterface methods related to checkpoint/restart.
- file [trick_memory_interface_xlate.cc](#)
Define JeodTrickMemoryInterface methods related to name translation.
- file [trick_message_handler.cc](#)
Define member functions for the class TrickMessageHandler.
- file [trick_sim_interface.cc](#)
Implement TrickSimInterface methods.

Namespaces

- [jeod](#)
Namespace jeod.
- [Trick](#)
Namespace [Trick](#) furnishes several standard functions for use in the [Trick](#) environment.
- [er7_utils](#)
Namespace [er7_utils](#) contains the state integration models used by JEOD.

Macros

- `#define JEOD_UNUSED`
- `#define ER7_UTILS_UNUSED`
- `#define ER7_UTILS_RESTRICT`
- `#define ER7_UTILS_ALWAYS_INLINE`
- `#define JEOD_ATTRIBUTES_TYPE int`
- `#define JEOD_ATTRIBUTES_POINTER_TYPE void *`
- `#define JEOD_SIM_INTEGRATOR_POINTER_TYPE void *`
- `#define JEOD_SIZE_T size_t`
- `#define JEOD_PTRDIFF_T long int`
- `#define JEOD_INTPTR_T long int`
- `#define JEOD_UINTPTR_T unsigned long int`
- `#define JEOD_CLASS_ESTABLISH_FRIENDS3(ns1, ns2, class_name)`
- `#define JEOD_CLASS_ESTABLISH_FRIENDS2(ns, class_name)`
- `#define JEOD_CLASS_ESTABLISH_FRIENDS1(class_name)`
- `#define JEOD_CLASS_ESTABLISH_FRIENDS(...) JEODVMACRO(JEOD_CLASS_ESTABLISH_FRIENDS1, __VA_ARGS__)`
- `#define JEOD_ATTRIBUTES_SIM_ENGINE_HEADER "sim_services/MemoryManager/include/attributes.h"`
- `#define JEOD_ATTRIBUTES_TYPE struct ATTRIBUTES_tag`
- `#define JEOD_ATTRIBUTES_POINTER_TYPE JEOD_ATTRIBUTES_TYPE *`

- `#define JEOD_SIM_INTEGRATOR_SIM_ENGINE_HEADER "sim_services/Integrator/include/Integrator.hh"`
- `#define JEOD_SIM_INTEGRATOR_FORWARD`
- `#define JEOD_SIM_INTEGRATOR_POINTER_TYPE Trick::Integrator *`
- `#define JEOD_SIM_INTEGRATOR_ENUM Integrator_type`
- `#define JEOD_MAKE_SIM_INTERFACES(...) JEOD_CLASS_ESTABLISH_FRIENDS(__VA_ARGS__);`
`JEOD_MAKE_SIM_INTERFACES(class_name)` Defines friends of the given class.
- `#define JEOD_DECLARE_SIM_INTERFACES(class_name)`
`JEOD_DECLARE_SIM_INTERFACES(class_name)` Forward declare classes and external functions needed to make the `JEOD_MAKE_SIM_INTERFACES(class_name)` expansion compilable.
- `#define JEODVMACRO_VA_SIZE(...) JEODVMACRO_GET_COUNT(__VA_ARGS__, JEODVMACRO_REVSEQ_COUNT())`
- `#define JEODVMACRO_GET_COUNT(...) JEODVMACRO_SEQ_COUNT(__VA_ARGS__)`
- `#define JEODVMACRO_SEQ_COUNT(_1, _2, _3, _4, _5, _6, _7, _8, _9, _10, _11, _12, _13, _14, _15, _16, _17, _18, _19, _20, _21, _22, _23, _24, _25, _26, _27, _28, _29, _30, _31, _32, _33, _34, _35, _36, _37, _38, _39, _40, _41, _42, _43, _44, _45, _46, _47, _48, _49, _50, _51, _52, _53, _54, _55, _56, _57, _58, _59, _60, _61, _62, _63, N, ...) N`
- `#define JEODVMACRO_REVSEQ_COUNT()`
- `#define JEODVMACRO_SELECT_FUNC_CAT(name, n) name##n`
- `#define JEODVMACRO_SELECT_FUNC(name, n) JEODVMACRO_SELECT_FUNC_CAT(name, n)`
- `#define JEODVMACRO(func, ...) JEODVMACRO_SELECT_FUNC(func, JEODVMACRO_VA_SIZE(__VA_ARGS__)(__VA_ARGS__))`
- `#define PATH "utils/sim_interface/"`
- `#define CLASS SimInterfaceMessages`
- `#define MAKE_MESSAGE_CODE(id) char const * CLASS::id = PATH #id`
- `#define MAX_MSG_SIZE 4096`

Variables

- `Trick::MemoryManager * trick_MM`
- `Trick::Integrator * trick_curr_integ`
- `Trick::MemoryManager * trick_MM`
- `Trick::MemoryManager * trick_MM`
- `Trick::MemoryManager * trick_MM`

6.3.1 Detailed Description

6.3.2 Macro Definition Documentation

6.3.2.1 CLASS

```
#define CLASS SimInterfaceMessages
```

Definition at line 37 of file `sim_interface_messages.cc`.

6.3.2.2 ER7_UTILS_ALWAYS_INLINE

```
#define ER7_UTILS_ALWAYS_INLINE
```

Definition at line 137 of file config.hh.

6.3.2.3 ER7_UTILS_RESTRICT

```
#define ER7_UTILS_RESTRICT
```

Definition at line 132 of file config.hh.

6.3.2.4 ER7_UTILS_UNUSED

```
#define ER7_UTILS_UNUSED
```

Definition at line 127 of file config.hh.

6.3.2.5 JEOD_ATTRIBUTES_POINTER_TYPE [1/2]

```
#define JEOD_ATTRIBUTES_POINTER_TYPE void *
```

Definition at line 78 of file config_test_harness.hh.

6.3.2.6 JEOD_ATTRIBUTES_POINTER_TYPE [2/2]

```
#define JEOD_ATTRIBUTES_POINTER_TYPE JEOD_ATTRIBUTES_TYPE *
```

Definition at line 102 of file config_trick10.hh.

6.3.2.7 JEOD_ATTRIBUTES_SIM_ENGINE_HEADER

```
#define JEOD_ATTRIBUTES_SIM_ENGINE_HEADER "sim_services/MemoryManager/include/attributes.h"
```

Definition at line 100 of file config_trick10.hh.

6.3.2.8 JEOD_ATTRIBUTES_TYPE [1/2]

```
#define JEOD_ATTRIBUTES_TYPE int
```

Definition at line 77 of file config_test_harness.hh.

6.3.2.9 JEOD_ATTRIBUTES_TYPE [2/2]

```
#define JEOD_ATTRIBUTES_TYPE struct ATTRIBUTES_tag
```

Definition at line 101 of file config_trick10.hh.

6.3.2.10 JEOD_CLASS_ESTABLISH_FRIENDS

```
#define JEOD_CLASS_ESTABLISH_FRIENDS(  
    ... ) JEODVMACRO(JEOD_CLASS_ESTABLISH_FRIENDS, __VA_ARGS__)
```

Definition at line 90 of file config_trick10.hh.

6.3.2.11 JEOD_CLASS_ESTABLISH_FRIENDS1

```
#define JEOD_CLASS_ESTABLISH_FRIENDS1(  
    class_name )
```

Value:

```
friend class InputProcessor;  
    \  
    friend void init_attr##class_name();
```

Definition at line 86 of file config_trick10.hh.

6.3.2.12 JEOD_CLASS_ESTABLISH_FRIENDS2

```
#define JEOD_CLASS_ESTABLISH_FRIENDS2(  
    ns,  
    class_name )
```

Value:

```
friend class InputProcessor;  
    \  
    friend void init_attr##ns##_##class_name()
```

Definition at line 82 of file config_trick10.hh.

6.3.2.13 JEOD_CLASS_ESTABLISH_FRIENDS3

```
#define JEOD_CLASS_ESTABLISH_FRIENDS3 (
    ns1,
    ns2,
    class_name )
```

Value:

```
friend class InputProcessor;
    \
    friend void init_attr##ns1##_##ns2##_##class_name()
```

Definition at line 78 of file config_trick10.hh.

6.3.2.14 JEOD_DECLARE_SIM_INTERFACES

```
#define JEOD_DECLARE_SIM_INTERFACES (
    class_name )
```

[JEOD_DECLARE_SIM_INTERFACES\(class_name\)](#) Forward declare classes and external functions needed to make the [JEOD_MAKE_SIM_INTERFACES\(class_name\)](#) expansion compilable.

All JEOD files that use JEOD_MAKE_SIM_INTERFACES within classes (will) make a parallel call to this macro at file scope in the global namespace.

Parameters

<i>class_name</i>	Name of the class defined later in the header in question.
-------------------	--

Definition at line 132 of file jeod_class.hh.

6.3.2.15 JEOD_INTPTR_T

```
#define JEOD_INTPTR_T long int
```

Definition at line 70 of file config_trick10.hh.

6.3.2.16 JEOD_MAKE_SIM_INTERFACES

```
#define JEOD_MAKE_SIM_INTERFACES (
    ... ) JEOD_CLASS_ESTABLISH_FRIENDS (__VA_ARGS__);
```

[JEOD_MAKE_SIM_INTERFACES\(class_name\)](#) Defines friends of the given class.

This macro is to be invoked in the body of all JEOD classes. The intent is to make all parts of the class visible to the designated simulation engine classes and functions.

Parameters

<i>class_name</i>	Name of the class being defined.
-------------------	----------------------------------

Definition at line 101 of file jeod_class.hh.

6.3.2.17 JEOD_PTRDIFF_T

```
#define JEOD_PTRDIFF_T long int
```

Definition at line 69 of file config_trick10.hh.

6.3.2.18 JEOD_SIM_INTEGRATOR_ENUM

```
#define JEOD_SIM_INTEGRATOR_ENUM Integrator_type
```

Definition at line 118 of file config_trick10.hh.

6.3.2.19 JEOD_SIM_INTEGRATOR_FORWARD

```
#define JEOD_SIM_INTEGRATOR_FORWARD
```

Value:

```
namespace Trick
{
    class Integrator;
}
```

Definition at line 112 of file config_trick10.hh.

6.3.2.20 JEOD_SIM_INTEGRATOR_POINTER_TYPE [1/2]

```
#define JEOD_SIM_INTEGRATOR_POINTER_TYPE void *
```

Definition at line 87 of file config_test_harness.hh.

6.3.2.21 JEOD_SIM_INTEGRATOR_POINTER_TYPE [2/2]

```
#define JEOD_SIM_INTEGRATOR_POINTER_TYPE Trick::Integrator *
```

Definition at line 117 of file config_trick10.hh.

6.3.2.22 JEOD_SIM_INTEGRATOR_SIM_ENGINE_HEADER

```
#define JEOD_SIM_INTEGRATOR_SIM_ENGINE_HEADER "sim_services/Integrator/include/Integrator.hh"
```

Definition at line 111 of file config_trick10.hh.

6.3.2.23 JEOD_SIZE_T

```
#define JEOD_SIZE_T size_t
```

Definition at line 68 of file config_trick10.hh.

6.3.2.24 JEOD_UINTPTR_T

```
#define JEOD_UINTPTR_T unsigned long int
```

Definition at line 71 of file config_trick10.hh.

6.3.2.25 JEOD_UNUSED

```
#define JEOD_UNUSED
```

Definition at line 122 of file config.hh.

6.3.2.26 JEODVAMCRO_SEQ_COUNT

```
#define JEODVAMCRO_SEQ_COUNT(  
    _1,  
    _2,  
    _3,  
    _4,  
    _5,  
    _6,  
    _7,  
    _8,  
    _9,  
    _10,  
    _11,  
    _12,  
    _13,  
    _14,  
    _15,  
    _16,  
    _17,  
    _18,  
    _19,  
    _20,  
    _21,  
    _22,  
    _23,  
    _24,  
    _25,  
    _26,  
    _27,  
    _28,  
    _29,  
    _30,  
    _31,  
    _32,  
    _33,  
    _34,  
    _35,  
    _36,  
    _37,  
    _38,  
    _39,  
    _40,  
    _41,  
    _42,  
    _43,  
    _44,  
    _45,  
    _46,  
    _47,  
    _48,  
    _49,  
    _50,  
    _51,  
    _52,  
    _53,  
    _54,  
    _55,
```

```

    _56,
    _57,
    _58,
    _59,
    _60,
    _61,
    _62,
    _63,
    N,
    ... ) N

```

Definition at line 66 of file jeod_va_macro_utility.hh.

6.3.2.27 JEODVMACRO

```

#define JEODVMACRO(
    func,
    ... ) JEODVMACRO_SELECT_FUNC(func, JEODVMACRO_VA_SIZE(__VA_ARGS__))(__VA_ARGS__↵
)

```

Definition at line 87 of file jeod_va_macro_utility.hh.

6.3.2.28 JEODVMACRO_GET_COUNT

```

#define JEODVMACRO_GET_COUNT(
    ... ) JEODVMACRO_SEQ_COUNT(__VA_ARGS__)

```

Definition at line 64 of file jeod_va_macro_utility.hh.

6.3.2.29 JEODVMACRO_REVSEQ_COUNT

```

#define JEODVMACRO_REVSEQ_COUNT( )

```

Value:

```

63, 62, 61, 60, \
  59, 58, 57, 56, 55, 54, 53, 52, 51, 50, \
  49, 48, 47, 46, 45, 44, 43, 42, 41, 40, \
  39, 38, 37, 36, 35, 34, 33, 32, 31, 30, \
  29, 28, 27, 26, 25, 24, 23, 22, 21, 20, \
  19, 18, 17, 16, 15, 14, 13, 12, 11, 10, \
  9, 8, 7, 6, 5, 4, 3, 2, 1, 0

```

Definition at line 74 of file jeod_va_macro_utility.hh.

6.3.2.30 JEODVMACRO_SELECT_FUNC

```
#define JEODVMACRO_SELECT_FUNC(  
    name,  
    n ) JEODVMACRO_SELECT_FUNC_CAT(name, n)
```

Definition at line 86 of file jeod_va_macro_utility.hh.

6.3.2.31 JEODVMACRO_SELECT_FUNC_CAT

```
#define JEODVMACRO_SELECT_FUNC_CAT(  
    name,  
    n ) name##n
```

Definition at line 85 of file jeod_va_macro_utility.hh.

6.3.2.32 JEODVMACRO_VA_SIZE

```
#define JEODVMACRO_VA_SIZE(  
    ... ) JEODVMACRO_GET_COUNT(__VA_ARGS__, JEODVMACRO_REVSEQ_COUNT())
```

Definition at line 63 of file jeod_va_macro_utility.hh.

6.3.2.33 MAKE_MESSAGE_CODE

```
#define MAKE_MESSAGE_CODE(  
    id ) char const * CLASS::id = PATH #id
```

Definition at line 38 of file sim_interface_messages.cc.

6.3.2.34 MAX_MSG_SIZE

```
#define MAX_MSG_SIZE 4096
```

Definition at line 44 of file trick_message_handler.cc.

Referenced by jeod::TrickMessageHandler::process_message().

6.3.2.35 PATH

```
#define PATH "utils/sim_interface/"
```

Definition at line 36 of file `sim_interface_messages.cc`.

6.3.3 Variable Documentation

6.3.3.1 `trick_curr_integ`

```
Trick::Integrator* trick_curr_integ
```

Referenced by `jeod::JeodDynbodyIntegrationLoop::integrate_dt()`.

6.3.3.2 `trick_MM` [1/4]

```
Trick::MemoryManager* trick_MM
```

6.3.3.3 `trick_MM` [2/4]

```
Trick::MemoryManager* trick_MM
```

Referenced by `jeod::JeodTrickMemoryInterface::deregister_allocation()`, and `jeod::JeodTrickMemoryInterface::register_allocation()`.

6.3.3.4 `trick_MM` [3/4]

```
Trick::MemoryManager* trick_MM
```

6.3.3.5 `trick_MM` [4/4]

```
Trick::MemoryManager* trick_MM
```

Referenced by `jeod::JeodTrick10MemoryInterface::JeodTrick10MemoryInterface()`.

Chapter 7

Namespace Documentation

7.1 er7_utils Namespace Reference

Namespace [er7_utils](#) contains the state integration models used by JEOD.

7.1.1 Detailed Description

Namespace [er7_utils](#) contains the state integration models used by JEOD.

7.2 jeod Namespace Reference

Namespace [jeod](#).

Data Structures

- class [BasicJeodTrickSimInterface](#)

The [BasicJeodTrickSimInterface](#) implements the required capabilities of the generic [JeodSimulationInterface](#) in a [Trick](#) simulation environment.

- class [CheckPointInputManager](#)

A [CheckPointInputManager](#) provides tools for reading a checkpoint file.

- class [CheckPointOutputManager](#)

A [CheckPointOutputManager](#) provides the basic tools for writing a checkpoint file.

- class [JeodDynbodyIntegrationLoop](#)

A [Trick::IntegLoopScheduler](#) that provides the ability to integrate a collection of [Trick::SimObject](#) instances over time, with the sim objects capable of being moved from one integration loop to another during run time.

- class [JeodIntegratorInterface](#)

A [JeodIntegratorInterface](#) extends the [ER7 IntegratorInterface](#) with the concept of a pointer to the simulation engine's integration object.

- class [JeodMemoryInterface](#)

Abstract interface between the JEOD memory manager and the simulation engine.

- class [JeodSimulationInterface](#)

This abstract class defines the basis for the interface between JEOD and a simulation engine.

- class [JeodSimulationInterfaceInit](#)

Define configuration data needed to configure the dynamically-created message handler and memory manager.

- class [JeodTrick10MemoryInterface](#)

A [TrickMemoryInterface](#) implements the two required methods needed to register and deregister memory with the simulation engine, [Trick](#) in this case.

- class [JeodTrickIntegrator](#)

A [JeodTrickIntegrator](#) specializes the [JeodIntegratorInterface](#) for use with [Trick](#) as the simulation engine.

- class [JeodTrickMemoryInterface](#)

A [TrickMemoryInterface](#) implements the two required methods needed to register and deregister memory with the simulation engine, [Trick](#) in this case.

- class [JeodTrickSimInterface](#)

A [JEODTrickSimInterface](#) implements the required capabilities of the generic [JeodSimulationInterface](#) in a [Trick](#) simulation environment.

- class [SectionedInputBuffer](#)

A [SectionedInputBuffer](#) is a `std::streambuf` that reads from a section in a checkpoint file.

- class [SectionedInputStream](#)

A [SectionedInputStream](#) is a `std::istream` that reads from a section in a checkpoint file.

- class [SectionedOutputBuffer](#)

A [SectionedOutputBuffer](#) is a `std::streambuf` that writes a section of a checkpoint file.

- class [SectionedOutputStream](#)

A [SectionedOutputStream](#) is a `std::ostream` that writes a section of a checkpoint file.

- class [SimInterfaceMessages](#)

Specifies the message IDs used in the `sim_interface` model.

- class [TrickJeodIntegrator](#)

A [TrickJeodIntegrator](#) specializes the `Trick::Integrator` to provide the [Trick](#) side of the integration interface between [Trick](#) and JEOD.

- class [TrickMessageHandler](#)

The `MessageHandler` class for designed for use in [Trick](#)-based simulations.

- class [TrickMessageHandlerMixin](#)

The [TrickMessageHandlerMixin](#) implements the required capabilities of the generic [JeodSimulationInterface](#) in a [Trick](#) simulation environment.

7.2.1 Detailed Description

Namespace [jeod](#).

7.3 Trick Namespace Reference

Namespace [Trick](#) furnishes several standard functions for use in the [Trick](#) environment.

7.3.1 Detailed Description

Namespace [Trick](#) furnishes several standard functions for use in the [Trick](#) environment.

Chapter 8

Data Structure Documentation

8.1 jeod::JeodTrickMemoryInterface::AllocationMapEntry Struct Reference

Describes a chunk of JEOD-allocated memory.

```
#include <trick_memory_interface.hh>
```

Public Member Functions

- [AllocationMapEntry](#) (const std::type_info &type_info, uint32_t nelem, bool arrayp)
Construct an [AllocationMapEntry](#) object.

Data Fields

- const std::type_info & [typeid_info](#)
Descriptor of the data type.
- uint32_t [nelements](#)
The number of elements in the allocated chunk of memory.
- bool [is_array](#)
Is the item an array or a single object?

8.1.1 Detailed Description

Describes a chunk of JEOD-allocated memory.

Definition at line 269 of file trick_memory_interface.hh.

8.1.2 Constructor & Destructor Documentation

8.1.2.1 AllocationMapEntry()

```
jeod::JeodTrickMemoryInterface::AllocationMapEntry::AllocationMapEntry (  
    const std::type_info & type_info,  
    uint32_t nelem,  
    bool arrayp ) [inline]
```

Construct an [AllocationMapEntry](#) object.

Parameters

<i>type_info</i>	Type info
<i>nelem</i>	Array size
<i>arrayp</i>	Is item an array?

Definition at line 292 of file `trick_memory_interface.hh`.

8.1.3 Field Documentation**8.1.3.1 is_array**

```
bool jeod::JeodTrickMemoryInterface::AllocationMapEntry::is_array
```

Is the item an array or a single object?

`trick_units(-)`

Definition at line 284 of file `trick_memory_interface.hh`.

Referenced by `jeod::JeodTrick10MemoryInterface::checkpoint_allocations()`.

8.1.3.2 nelements

```
uint32_t jeod::JeodTrickMemoryInterface::AllocationMapEntry::nelements
```

The number of elements in the allocated chunk of memory.

`trick_units(-)`

Definition at line 279 of file `trick_memory_interface.hh`.

Referenced by `jeod::JeodTrick10MemoryInterface::checkpoint_allocations()`.

8.1.3.3 typeid_info

```
const std::type_info& jeod::JeodTrickMemoryInterface::AllocationMapEntry::typeid_info
```

Descriptor of the data type.

`trick_units(-)`

Definition at line 274 of file `trick_memory_interface.hh`.

Referenced by `jeod::JeodTrick10MemoryInterface::checkpoint_allocations()`.

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

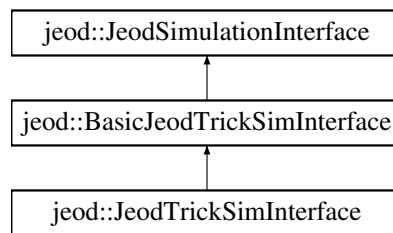
- [trick_memory_interface.hh](#)

8.2 jeod::BasicJeodTrickSimInterface Class Reference

The [BasicJeodTrickSimInterface](#) implements the required capabilities of the generic [JeodSimulationInterface](#) in a [Trick](#) simulation environment.

```
#include <trick_sim_interface.hh>
```

Inheritance diagram for jeod::BasicJeodTrickSimInterface:



Public Member Functions

- [BasicJeodTrickSimInterface](#) (MessageHandler &message_handler)
Construct a [BasicJeodTrickSimInterface](#) object.
- [~BasicJeodTrickSimInterface](#) () override
Destroy a [BasicJeodTrickSimInterface](#) object.
- void [set_checkpoint_file_name](#) (const std::string &name)
Set the checkpoint file name.
- std::string [get_checkpoint_file_name](#) () const
Get the checkpoint file name.
- void [set_mode](#) (JeodSimulationInterface::Mode new_mode) override
Set the mode.
- void [checkpoint_allocations](#) ()
Dump the allocation information to the checkpoint file.
- void [restore_allocations](#) ()
Restore the allocated data per the checkpoint file.
- void [checkpoint_containers](#) ()
Dump the container objects to the checkpoint file.
- void [restore_containers](#) ()
Restore the container objects from the checkpoint file.
- void [open_checkpoint_file](#) ()
Open the checkpoint output file.
- void [close_checkpoint_file](#) ()
Close the checkpoint output file.
- void [open_restart_file](#) ()
Open the checkpoint input file.
- void [close_restart_file](#) ()
Close the checkpoint input file.
- [BasicJeodTrickSimInterface](#) (const [BasicJeodTrickSimInterface](#) &)=delete
- [BasicJeodTrickSimInterface](#) & operator= (const [BasicJeodTrickSimInterface](#) &)=delete

Protected Member Functions

- [JeodIntegratorInterface](#) * [create_integrator_internal](#) () override
Create an integration interface object.
- double [get_job_cycle_internal](#) () override
Get the current job's cycle time.
- [JeodMemoryInterface](#) & [get_memory_interface_internal](#) () override
Get the memory interface.
- [SectionedInputStream](#) [get_checkpoint_reader_internal](#) (const std::string §ion_id) override
Get a reader to a section of the currently open checkpoint file.
- [SectionedOutputStream](#) [get_checkpoint_writer_internal](#) (const std::string §ion_id) override
Get a writer to a section of the currently open checkpoint file.

Protected Attributes

- MessageHandler & [generic_message_handler](#)
The global MessageHandler.
- [JeodTrick10MemoryInterface](#) [trick_memory_interface](#)
The interface between JEOD and [Trick](#)'s memory management schemes.
- JeodMemoryManager [memory_manager](#)
The global JEOD memory manager.
- std::string [checkpoint_file_name](#)
The name of the segmented checkpoint file used for the next checkpoint / restart action.
- std::string [section_start](#)
String indicating the start of a checkpoint file section.
- std::string [section_end](#)
String indicating the end of a checkpoint file section.
- [CheckPointInputManager](#) * [checkpoint_reader](#) {}
The object that manages reading from a checkpoint file.
- [CheckPointOutputManager](#) * [checkpoint_writer](#) {}
The object that manages writing to a checkpoint file.

Friends

- class [InputProcessor](#)
- void [init_atrjeod__BasicJeodTrickSimInterface](#) ()

Additional Inherited Members

8.2.1 Detailed Description

The [BasicJeodTrickSimInterface](#) implements the required capabilities of the generic [JeodSimulationInterface](#) in a [Trick](#) simulation environment.

By virtue of member data ownership, the class creates the requisite MessageHandler and MemoryManager and does so in the correct order.

Definition at line 89 of file [trick_sim_interface.hh](#).

8.2.2 Constructor & Destructor Documentation

8.2.2.1 BasicJeodTrickSimInterface() [1/2]

```
jeod::BasicJeodTrickSimInterface::BasicJeodTrickSimInterface (
    MessageHandler & message_handler ) [explicit]
```

Construct a [BasicJeodTrickSimInterface](#) object.

Parameters

<i>in, out</i>	<i>message_handler</i>	handler Units: Message
----------------	------------------------	---------------------------

Definition at line 61 of file `trick_sim_interface.cc`.

References `generic_message_handler`, `section_end`, and `section_start`.

8.2.2.2 ~BasicJeodTrickSimInterface()

```
jeod::BasicJeodTrickSimInterface::~~BasicJeodTrickSimInterface ( ) [override]
```

Destroy a [BasicJeodTrickSimInterface](#) object.

Definition at line 89 of file `trick_sim_interface.cc`.

References `checkpoint_reader`, and `checkpoint_writer`.

8.2.2.3 BasicJeodTrickSimInterface() [2/2]

```
jeod::BasicJeodTrickSimInterface::BasicJeodTrickSimInterface (
    const BasicJeodTrickSimInterface & ) [delete]
```

8.2.3 Member Function Documentation

8.2.3.1 checkpoint_allocations()

```
void jeod::BasicJeodTrickSimInterface::checkpoint_allocations ( )
```

Dump the allocation information to the checkpoint file.

Definition at line 284 of file trick_sim_interface.cc.

References [jeod::JeodTrick10MemoryInterface::checkpoint_allocations\(\)](#), [jeod::JeodTrick10MemoryInterface::is_checkpoint_restart_supported\(\)](#), and [trick_memory_interface](#).

8.2.3.2 checkpoint_containers()

```
void jeod::BasicJeodTrickSimInterface::checkpoint_containers ( )
```

Dump the container objects to the checkpoint file.

Definition at line 306 of file trick_sim_interface.cc.

References [jeod::JeodTrick10MemoryInterface::checkpoint_containers\(\)](#), [jeod::JeodTrick10MemoryInterface::is_checkpoint_restart_supported\(\)](#), and [trick_memory_interface](#).

8.2.3.3 close_checkpoint_file()

```
void jeod::BasicJeodTrickSimInterface::close_checkpoint_file ( )
```

Close the checkpoint output file.

Definition at line 205 of file trick_sim_interface.cc.

References [checkpoint_writer](#).

8.2.3.4 close_restart_file()

```
void jeod::BasicJeodTrickSimInterface::close_restart_file ( )
```

Close the checkpoint input file.

Definition at line 275 of file trick_sim_interface.cc.

References [checkpoint_reader](#).

8.2.3.5 create_integrator_internal()

```
JeodIntegratorInterface * jeod::BasicJeodTrickSimInterface::create_integrator_internal ( )
[override], [protected], [virtual]
```

Create an integration interface object.

Returns

Integrator interface that encapsulates an sim engine integrator.

Implements [jeod::JeodSimulationInterface](#).

Definition at line 118 of file `trick_sim_interface.cc`.

8.2.3.6 get_checkpoint_file_name()

```
std::string jeod::BasicJeodTrickSimInterface::get_checkpoint_file_name ( ) const [inline]
```

Get the checkpoint file name.

Definition at line 110 of file `trick_sim_interface.hh`.

8.2.3.7 get_checkpoint_reader_internal()

```
SectionedInputStream jeod::BasicJeodTrickSimInterface::get_checkpoint_reader_internal (
    const std::string & section_id ) [override], [protected], [virtual]
```

Get a reader to a section of the currently open checkpoint file.

Returns

Checkpoint reader

Parameters

in	<i>section_id</i>	Section name
----	-------------------	--------------

Implements [jeod::JeodSimulationInterface](#).

Definition at line 256 of file `trick_sim_interface.cc`.

References `checkpoint_reader`, `jeod::CheckPointInputManager::create_section_reader()`, and `jeod::SimInterface↔ Messages::phasing_error`.

8.2.3.8 get_checkpoint_writer_internal()

```
SectionedOutputStream jeod::BasicJeodTrickSimInterface::get_checkpoint_writer_internal (
    const std::string & section_id ) [override], [protected], [virtual]
```

Get a writer to a section of the currently open checkpoint file.

Returns

Checkpoint writer

Parameters

in	<i>section_id</i>	Section name
----	-------------------	--------------

Implements [jeod::JeodSimulationInterface](#).

Definition at line 186 of file `trick_sim_interface.cc`.

References `checkpoint_writer`, `jeod::CheckPointOutputManager::create_section_writer()`, and `jeod::SimInterface::Messages::phasing_error`.

8.2.3.9 get_job_cycle_internal()

```
double jeod::BasicJeodTrickSimInterface::get_job_cycle_internal ( ) [override], [protected],
[virtual]
```

Get the current job's cycle time.

Returns

Current job's cycle time
Units: s

Implements [jeod::JeodSimulationInterface](#).

Definition at line 127 of file `trick_sim_interface.cc`.

8.2.3.10 get_memory_interface_internal()

```
JeodMemoryInterface & jeod::BasicJeodTrickSimInterface::get_memory_interface_internal ( )
[override], [protected], [virtual]
```

Get the memory interface.

Returns

Memory interface

Implements [jeod::JeodSimulationInterface](#).

Definition at line 136 of file `trick_sim_interface.cc`.

References `trick_memory_interface`.

8.2.3.11 open_checkpoint_file()

```
void jeod::BasicJeodTrickSimInterface::open_checkpoint_file ( )
```

Open the checkpoint output file.

Definition at line 144 of file `trick_sim_interface.cc`.

References `checkpoint_file_name`, `checkpoint_writer`, `jeod::JeodTrick10MemoryInterface::get_trick_checkpoint_file()`, `jeod::JeodTrick10MemoryInterface::is_checkpoint_restart_supported()`, `section_end`, `section_start`, and `trick_memory_interface`.

8.2.3.12 open_restart_file()

```
void jeod::BasicJeodTrickSimInterface::open_restart_file ( )
```

Open the checkpoint input file.

Definition at line 214 of file `trick_sim_interface.cc`.

References `checkpoint_file_name`, `checkpoint_reader`, `jeod::JeodTrick10MemoryInterface::get_trick_checkpoint_file()`, `jeod::JeodTrick10MemoryInterface::is_checkpoint_restart_supported()`, `section_end`, `section_start`, and `trick_memory_interface`.

8.2.3.13 operator=()

```
BasicJeodTrickSimInterface& jeod::BasicJeodTrickSimInterface::operator= (
    const BasicJeodTrickSimInterface & ) [delete]
```

8.2.3.14 restore_allocations()

```
void jeod::BasicJeodTrickSimInterface::restore_allocations ( )
```

Restore the allocated data per the checkpoint file.

Definition at line 295 of file `trick_sim_interface.cc`.

References `jeod::JeodTrick10MemoryInterface::is_checkpoint_restart_supported()`, `memory_manager`, `jeod::JeodTrick10MemoryInterface::restore_allocations()`, and `trick_memory_interface`.

8.2.3.15 restore_containers()

```
void jeod::BasicJeodTrickSimInterface::restore_containers ( )
```

Restore the container objects from the checkpoint file.

Definition at line 317 of file `trick_sim_interface.cc`.

References `jeod::JeodTrick10MemoryInterface::is_checkpoint_restart_supported()`, `jeod::JeodTrick10MemoryInterface::restore_containers()`, and `trick_memory_interface`.

8.2.3.16 set_checkpoint_file_name()

```
void jeod::BasicJeodTrickSimInterface::set_checkpoint_file_name (
    const std::string & name ) [inline]
```

Set the checkpoint file name.

Definition at line 102 of file `trick_sim_interface.hh`.

8.2.3.17 set_mode()

```
void jeod::BasicJeodTrickSimInterface::set_mode (
    JeodSimulationInterface::Mode new_mode ) [override], [virtual]
```

Set the mode.

Assumptions and Limitations

- See `SimulationInterface::set_mode`.

Parameters

in	<i>new_mode</i>	New mode.
----	-----------------	-----------

Reimplemented from [jeod::JeodSimulationInterface](#).

Definition at line 104 of file `trick_sim_interface.cc`.

References `generic_message_handler`, `jeod::JeodSimulationInterface::get_mode()`, `memory_manager`, `jeod::JeodTrickMemoryInterface::set_mode()`, `jeod::JeodSimulationInterface::set_mode()`, and `trick_memory_interface`.

8.2.4 Friends And Related Function Documentation

8.2.4.1 init_attrjeod__BasicJeodTrickSimInterface

```
void init_attrjeod__BasicJeodTrickSimInterface ( ) [friend]
```

8.2.4.2 InputProcessor

```
friend class InputProcessor [friend]
```

Definition at line 91 of file trick_sim_interface.hh.

8.2.5 Field Documentation

8.2.5.1 checkpoint_file_name

```
std::string jeod::BasicJeodTrickSimInterface::checkpoint_file_name [protected]
```

The name of the segmented checkpoint file used for the next checkpoint / restart action.

If the name is the empty string (default), the checkpoint / restart mechanisms will attempt to construct a name from the corresponding [Trick](#) checkpoint file name.trick_units(-)

Definition at line 197 of file trick_sim_interface.hh.

Referenced by open_checkpoint_file(), and open_restart_file().

8.2.5.2 checkpoint_reader

```
CheckPointInputManager* jeod::BasicJeodTrickSimInterface::checkpoint_reader {} [protected]
```

The object that manages reading from a checkpoint file.

```
trick_io(**)
```

Definition at line 212 of file trick_sim_interface.hh.

Referenced by close_restart_file(), get_checkpoint_reader_internal(), open_restart_file(), and ~BasicJeodTrickSimInterface().

8.2.5.3 checkpoint_writer

```
CheckpointOutputManager* jeod::BasicJeodTrickSimInterface::checkpoint_writer {} [protected]
```

The object that manages writing to a checkpoint file.

trick_io(**)

Definition at line 217 of file trick_sim_interface.hh.

Referenced by close_checkpoint_file(), get_checkpoint_writer_internal(), open_checkpoint_file(), and ~BasicJeodTrickSimInterface().

8.2.5.4 generic_message_handler

```
MessageHandler& jeod::BasicJeodTrickSimInterface::generic_message_handler [protected]
```

The global MessageHandler.

trick_units(-)

Definition at line 178 of file trick_sim_interface.hh.

Referenced by BasicJeodTrickSimInterface(), and set_mode().

8.2.5.5 memory_manager

```
JeodMemoryManager jeod::BasicJeodTrickSimInterface::memory_manager [protected]
```

The global JEOD memory manager.

trick_units(-)

Definition at line 188 of file trick_sim_interface.hh.

Referenced by restore_allocations(), and set_mode().

8.2.5.6 section_end

```
std::string jeod::BasicJeodTrickSimInterface::section_end [protected]
```

String indicating the end of a checkpoint file section.

trick_io(*o) trick_units(-)

Definition at line 207 of file trick_sim_interface.hh.

Referenced by BasicJeodTrickSimInterface(), open_checkpoint_file(), and open_restart_file().

8.2.5.7 section_start

```
std::string jeod::BasicJeodTrickSimInterface::section_start [protected]
```

String indicating the start of a checkpoint file section.

```
trick_io(*o) trick_units(-)
```

Definition at line 202 of file `trick_sim_interface.hh`.

Referenced by `BasicJeodTrickSimInterface()`, `open_checkpoint_file()`, and `open_restart_file()`.

8.2.5.8 trick_memory_interface

```
JeodTrick10MemoryInterface jeod::BasicJeodTrickSimInterface::trick_memory_interface [protected]
```

The interface between JEOD and [Trick](#)'s memory management schemes.

```
trick_units(-)
```

Definition at line 183 of file `trick_sim_interface.hh`.

Referenced by `checkpoint_allocations()`, `checkpoint_containers()`, `get_memory_interface_internal()`, `open_checkpoint_file()`, `open_restart_file()`, `restore_allocations()`, `restore_containers()`, and `set_mode()`.

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

- [trick_sim_interface.hh](#)
- [trick_sim_interface.cc](#)

8.3 jeod::CheckpointInputManager Class Reference

A [CheckpointInputManager](#) provides tools for reading a checkpoint file.

```
#include <checkpoint_input_manager.hh>
```

Data Structures

- struct [SectionInfo](#)

A [SectionInfo](#) contains the start and end positions of a checkpoint file section.

Public Member Functions

- [CheckpointInputManager](#) (const std::string &fname, const std::string &start_marker, const std::string &end↵_marker)
Construct a [CheckpointInputManager](#) object.
- [CheckpointInputManager](#) (const [CheckpointInputManager](#) &)=delete
- [CheckpointInputManager](#) & operator= (const [CheckpointInputManager](#) &)=delete
- [SectionedInputStream](#) create_section_reader (const std::string &tag)
Create a C++ input stream that reads from a checkpoint file section.
- bool [operator!](#) () const
Conversion to boolean.
- bool [have_active_reader](#) () const
Is there an active checkpoint section reader?
- bool [register_reader](#) ([SectionedInputStream](#) *reader)
Register the supplied section reader as the currently-active reader.
- bool [deregister_reader](#) (const [SectionedInputStream](#) *reader)
Deregister the supplied section reader as the currently-active reader.

Private Member Functions

- void [initialize](#) ()
Determine the locations of the various sections that comprise the file.
- [SectionedInputStream](#) create_section_reader (bool trick, const std::string &tag)
Create a C++ input stream that reads from a checkpoint file section.
- [SectionedInputStream](#) create_trick_section_reader ()
Create a C++ input stream that reads from the *Trick* checkpoint file section.

Private Attributes

- std::map< std::string, [SectionInfo](#) > [sections](#)
Maps section names to section start/end positions.
- std::ifstream [stream](#)
The C++ file stream that reads the checkpoint file.
- [SectionedInputStream](#) * [current_reader](#) {}
The reader that currently is active.
- const std::string [filename](#)
The name of the checkpoint file.
- const std::string & [section_start](#)
The string that indicates the start of a checkpoint file section.
- const std::string & [section_end](#)
The string that indicates the start of a checkpoint file section.
- bool [is_open](#) {true}
Is the checkpoint file open?

8.3.1 Detailed Description

A [CheckPointInputManager](#) provides tools for reading a checkpoint file.

A [Trick](#) 10 checkpoint file comprises multiple sections delineated by section markers. This class recognizes those markers and generates C++ input streams that other objects can use to read the contents of one of those checkpoint file sections. The interpretation of the contents of a checkpoint file section is the responsibility of those other objects.

Definition at line 404 of file `checkpoint_input_manager.hh`.

8.3.2 Constructor & Destructor Documentation

8.3.2.1 CheckPointInputManager() [1/2]

```
jeod::CheckPointInputManager::CheckPointInputManager (
    const std::string & fname,
    const std::string & start_marker,
    const std::string & end_marker )
```

Construct a [CheckPointInputManager](#) object.

Parameters

in	<i>fname</i>	Name of file to be opened
in	<i>start_marker</i>	Start of section marker
in	<i>end_marker</i>	End of section marker

Definition at line 277 of file `checkpoint_input_manager.cc`.

References `filename`, `jeod::SimInterfaceMessages::implementation_error`, `initialize()`, `is_open`, and `stream`.

8.3.2.2 CheckPointInputManager() [2/2]

```
jeod::CheckPointInputManager::CheckPointInputManager (
    const CheckPointInputManager & ) [delete]
```

8.3.3 Member Function Documentation

8.3.3.1 `create_section_reader()` [1/2]

```
SectionedInputStream jeod::CheckPointInputManager::create_section_reader (
    const std::string & tag ) [inline]
```

Create a C++ input stream that reads from a checkpoint file section.

Error handling

A null [SectionedInputStream](#) is created when the [CheckPointInputManager](#) itself is invalid or when the designated section is not present in the checkpoint file.

Parameters

<i>tag</i>	Tag that identifies the section to be read.
------------	---

Returns

A [SectionedInputStream](#) object, which must be used to initialize a local [SectionedInputStream](#) variable.

Definition at line 423 of file `checkpoint_input_manager.hh`.

Referenced by `create_trick_section_reader()`, and `jeod::BasicJeodTrickSimInterface::get_checkpoint_reader_↔internal()`.

8.3.3.2 `create_section_reader()` [2/2]

```
SectionedInputStream jeod::CheckPointInputManager::create_section_reader (
    bool trick,
    const std::string & tag ) [private]
```

Create a C++ input stream that reads from a checkpoint file section.

Usage

Use this function as the initializer of a section reader variable.

Error handling

A null [SectionedInputStream](#) is created when the [CheckPointInputManager](#) itself is invalid or when the designated section is not present in the checkpoint file.

Returns

A [SectionedInputStream](#) object.

Parameters

in	<i>trick</i>	OK to create the Trick section reader?
in	<i>tag</i>	Tag identifying the section to be read.

Definition at line 389 of file `checkpoint_input_manager.cc`.

References `jeod::CheckPointInputManager::SectionInfo::end_pos`, `filename`, `jeod::SimInterfaceMessages::implementation_error`, `is_open`, `sections`, `jeod::CheckPointInputManager::SectionInfo::start_pos`, and `stream`.

8.3.3.3 `create_trick_section_reader()`

```
SectionedInputStream jeod::CheckPointInputManager::create_trick_section_reader ( ) [private]
```

Create a C++ input stream that reads from the [Trick](#) checkpoint file section.

Returns

[Trick SectionedInputStream](#) object.

Definition at line 441 of file `checkpoint_input_manager.cc`.

References `create_section_reader()`, `current_reader`, and `jeod::SectionedInputStream::deactivate()`.

8.3.3.4 `deregister_reader()`

```
bool jeod::CheckPointInputManager::deregister_reader (
    const SectionedInputStream * reader )
```

Deregister the supplied section reader as the currently-active reader.

Returns

True => success.

Parameters

in	<i>reader</i>	Reader to be deregistered
----	---------------	---------------------------

Definition at line 475 of file `checkpoint_input_manager.cc`.

References `current_reader`.

Referenced by `jeod::SectionedInputStream::deactivate()`, and `jeod::SectionedInputStream::~~SectionedInputStream()`.

8.3.3.5 have_active_reader()

```
bool jeod::CheckPointInputManager::have_active_reader ( ) const [inline]
```

Is there an active checkpoint section reader?

Returns

True if there is an active reader, false otherwise.

Definition at line 443 of file checkpoint_input_manager.hh.

References `current_reader`.

Referenced by `jeod::SectionedInputStream::is_activatable()`.

8.3.3.6 initialize()

```
void jeod::CheckPointInputManager::initialize ( ) [private]
```

Determine the locations of the various sections that comprise the file.

Definition at line 303 of file checkpoint_input_manager.cc.

References `filename`, `jeod::SimInterfaceMessages::implementation_error`, `section_end`, `section_start`, `sections`, and `stream`.

Referenced by `CheckPointInputManager()`.

8.3.3.7 operator!()

```
bool jeod::CheckPointInputManager::operator! ( ) const [inline]
```

Conversion to boolean.

Returns

False if object is OK.

Definition at line 433 of file checkpoint_input_manager.hh.

References `is_open`, and `stream`.

8.3.3.8 operator=()

```
CheckPointInputManager& jeod::CheckPointInputManager::operator= (
    const CheckPointInputManager & ) [delete]
```

8.3.3.9 register_reader()

```
bool jeod::CheckPointInputManager::register_reader (
    SectionedInputStream * reader )
```

Register the supplied section reader as the currently-active reader.

Returns

True => success.

Parameters

in	<i>reader</i>	Reader to be registered
----	---------------	-------------------------

Definition at line 457 of file checkpoint_input_manager.cc.

References `current_reader`.

Referenced by `jeod::SectionedInputStream::activate()`.

8.3.4 Field Documentation

8.3.4.1 `current_reader`

```
SectionedInputStream* jeod::CheckPointInputManager::current_reader {} [private]
```

The reader that currently is active.

`trick_io(**)`

Definition at line 511 of file checkpoint_input_manager.hh.

Referenced by `create_trick_section_reader()`, `deregister_reader()`, `have_active_reader()`, and `register_reader()`.

8.3.4.2 `filename`

```
const std::string jeod::CheckPointInputManager::filename [private]
```

The name of the checkpoint file.

Definition at line 516 of file checkpoint_input_manager.hh.

Referenced by `CheckPointInputManager()`, `create_section_reader()`, and `initialize()`.

8.3.4.3 `is_open`

```
bool jeod::CheckPointInputManager::is_open {true} [private]
```

Is the checkpoint file open?

`trick_io(**)`

Definition at line 531 of file checkpoint_input_manager.hh.

Referenced by `CheckPointInputManager()`, `create_section_reader()`, and `operator!()`.

8.3.4.4 section_end

```
const std::string& jeod::CheckPointInputManager::section_end [private]
```

The string that indicates the start of a checkpoint file section.

Definition at line 526 of file checkpoint_input_manager.hh.

Referenced by initialize().

8.3.4.5 section_start

```
const std::string& jeod::CheckPointInputManager::section_start [private]
```

The string that indicates the start of a checkpoint file section.

Definition at line 521 of file checkpoint_input_manager.hh.

Referenced by initialize().

8.3.4.6 sections

```
std::map<std::string, SectionInfo> jeod::CheckPointInputManager::sections [private]
```

Maps section names to section start/end positions.

trick_io(**)

Definition at line 501 of file checkpoint_input_manager.hh.

Referenced by create_section_reader(), and initialize().

8.3.4.7 stream

```
std::ifstream jeod::CheckPointInputManager::stream [private]
```

The C++ file stream that reads the checkpoint file.

trick_io(**)

Definition at line 506 of file checkpoint_input_manager.hh.

Referenced by CheckPointInputManager(), create_section_reader(), initialize(), and operator!().

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

- [checkpoint_input_manager.hh](#)
- [checkpoint_input_manager.cc](#)

8.4 jeod::CheckPointOutputManager Class Reference

A [CheckPointOutputManager](#) provides the basic tools for writing a checkpoint file.

```
#include <checkpoint_output_manager.hh>
```

Public Member Functions

- [CheckPointOutputManager](#) (const std::string &fname, const std::string &start_marker, const std::string &end_marker)

Construct a [CheckPointOutputManager](#) object.

- [CheckPointOutputManager](#) (const [CheckPointOutputManager](#) &)=delete
- [CheckPointOutputManager](#) & operator= (const [CheckPointOutputManager](#) &)=delete
- [SectionedOutputStream](#) create_section_writer (const std::string &tag)

Create a C++ output stream that writes a checkpoint file section.

- bool operator! () const

Conversion to boolean.

- bool have_active_writer () const

Is there an active checkpoint section writer?

- bool register_writer ([SectionedOutputStream](#) *writer)

Register the supplied section writer as the currently-active writer.

- bool deregister_writer (const [SectionedOutputStream](#) *writer)

Deregister the supplied section writer as the currently-active writer.

Private Member Functions

- [SectionedOutputStream](#) create_section_writer (bool trick, const std::string &tag)

Create a C++ output stream that writes to a checkpoint file section.

- [SectionedOutputStream](#) create_trick_section_writer ()

Create a C++ output stream that writes a [Trick](#) checkpoint file section.

Private Attributes

- std::ofstream stream

The C++ file stream that writes to the checkpoint file.

- [SectionedOutputStream](#) * current_writer {}

The writer that currently is active.

- const std::string filename

The name of the checkpoint file.

- const std::string & section_start

The string that indicates the start of a checkpoint file section.

- const std::string & section_end

The string that indicates the start of a checkpoint file section.

- bool is_open {true}

Is the checkpoint file open?

Friends

- class [MemoryManagerWrapper](#)

8.4.1 Detailed Description

A [CheckPointOutputManager](#) provides the basic tools for writing a checkpoint file.

Section markers split a [Trick](#) 10 checkpoint file into multiple parts. This class generates C++ output streams that write the section markers and that other objects can use to write checkpoint file section data.

Definition at line 269 of file `checkpoint_output_manager.hh`.

8.4.2 Constructor & Destructor Documentation

8.4.2.1 CheckPointOutputManager() [1/2]

```
jeod::CheckPointOutputManager::CheckPointOutputManager (
    const std::string & fname,
    const std::string & start_marker,
    const std::string & end_marker )
```

Construct a [CheckPointOutputManager](#) object.

Parameters

in	<i>fname</i>	Name of file to be opened
in	<i>start_marker</i>	Start of section marker
in	<i>end_marker</i>	End of section marker

Definition at line 289 of file `checkpoint_output_manager.cc`.

References `filename`, `jeod::SimInterfaceMessages::implementation_error`, `is_open`, and `stream`.

8.4.2.2 CheckPointOutputManager() [2/2]

```
jeod::CheckPointOutputManager::CheckPointOutputManager (
    const CheckPointOutputManager & ) [delete]
```

8.4.3 Member Function Documentation

8.4.3.1 create_section_writer() [1/2]

```
SectionedOutputStream jeod::CheckPointOutputManager::create_section_writer (
    const std::string & tag ) [inline]
```

Create a C++ output stream that writes a checkpoint file section.

Returns

Constructed [SectionedOutputStream](#).

Definition at line 286 of file checkpoint_output_manager.hh.

Referenced by [create_trick_section_writer\(\)](#), and [jeod::BasicJeodTrickSimInterface::get_checkpoint_writer_↔internal\(\)](#).

8.4.3.2 create_section_writer() [2/2]

```
SectionedOutputStream jeod::CheckPointOutputManager::create_section_writer (
    bool trick,
    const std::string & tag ) [private]
```

Create a C++ output stream that writes to a checkpoint file section.

Usage

Use this function as the initializer of a section writer variable.

Error handling

A null [SectionedOutputStream](#) is created when the [CheckPointOutputManager](#) itself is invalid or the designated section is invalid.

Returns

A [SectionedOutputStream](#) object.

Parameters

in	<i>trick</i>	OK to create the Trick section writer?
in	<i>tag</i>	Tag identifying the section to be written.

Definition at line 319 of file checkpoint_output_manager.cc.

References [filename](#), [jeod::SimInterfaceMessages::implementation_error](#), [is_open](#), [section_end](#), [section_start](#), and [stream](#).

8.4.3.3 create_trick_section_writer()

```
SectionedOutputStream jeod::CheckPointOutputManager::create_trick_section_writer ( ) [private]
```

Create a C++ output stream that writes a [Trick](#) checkpoint file section.

Create a C++ output stream that writes to the [Trick](#) checkpoint file section.

Returns

A [SectionedOutputStream](#) object, which must be used to initialize a local [SectionedOutputStream](#) variable.
[Trick SectionedOutputStream](#) object.

Definition at line 357 of file checkpoint_output_manager.cc.

References [create_section_writer\(\)](#), [current_writer](#), and [jeod::SectionedOutputStream::deactivate\(\)](#).

8.4.3.4 deregister_writer()

```
bool jeod::CheckPointOutputManager::deregister_writer (
    const SectionedOutputStream * writer )
```

Deregister the supplied section writer as the currently-active writer.

Returns

True => success.

Parameters

in	<i>writer</i>	Writer to be deregistered
----	---------------	---------------------------

Definition at line 391 of file checkpoint_output_manager.cc.

References [current_writer](#).

Referenced by [jeod::SectionedOutputStream::deactivate\(\)](#).

8.4.3.5 have_active_writer()

```
bool jeod::CheckPointOutputManager::have_active_writer ( ) const [inline]
```

Is there an active checkpoint section writer?

Returns

True if there is an active writer, false otherwise.

Definition at line 306 of file checkpoint_output_manager.hh.

References `current_writer`.

Referenced by `jeod::SectionedOutputStream::is_activatable()`.

8.4.3.6 operator!()

```
bool jeod::CheckPointOutputManager::operator! ( ) const [inline]
```

Conversion to boolean.

Returns

False if object is OK.

Definition at line 296 of file checkpoint_output_manager.hh.

References `is_open`, and `stream`.

8.4.3.7 operator=()

```
CheckPointOutputManager& jeod::CheckPointOutputManager::operator= (
    const CheckPointOutputManager & ) [delete]
```

8.4.3.8 register_writer()

```
bool jeod::CheckPointOutputManager::register_writer (
    SectionedOutputStream * writer )
```

Register the supplied section writer as the currently-active writer.

Returns

True => success.

Parameters

in	<i>writer</i>	Writer to be Registered
----	---------------	-------------------------

Definition at line 373 of file checkpoint_output_manager.cc.

References `current_writer`.

Referenced by `jeod::SectionedOutputStream::activate()`.

8.4.4 Friends And Related Function Documentation

8.4.4.1 MemoryManagerWrapper

```
friend class MemoryManagerWrapper [friend]
```

Definition at line 271 of file checkpoint_output_manager.hh.

8.4.5 Field Documentation

8.4.5.1 current_writer

```
SectionedOutputStream* jeod::CheckPointOutputManager::current_writer {} [private]
```

The writer that currently is active.

`trick_io(**)`

Definition at line 340 of file checkpoint_output_manager.hh.

Referenced by `create_trick_section_writer()`, `deregister_writer()`, `have_active_writer()`, and `register_writer()`.

8.4.5.2 filename

```
const std::string jeod::CheckPointOutputManager::filename [private]
```

The name of the checkpoint file.

Definition at line 345 of file checkpoint_output_manager.hh.

Referenced by `CheckPointOutputManager()`, and `create_section_writer()`.

8.4.5.3 is_open

```
bool jeod::CheckPointOutputManager::is_open {true} [private]
```

Is the checkpoint file open?

trick_io(**)

Definition at line 360 of file checkpoint_output_manager.hh.

Referenced by CheckPointOutputManager(), create_section_writer(), and operator!().

8.4.5.4 section_end

```
const std::string& jeod::CheckPointOutputManager::section_end [private]
```

The string that indicates the start of a checkpoint file section.

Definition at line 355 of file checkpoint_output_manager.hh.

Referenced by create_section_writer().

8.4.5.5 section_start

```
const std::string& jeod::CheckPointOutputManager::section_start [private]
```

The string that indicates the start of a checkpoint file section.

Definition at line 350 of file checkpoint_output_manager.hh.

Referenced by create_section_writer().

8.4.5.6 stream

```
std::ofstream jeod::CheckPointOutputManager::stream [private]
```

The C++ file stream that writes to the checkpoint file.

trick_io(**)

Definition at line 335 of file checkpoint_output_manager.hh.

Referenced by CheckPointOutputManager(), create_section_writer(), and operator!().

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

- [checkpoint_output_manager.hh](#)
- [checkpoint_output_manager.cc](#)

8.5 jeod::JeodTrickMemoryInterface::ContainerListEntry Struct Reference

Describes a Checkpointable object.

```
#include <trick_memory_interface.hh>
```

Public Member Functions

- [ContainerListEntry](#) (const void *parent, const JeodMemoryTypeDescriptor &tdesc, std::string sub_id, JeodCheckpointable &obj)
Construct an [ContainerListEntry](#) object.

Data Fields

- const void * [owner](#)
The object that contains the container.
- const JeodMemoryTypeDescriptor & [owner_type](#)
Type description of the object that contains the container.
- std::string [elem_name](#)
The name of the element of the container in the owning object.
- JeodCheckpointable & [container](#)
The container itself.

8.5.1 Detailed Description

Describes a Checkpointable object.

Definition at line 225 of file trick_memory_interface.hh.

8.5.2 Constructor & Destructor Documentation

8.5.2.1 ContainerListEntry()

```
jeod::JeodTrickMemoryInterface::ContainerListEntry::ContainerListEntry (
    const void * parent,
    const JeodMemoryTypeDescriptor & tdesc,
    std::string sub_id,
    JeodCheckpointable & obj ) [inline]
```

Construct an [ContainerListEntry](#) object.

Parameters

<i>parent</i>	Parent object
<i>tdesc</i>	Type descriptor
<i>sub_id</i>	Parent element
<i>obj</i>	Checkpointable itself

Definition at line 254 of file `trick_memory_interface.hh`.

8.5.3 Field Documentation

8.5.3.1 container

```
JeodCheckpointable& jeod::JeodTrickMemoryInterface::ContainerListEntry::container
```

The container itself.

`trick_units(-)`

Definition at line 245 of file `trick_memory_interface.hh`.

8.5.3.2 elem_name

```
std::string jeod::JeodTrickMemoryInterface::ContainerListEntry::elem_name
```

The name of the element of the container in the owning object.

`trick_units(-)`

Definition at line 240 of file `trick_memory_interface.hh`.

Referenced by `jeod::JeodTrick10MemoryInterface::deregister_container()`, and `jeod::JeodTrick10MemoryInterface::get_container_id()`.

8.5.3.3 owner

```
const void* jeod::JeodTrickMemoryInterface::ContainerListEntry::owner
```

The object that contains the container.

`trick_units(-)`

Definition at line 230 of file `trick_memory_interface.hh`.

Referenced by `jeod::JeodTrick10MemoryInterface::deregister_container()`, and `jeod::JeodTrick10MemoryInterface::get_container_id()`.

8.5.3.4 owner_type

```
const JeodMemoryTypeDescriptor& jeod::JeodTrickMemoryInterface::ContainerListEntry::owner_type
```

Type description of the object that contains the container.

trick_units(-)

Definition at line 235 of file trick_memory_interface.hh.

Referenced by `jeod::JeodTrick10MemoryInterface::deregister_container()`, and `jeod::JeodTrick10MemoryInterface::get_container_id()`.

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

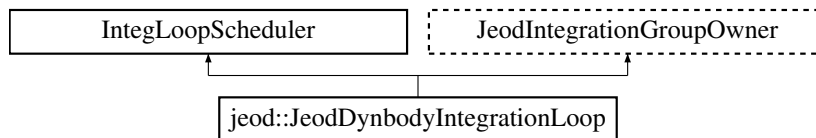
- [trick_memory_interface.hh](#)

8.6 jeod::JeodDynbodyIntegrationLoop Class Reference

A Trick::IntegLoopScheduler that provides the ability to integrate a collection of Trick::SimObject instances over time, with the sim objects capable of being moved from one integration loop to another during run time.

```
#include <trick_dynbody_integ_loop.hh>
```

Inheritance diagram for jeod::JeodDynbodyIntegrationLoop:



Public Member Functions

- [JeodDynbodyIntegrationLoop](#) ()
JeodDynbodyIntegrationLoop default constructor.
- [JeodDynbodyIntegrationLoop](#) (double cycle, Trick::SimObject &sim_object_in, TimeManager &time_manager_in, DynManager &dyn_manager_in, GravityManager &grav_manager_in, er7_utils::IntegratorConstructor *&integ_cotr_in, DynamicsIntegrationGroup &integ_group_factory)
JeodDynbodyIntegrationLoop non-default constructor.
- [~JeodDynbodyIntegrationLoop](#) () override
JeodDynbodyIntegrationLoop destructor.
- [JeodDynbodyIntegrationLoop](#) (const [JeodDynbodyIntegrationLoop](#) &)=delete
- [JeodDynbodyIntegrationLoop](#) & operator= (const [JeodDynbodyIntegrationLoop](#) &)=delete
- void [initialize_integ_loop](#) ()
S_define-level function to initialize the integration loop.
- void [set_time_to_loop_start](#) ()
S_define-level function to reset JEOD time to the time at the start of the current integration loop.
- void [update_integration_group](#) (JeodIntegrationGroup &group) override
Update the provided integration group, which must be the integration group contained within this integration loop object.

- int [add_sim_object](#) (Trick::SimObject &sim_obj) override
Add a sim object to the set of objects to be integrated by this integration loop object.
- virtual void [add_integrable_object](#) (er7_utils::IntegrableObject &integrable_object)
Add the specified integrable object, which should not be a DynBody, to the integration group's set of integrable objects.
- int [remove_sim_object](#) (Trick::SimObject &sim_obj) override
Remove a sim object from the set of objects to be integrated by this integration loop object.
- virtual void [remove_integrable_object](#) (er7_utils::IntegrableObject &integrable_object)
Remove the specified integrable object from the integration group's set of integrable objects.
- virtual void [gravitation](#) ()
Compute the gravitational accelerations of each dynamic body that is integrated by this integration loop.
- virtual void [collect_derivatives](#) ()
Collect the derivatives for each dynamic body that is integrated by this integration loop.
- virtual void [set_deriv_ephem_update](#) (bool val)
Set the deriv_ephem_update flag for the integration group.

Protected Member Functions

- Trick::SimObject * [find_containing_sim_object](#) (er7_utils::IntegrableObject &integrable_object)
Find the sim object that contains the specified integrable object.
- virtual void [add_sim_object_bodies](#) (Trick::SimObject &sim_obj)
Add the DynBody objects contained in the specified sim object to the set of DynBody objects integrated by this integration loop.
- virtual void [add_sim_object_bodies](#) ()
Add the dyn bodies contained in all the sim objects integrated by this integration loop to the loop's integration group.
- virtual void [remove_sim_object_bodies](#) (Trick::SimObject &sim_obj)
Remove the DynBody objects contained in the specified sim object from the set of DynBody objects integrated by this integration loop.
- int [integrate_dt](#) (double beg_sim_time, double del_sim_time) override
Integrate sim objects over the specified time span.

Protected Attributes

- Trick::SimObject * [loop_sim_object](#) {}
The simulation object that contains this integration loop object.
- DynManager * [dyn_manager](#) {}
The JEOD dynamics manager.
- TimeManager * [time_manager](#) {}
The JEOD time manager.
- GravityManager * [gravity_manager](#) {}
The gravity model manager.
- [JeodTrickIntegrator integ_interface](#)
Dummy integration interface; needed by the integ_group.
- er7_utils::IntegratorConstructor ** [integ_constructor](#) {}
Handle to the integration constructor used to create integrators.
- const DynamicsIntegrationGroup * [integ_group_factory](#) {}
The externally-supplied integration group used as a template for creating this integration loop's integration group.
- DynamicsIntegrationGroup * [integ_group](#) {}
The integration group that performs the integration.
- bool [deriv_ephem_update](#) {}
If set, ephemerides will be updated at the derivative rate.

Friends

- class [InputProcessor](#)
- void [init_attrjeod__JeodDynbodyIntegrationLoop](#) ()

8.6.1 Detailed Description

A `Trick::IntegLoopScheduler` that provides the ability to integrate a collection of `Trick::SimObject` instances over time, with the sim objects capable of being moved from one integration loop to another during run time.

A [JeodDynbodyIntegrationLoop](#) augments this capability in a number of regards:

- All `DynBody` objects contained in the sim objects integrated by a [JeodDynbodyIntegrationLoop](#) object are integrated using JEOD integration.
- The `DynBody` objects to be integrated by a [JeodDynbodyIntegrationLoop](#) object are automatically collected as a member of the `DynamicsIntegrationGroup` object contained within a [JeodDynbodyIntegrationLoop](#) object.
- Non-`DynBody` integrable objects can also be integrated using JEOD integration.
- Non-`DynBody` integrable objects that are elsewhere identified as being associated with a `DynBody` object are automatically collected along with the `DynBody` objects with which they are associated.
- The `DynBody` and associated integrable objects are integrated using the `DynamicsIntegrationGroup` object contained in the loop object.

Users of this class are strongly encouraged to do so via a `JeodIntegLoopSimObject`. See `$JEOD_HOME/lib/jeod/↔JEOD_S_modules/integ_loop.sm`.

Definition at line 134 of file `trick_dynbody_integ_loop.hh`.

8.6.2 Constructor & Destructor Documentation

8.6.2.1 [JeodDynbodyIntegrationLoop](#)() [1/3]

```
jeod::JeodDynbodyIntegrationLoop::JeodDynbodyIntegrationLoop ( )
```

[JeodDynbodyIntegrationLoop](#) default constructor.

Note

This exists only for the purpose of automated checkpoint/restart.

Warning

Do not use the default constructor outside of this context.

Definition at line 54 of file `trick_dynbody_integ_loop.cc`.

8.6.2.2 JeodDynbodyIntegrationLoop() [2/3]

```
jeod::JeodDynbodyIntegrationLoop::JeodDynbodyIntegrationLoop (
    double cycle,
    Trick::SimObject & sim_object_in,
    TimeManager & time_manager_in,
    DynManager & dyn_manager_in,
    GravityManager & grav_manager_in,
    er7_utils::IntegratorConstructor *& integ_cotr_in,
    DynamicsIntegrationGroup & integ_group_factory )
```

[JeodDynbodyIntegrationLoop](#) non-default constructor.

This is the constructor that should be used in the S_define file. The SimObject that contains this [JeodDynbodyIntegrationLoop](#) instance must register an "integ_loop" class job that calls the loop's integrate method.

Parameters

<i>cycle</i>	The integration interval in simulation seconds. This must be the same interval as specified in the integ_loop job specification.
<i>sim_object_in</i>	The SimObject that contains this JeodDynbodyIntegrationLoop instance.
<i>time_manager_in</i>	The simulation's time manager object.
<i>dyn_manager_in</i>	The simulation's dynamics manager object.
<i>grav_manager_in</i>	The simulation's gravity manager object.
<i>integ_cotr_in</i>	The integrator constructor used to create integration artifacts.
<i>integ_group_factory</i>	The integration group object used to create this loop's integ group.

Definition at line 60 of file trick_dynbody_integ_loop.cc.

References [add_sim_object\(\)](#), [jeod::SimInterfaceMessages::integration_error](#), and [loop_sim_object](#).

8.6.2.3 ~JeodDynbodyIntegrationLoop()

```
jeod::JeodDynbodyIntegrationLoop::~JeodDynbodyIntegrationLoop ( ) [override]
```

[JeodDynbodyIntegrationLoop](#) destructor.

Definition at line 91 of file trick_dynbody_integ_loop.cc.

References [integ_group](#).

8.6.2.4 JeodDynbodyIntegrationLoop() [3/3]

```
jeod::JeodDynbodyIntegrationLoop::~JeodDynbodyIntegrationLoop (
    const JeodDynbodyIntegrationLoop & ) [delete]
```

8.6.3 Member Function Documentation

8.6.3.1 add_integrable_object()

```
void jeod::JeodDynbodyIntegrationLoop::add_integrable_object (
    er7_utils::IntegrableObject & integrable_object ) [virtual]
```

Add the specified integrable object, which should not be a DynBody, to the integration group's set of integrable objects.

Parameters

<i>integrable_object</i>	Object to be added.
--------------------------	---------------------

Definition at line 159 of file `trick_dynbody_integ_loop.cc`.

References `add_sim_object()`, `find_containing_sim_object()`, and `integ_group`.

8.6.3.2 add_sim_object()

```
int jeod::JeodDynbodyIntegrationLoop::add_sim_object (
    Trick::SimObject & sim_obj ) [override]
```

Add a sim object to the set of objects to be integrated by this integration loop object.

The job queues for this loop are rebuilt after adding the sim object.

Parameters

<i>sim_obj</i>	The SimObject to be added to this loop object.
----------------	--

Returns

Zero => success, non-zero => error.

Definition at line 250 of file `trick_dynbody_integ_loop.cc`.

References `add_sim_object_bodies()`, and `dyn_manager`.

Referenced by `add_integrable_object()`, and `JeodDynbodyIntegrationLoop()`.

8.6.3.3 add_sim_object_bodies() [1/2]

```
void jeod::JeodDynbodyIntegrationLoop::add_sim_object_bodies (
    Trick::SimObject & sim_obj ) [protected], [virtual]
```

Add the DynBody objects contained in the specified sim object to the set of DynBody objects integrated by this integration loop.

Parameters

<i>sim_obj</i>	The SimObject being added to this loop object.
----------------	--

Definition at line 292 of file `trick_dynbody_integ_loop.cc`.

References `dyn_manager`, `find_containing_sim_object()`, and `integ_group`.

8.6.3.4 add_sim_object_bodies() [2/2]

```
void jeod::JeodDynbodyIntegrationLoop::add_sim_object_bodies ( ) [protected], [virtual]
```

Add the dyn bodies contained in all the sim objects integrated by this integration loop to the loop's integration group.

Definition at line 307 of file `trick_dynbody_integ_loop.cc`.

References `dyn_manager`, `find_containing_sim_object()`, and `integ_group`.

Referenced by `add_sim_object()`, and `update_integration_group()`.

8.6.3.5 collect_derivatives()

```
virtual void jeod::JeodDynbodyIntegrationLoop::collect_derivatives ( ) [inline], [virtual]
```

Collect the derivatives for each dynamic body that is integrated by this integration loop.

Definition at line 268 of file `trick_dynbody_integ_loop.hh`.

8.6.3.6 find_containing_sim_object()

```
Trick::SimObject * jeod::JeodDynbodyIntegrationLoop::find_containing_sim_object (
    er7_utils::IntegrableObject & integrable_object ) [protected]
```

Find the sim object that contains the specified integrable object.

Parameters

<i>integrable_object</i>	Object to be found.
--------------------------	---------------------

Returns

Sim object that contains the specified object, or null if none.

Definition at line 126 of file `trick_dynbody_integ_loop.cc`.

References `jeod::JeodSimulationInterface::get_address_at_name()`, and `jeod::JeodSimulationInterface::get_name_at_address()`.

Referenced by `add_integrable_object()`, `add_sim_object_bodies()`, and `remove_sim_object_bodies()`.

8.6.3.7 gravitation()

```
virtual void jeod::JeodDynbodyIntegrationLoop::gravitation ( ) [inline], [virtual]
```

Compute the gravitational accelerations of each dynamic body that is integrated by this integration loop.

Definition at line 259 of file `trick_dynbody_integ_loop.hh`.

8.6.3.8 initialize_integ_loop()

```
void jeod::JeodDynbodyIntegrationLoop::initialize_integ_loop ( )
```

S_define-level function to initialize the integration loop.

This function should be called as a very low phase integration class job.

Definition at line 97 of file `trick_dynbody_integ_loop.cc`.

References `deriv_ephem_update`, `dyn_manager`, `integ_constructor`, `integ_group`, `integ_group_factory`, `integ_interface`, `jeod::SimInterfaceMessages::integration_error`, and `time_manager`.

8.6.3.9 integrate_dt()

```
int jeod::JeodDynbodyIntegrationLoop::integrate_dt (
    double beg_sim_time,
    double del_sim_time ) [override], [protected]
```

Integrate sim objects over the specified time span.

This is an overridable internal integration function and is called by the externally-visible `integrate` method and by `call_dynamic_event_jobs`.

Returns

Zero/non-zero success indicator. Out-of-sync integrators cause a non-zero return.

Parameters

<i>beg_sim_time</i>	The time at the start of the integration interval.
<i>del_sim_time</i>	The time span of the integration interval.

Definition at line 187 of file `trick_dynbody_integ_loop.cc`.

References `integ_group`, `jeod::SimInterfaceMessages::integration_error`, and `trick_curr_integ`.

8.6.3.10 operator=()

```
JeodDynbodyIntegrationLoop& jeod::JeodDynbodyIntegrationLoop::operator= (
    const JeodDynbodyIntegrationLoop & ) [delete]
```

8.6.3.11 remove_integrable_object()

```
void jeod::JeodDynbodyIntegrationLoop::remove_integrable_object (
    er7_utils::IntegrableObject & integrable_object ) [virtual]
```

Remove the specified integrable object from the integration group's set of integrable objects.

Parameters

<i>integrable_object</i>	Object to be removed.
--------------------------	-----------------------

Definition at line 175 of file `trick_dynbody_integ_loop.cc`.

References `integ_group`.

8.6.3.12 remove_sim_object()

```
int jeod::JeodDynbodyIntegrationLoop::remove_sim_object (
    Trick::SimObject & sim_obj ) [override]
```

Remove a sim object from the set of objects to be integrated by this integration loop object.

The job queues for this loop are rebuilt after removing the sim object.

Parameters

<i>sim_obj</i>	The SimObject to be removed from this loop object.
----------------	--

Returns

Zero => success, non-zero => error.

Definition at line 271 of file `trick_dynbody_integ_loop.cc`.

References `dyn_manager`, and `remove_sim_object_bodies()`.

8.6.3.13 remove_sim_object_bodies()

```
void jeod::JeodDynbodyIntegrationLoop::remove_sim_object_bodies (
    Trick::SimObject & sim_obj ) [protected], [virtual]
```

Remove the DynBody objects contained in the specified sim object from the set of DynBody objects integrated by this integration loop.

Parameters

<i>sim_obj</i>	The SimObject being removed from this loop object.
----------------	--

Definition at line 323 of file `trick_dynbody_integ_loop.cc`.

References `dyn_manager`, `find_containing_sim_object()`, and `integ_group`.

Referenced by `remove_sim_object()`.

8.6.3.14 set_deriv_ephem_update()

```
virtual void jeod::JeodDynbodyIntegrationLoop::set_deriv_ephem_update (
    bool val ) [inline], [virtual]
```

Set the `deriv_ephem_update` flag for the integration group.

Parameters

<i>val</i>	New value for <code>deriv_ephem_update</code> .
------------	---

Definition at line 277 of file `trick_dynbody_integ_loop.hh`.

8.6.3.15 set_time_to_loop_start()

```
void jeod::JeodDynbodyIntegrationLoop::set_time_to_loop_start ( )
```

S_define-level function to reset JEOD time to the time at the start of the current integration loop.

This function should be called as a very low phase pre-integration class job in simulations that have multiple integration loops.

Definition at line 181 of file `trick_dynbody_integ_loop.cc`.

References `time_manager`.

8.6.3.16 `update_integration_group()`

```
void jeod::JeodDynbodyIntegrationLoop::update_integration_group (
    JeodIntegrationGroup & group ) [override]
```

Update the provided integration group, which must be the integration group contained within this integration loop object.

Note

This function is public because it is called (indirectly) from `DynManager::initialize_simulation`. It should otherwise be viewed as a protected or private function.

Parameters

<i>group</i>	The IntegrationGroup to be updated, which must be the integration loop's integration group object.
--------------	--

Definition at line 338 of file `trick_dynbody_integ_loop.cc`.

References `add_sim_object_bodies()`, `integ_group`, and `jeod::SimInterfaceMessages::integration_error`.

8.6.4 Friends And Related Function Documentation

8.6.4.1 `init_attrjeod__JeodDynbodyIntegrationLoop`

```
void init_attrjeod__JeodDynbodyIntegrationLoop ( ) [friend]
```

8.6.4.2 `InputProcessor`

```
friend class InputProcessor [friend]
```

Definition at line 137 of file `trick_dynbody_integ_loop.hh`.

8.6.5 Field Documentation

8.6.5.1 deriv_ephem_update

```
bool jeod::JeodDynbodyIntegrationLoop::deriv_ephem_update {} [protected]
```

If set, ephemerides will be updated at the derivative rate.

If clear, ephemerides will not be updated at the derivative rate by the ephemerides manager. Derivative-rate updates can still be attained by explicitly calling the various ephemerides model's update functions as derivative class jobs.↔
trick_units(-)

Definition at line 380 of file trick_dynbody_integ_loop.hh.

Referenced by initialize_integ_loop().

8.6.5.2 dyn_manager

```
DynManager* jeod::JeodDynbodyIntegrationLoop::dyn_manager {} [protected]
```

The JEOD dynamics manager.

trick_units(-)

Definition at line 340 of file trick_dynbody_integ_loop.hh.

Referenced by add_sim_object(), add_sim_object_bodies(), initialize_integ_loop(), remove_sim_object(), and remove_sim_object_bodies().

8.6.5.3 gravity_manager

```
GravityManager* jeod::JeodDynbodyIntegrationLoop::gravity_manager {} [protected]
```

The gravity model manager.

trick_units(-)

Definition at line 350 of file trick_dynbody_integ_loop.hh.

8.6.5.4 integ_constructor

```
er7_utils::IntegratorConstructor** jeod::JeodDynbodyIntegrationLoop::integ_constructor {}  
[protected]
```

Handle to the integration constructor used to create integrators.

trick_units(-)

Definition at line 360 of file trick_dynbody_integ_loop.hh.

Referenced by initialize_integ_loop().

8.6.5.5 integ_group

```
DynamicsIntegrationGroup* jeod::JeodDynbodyIntegrationLoop::integ_group {} [protected]
```

The integration group that performs the integration.

trick_units(-)

Definition at line 371 of file trick_dynbody_integ_loop.hh.

Referenced by add_integrable_object(), add_sim_object_bodies(), initialize_integ_loop(), integrate_dt(), remove_integrable_object(), remove_sim_object_bodies(), update_integration_group(), and ~JeodDynbodyIntegrationLoop().

8.6.5.6 integ_group_factory

```
const DynamicsIntegrationGroup* jeod::JeodDynbodyIntegrationLoop::integ_group_factory {} [protected]
```

The externally-supplied integration group used as a template for creating this integration loop's integration group.

trick_units(-)

Definition at line 366 of file trick_dynbody_integ_loop.hh.

Referenced by initialize_integ_loop().

8.6.5.7 integ_interface

```
JeodTrickIntegrator jeod::JeodDynbodyIntegrationLoop::integ_interface [protected]
```

Dummy integration interface; needed by the integ_group.

trick_units(-)

Definition at line 355 of file trick_dynbody_integ_loop.hh.

Referenced by initialize_integ_loop().

8.6.5.8 loop_sim_object

```
Trick::SimObject* jeod::JeodDynbodyIntegrationLoop::loop_sim_object {} [protected]
```

The simulation object that contains this integration loop object.

trick_units(–)

Definition at line 335 of file trick_dynbody_integ_loop.hh.

Referenced by JeodDynbodyIntegrationLoop().

8.6.5.9 time_manager

```
TimeManager* jeod::JeodDynbodyIntegrationLoop::time_manager {} [protected]
```

The JEOD time manager.

trick_units(–)

Definition at line 345 of file trick_dynbody_integ_loop.hh.

Referenced by initialize_integ_loop(), and set_time_to_loop_start().

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

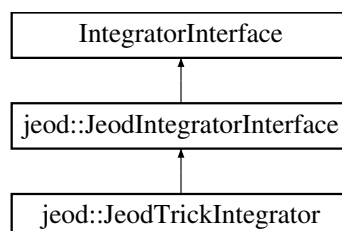
- [trick_dynbody_integ_loop.hh](#)
- [trick_dynbody_integ_loop.cc](#)

8.7 jeod::JeodIntegratorInterface Class Reference

A [JeodIntegratorInterface](#) extends the ER7 IntegratorInterface with the concept of a pointer to the simulation engine's integration object.

```
#include <jeod_integrator_interface.hh>
```

Inheritance diagram for jeod::JeodIntegratorInterface:



Public Member Functions

- [~JeodIntegratorInterface](#) () override=default
Destructor.
- virtual `er7_utils::Integration::Technique` [interpret_integration_type](#) (int) const =0
Interpret the integration technique.
- virtual `Trick::Integrator *` [get_integrator](#) ()=0
Get the simulation engine's integrator.

Friends

- class [InputProcessor](#)
- void [init_attrjeod__JeodIntegratorInterface](#) ()

8.7.1 Detailed Description

A [JeodIntegratorInterface](#) extends the ER7 IntegratorInterface with the concept of a pointer to the simulation engine's integration object.

Definition at line 84 of file `jeod_integrator_interface.hh`.

8.7.2 Constructor & Destructor Documentation

8.7.2.1 ~JeodIntegratorInterface()

```
jeod::JeodIntegratorInterface::~~JeodIntegratorInterface ( ) [override], [default]
```

Destructor.

8.7.3 Member Function Documentation

8.7.3.1 get_integrator()

```
virtual Trick::Integrator* jeod::JeodIntegratorInterface::get_integrator ( ) [pure virtual]
```

Get the simulation engine's integrator.

Returns

Pointer to the simulation engine's integrator.

Implemented in [jeod::JeodTrickIntegrator](#).

8.7.3.2 interpret_integration_type()

```
virtual er7_utils::Integration::Technique jeod::JeodIntegratorInterface::interpret_integration↵
_type (
    int ) const [pure virtual]
```

Interpret the integration technique.

Implemented in [jeod::JeodTrickIntegrator](#).

8.7.4 Friends And Related Function Documentation

8.7.4.1 init_attrjeod__JeodIntegratorInterface

```
void init_attrjeod__JeodIntegratorInterface ( ) [friend]
```

8.7.4.2 InputProcessor

```
friend class InputProcessor [friend]
```

Definition at line 86 of file jeod_integrator_interface.hh.

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

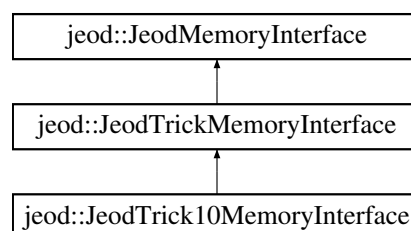
- [jeod_integrator_interface.hh](#)

8.8 jeod::JeodMemoryInterface Class Reference

Abstract interface between the JEOD memory manager and the simulation engine.

```
#include <memory_interface.hh>
```

Inheritance diagram for jeod::JeodMemoryInterface:



Public Member Functions

- [JeodMemoryInterface](#) ()=default
- virtual [~JeodMemoryInterface](#) ()=default
- [JeodMemoryInterface](#) (const [JeodMemoryInterface](#) &)=default
- [JeodMemoryInterface](#) & [operator=](#) (const [JeodMemoryInterface](#) &)=default
- virtual const struct ATTRIBUTES_tag * [find_attributes](#) (const std::string &type_name) const =0
Find the attributes for a given class name.
- virtual const struct ATTRIBUTES_tag * [find_attributes](#) (const std::type_info &data_type) const =0
Find the attributes for a given class.
- virtual struct ATTRIBUTES_tag [primitive_attributes](#) (const std::type_info &data_type) const =0
Create an attributes structure that represents a primitive type.
- virtual struct ATTRIBUTES_tag [pointer_attributes](#) (const struct ATTRIBUTES_tag &pointed_to_attr) const =0
Create an attributes structure that represents a pointer type.
- virtual struct ATTRIBUTES_tag [void_pointer_attributes](#) () const =0
Create a simulation engine description of void.*
- virtual struct ATTRIBUTES_tag [structure_attributes](#) (const struct ATTRIBUTES_tag *target_attr, std::size_t target_size) const =0
Create an attributes structure that represents a structured type.
- virtual bool [register_allocation](#) (const void *addr, const JeodMemoryItem &item, const JeodMemoryTypeDescriptor &tdesc, const char *file, unsigned int line)=0
Register allocated memory with the simulation engine.
- virtual void [deregister_allocation](#) (const void *addr, const JeodMemoryItem &item, const JeodMemoryTypeDescriptor &tdesc, const char *file, unsigned int line)=0
Revoke registration of memory that is about to be deleted.
- virtual void [register_container](#) (const void *container, const JeodMemoryTypeDescriptor &container_type, const std::string &elem_name, JeodCheckpointable &checkpointable)=0
Register a JeodCheckpointable object with the simulation engine.
- virtual void [deregister_container](#) (const void *container, const JeodMemoryTypeDescriptor &container_type, const std::string &elem_name, JeodCheckpointable &checkpointable)=0
Deregister a JeodCheckpointable object with the simulation engine.
- virtual bool [is_checkpoint_restart_supported](#) () const =0
Indicates whether the checkpoint/restart methods are viable.
- virtual const std::string [get_name_at_address](#) (const void *addr, const JeodMemoryTypeDescriptor *tdesc) const =0
Get the simulation engine's name (if any) of the address.
- virtual void * [get_address_at_name](#) (const std::string &name) const =0
Get the address (if any) identified by the given name.

Friends

- class [InputProcessor](#)
- void [init_attrjeod__JeodMemoryInterface](#) ()

8.8.1 Detailed Description

Abstract interface between the JEOD memory manager and the simulation engine.

Definition at line 83 of file `memory_interface.hh`.

8.8.2 Constructor & Destructor Documentation

8.8.2.1 JeodMemoryInterface() [1/2]

```
jeod::JeodMemoryInterface::JeodMemoryInterface ( ) [default]
```

8.8.2.2 ~JeodMemoryInterface()

```
virtual jeod::JeodMemoryInterface::~~JeodMemoryInterface ( ) [virtual], [default]
```

8.8.2.3 JeodMemoryInterface() [2/2]

```
jeod::JeodMemoryInterface::JeodMemoryInterface (
    const JeodMemoryInterface & ) [explicit], [default]
```

8.8.3 Member Function Documentation

8.8.3.1 deregister_allocation()

```
virtual void jeod::JeodMemoryInterface::deregister_allocation (
    const void * addr,
    const JeodMemoryItem & item,
    const JeodMemoryTypeDescriptor & tdesc,
    const char * file,
    unsigned int line ) [pure virtual]
```

Revoke registration of memory that is about to be deleted.

Parameters

in	<i>addr</i>	Address of allocated memory to be de-registered.
in	<i>item</i>	JEOD descriptor of the memory
in	<i>tdesc</i>	JEOD descriptor of the type of the allocated memory
in	<i>file</i>	File in which allocation was performed
in	<i>line</i>	Line number in that file

Implemented in [jeod::JeodTrickMemoryInterface](#).

8.8.3.2 deregister_container()

```
virtual void jeod::JeodMemoryInterface::deregister_container (
    const void * container,
    const JeodMemoryTypeDescriptor & container_type,
    const std::string & elem_name,
    JeodCheckpointable & checkpointable ) [pure virtual]
```

Deregister a JeodCheckpointable object with the simulation engine.

Parameters

in	<i>container</i>	Object that contains the checkpointable
in	<i>container_type</i>	Checkpointable container type info
in	<i>elem_name</i>	Element name of checkpointable object
in, out	<i>checkpointable</i>	The checkpointable object itself

Implemented in [jeod::JeodTrickMemoryInterface](#), and [jeod::JeodTrick10MemoryInterface](#).

8.8.3.3 find_attributes() [1/2]

```
virtual const struct ATTRIBUTES_tag* jeod::JeodMemoryInterface::find_attributes (
    const std::string & type_name ) const [pure virtual]
```

Find the attributes for a given class name.

Parameters

in	<i>type_name</i>	Name of the class.
----	------------------	--------------------

Returns

Attributes pointer. Note: This is not an allocated pointer.

Implemented in [jeod::JeodTrickMemoryInterface](#).

8.8.3.4 find_attributes() [2/2]

```
virtual const struct ATTRIBUTES_tag* jeod::JeodMemoryInterface::find_attributes (
    const std::type_info & data_type ) const [pure virtual]
```

Find the attributes for a given class.

Parameters

in	<i>data_type</i>	RTTI descriptor of the type.
----	------------------	------------------------------

Returns

Attributes pointer. Note: This is not an allocated pointer.

Implemented in [jeod::JeodTrickMemoryInterface](#).

8.8.3.5 get_address_at_name()

```
virtual void* jeod::JeodMemoryInterface::get_address_at_name (
    const std::string & name ) const [pure virtual]
```

Get the address (if any) identified by the given name.

Note

An implementation that does not support name translation will return the null pointer.
A stubbed implementation should have its `is_checkpoint_restart_supported` method return false.

Returns

Address corresponding to the given name, if any

Parameters

in	<i>name</i>	Value previously constructed by get_name_at_address()
----	-------------	---

Implemented in [jeod::JeodTrickMemoryInterface](#), and [jeod::JeodTrick10MemoryInterface](#).

Referenced by `jeod::JeodSimulationInterface::get_address_at_name()`.

8.8.3.6 get_name_at_address()

```
virtual const std::string jeod::JeodMemoryInterface::get_name_at_address (
    const void * addr,
    const JeodMemoryTypeDescriptor * tdesc ) const [pure virtual]
```

Get the simulation engine's name (if any) of the address.

A derived class associated with a simulation engine that does not support this translation should return an empty string for all calls. When the underlying simulation engine does support this translation, the implementation should return values as follows:

- The string "NULL" if the input address is the null pointer.
- The empty string to indicate an invalid input address or an input address that is unknown to the simulation engine.
- A non-empty, non-"NULL" string to indicate a valid address. Applying the `get_address_at_name` method to this result must yield the input address.

Note

A stubbed implementation should have its `is_checkpoint_restart_supported` method return false.

Returns

Name of the address, if any

Parameters

in	<i>addr</i>	Address of memory to identified by name
in	<i>tdesc</i>	Type context in which to interpret the address

Implemented in [jeod::JeodTrickMemoryInterface](#), and [jeod::JeodTrick10MemoryInterface](#).

Referenced by `jeod::JeodSimulationInterface::get_name_at_address()`.

8.8.3.7 is_checkpoint_restart_supported()

```
virtual bool jeod::JeodMemoryInterface::is_checkpoint_restart_supported ( ) const [pure virtual]
```

Indicates whether the checkpoint/restart methods are viable.

Checkpoint/restart can be used only in an environment that provides viable checkpoint/restart methods.

Returns

True if the checkpoint / restart is supported, false otherwise.

Implemented in [jeod::JeodTrickMemoryInterface](#), and [jeod::JeodTrick10MemoryInterface](#).

8.8.3.8 operator=()

```
JeodMemoryInterface& jeod::JeodMemoryInterface::operator= (
    const JeodMemoryInterface & ) [default]
```

8.8.3.9 pointer_attributes()

```
virtual struct ATTRIBUTES_tag jeod::JeodMemoryInterface::pointer_attributes (
    const struct ATTRIBUTES_tag & pointed_to_attr ) const [pure virtual]
```

Create an attributes structure that represents a pointer type.

Parameters

in	<i>pointed_to_attr</i>	Attributes of the pointed-to type.
----	------------------------	------------------------------------

Returns

Attribute structure.

Implemented in [jeod::JeodTrickMemoryInterface](#).

8.8.3.10 primitive_attributes()

```
virtual struct ATTRIBUTES_tag jeod::JeodMemoryInterface::primitive_attributes (
    const std::type_info & data_type ) const [pure virtual]
```

Create an attributes structure that represents a primitive type.

Parameters

in	<i>data_type</i>	RTTI descriptor of the type.
----	------------------	------------------------------

Returns

Attributes structure.

Implemented in [jeod::JeodTrickMemoryInterface](#).

8.8.3.11 register_allocation()

```
virtual bool jeod::JeodMemoryInterface::register_allocation (
    const void * addr,
    const JeodMemoryItem & item,
    const JeodMemoryTypeDescriptor & tdesc,
    const char * file,
    unsigned int line ) [pure virtual]
```

Register allocated memory with the simulation engine.

Parameters

in	<i>addr</i>	Address of allocated memory to be registered.
in	<i>item</i>	JEOD descriptor of the allocated memory
in	<i>tdesc</i>	JEOD descriptor of the type of the allocated memory
in	<i>file</i>	File in which allocation was performed
in	<i>line</i>	Line number in that file

Implemented in [jeod::JeodTrickMemoryInterface](#).

8.8.3.12 register_container()

```
virtual void jeod::JeodMemoryInterface::register_container (
    const void * container,
    const JeodMemoryTypeDescriptor & container_type,
    const std::string & elem_name,
    JeodCheckpointable & checkpointable ) [pure virtual]
```

Register a JeodCheckpointable object with the simulation engine.

Parameters

in	<i>container</i>	Object that contains the checkpointable
in	<i>container_type</i>	Checkpointable container type info
in	<i>elem_name</i>	Element name of checkpointable object
in, out	<i>checkpointable</i>	The checkpointable object itself

Implemented in [jeod::JeodTrickMemoryInterface](#), and [jeod::JeodTrick10MemoryInterface](#).

8.8.3.13 structure_attributes()

```
virtual struct ATTRIBUTES_tag jeod::JeodMemoryInterface::structure_attributes (
    const struct ATTRIBUTES_tag * target_attr,
    std::size_t target_size ) const [pure virtual]
```

Create an attributes structure that represents a structured type.

Parameters

in	<i>target_attr</i>	Attributes from find_attributes
in	<i>target_size</i>	Size of the underlying type

Returns

Attribute structure.

Implemented in [jeod::JeodTrickMemoryInterface](#).

8.8.3.14 void_pointer_attributes()

```
virtual struct ATTRIBUTES_tag jeod::JeodMemoryInterface::void_pointer_attributes ( ) const
[pure virtual]
```

Create a simulation engine description of void*.

Returns

Attribute structure.

Implemented in [jeod::JeodTrickMemoryInterface](#).

8.8.4 Friends And Related Function Documentation**8.8.4.1 init_attrjeod__JeodMemoryInterface**

```
void init_attrjeod__JeodMemoryInterface ( ) [friend]
```

8.8.4.2 InputProcessor

```
friend class InputProcessor [friend]
```

Definition at line 85 of file `memory_interface.hh`.

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

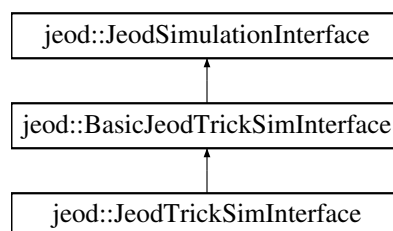
- [memory_interface.hh](#)

8.9 jeod::JeodSimulationInterface Class Reference

This abstract class defines the basis for the interface between JEOD and a simulation engine.

```
#include <simulation_interface.hh>
```

Inheritance diagram for `jeod::JeodSimulationInterface`:

**Public Types**

- enum [Mode](#) {
[Construction](#) = 0, [PreCheckpoint](#) = 1, [Checkpoint](#) = 2, [PostCheckpoint](#) = 3,
[Restart](#) = 4, [Restore](#) = 5, [Initialization](#) = 6, [Operational](#) = 7,
[Shutdown](#) = 8, [Dead](#) = 9, [NumModes](#) = 10 }

Defines the states of the [JeodSimulationInterface](#) state machine.

Public Member Functions

- [JeodSimulationInterface](#) ()
Construct a [JeodSimulationInterface](#) object.
- virtual [~JeodSimulationInterface](#) ()
Destruct a [JeodSimulationInterface](#) object.
- [JeodSimulationInterface](#) (const [JeodSimulationInterface](#) &)=delete
- [JeodSimulationInterface](#) & operator= (const [JeodSimulationInterface](#) &)=delete
- virtual void [configure](#) (const [JeodSimulationInterfaceInit](#) &config)
Configure a [JeodSimulationInterface](#) object.
- [Mode](#) [get_mode](#) () const
Get the current mode.
- virtual void [set_mode](#) ([Mode](#) new_mode)
Set the mode, but only if allowed per the mode state transition diagram.

Static Public Member Functions

- static [JeodIntegratorInterface](#) * [create_integrator_interface](#) ()
Create a simulation integrator interface object.
- static double [get_job_cycle](#) ()
Get the cycle time of the currently executing job.
- static std::string [get_name_at_address](#) (const void *addr, const [JeodMemoryTypeDescriptor](#) *tdesc)
Translate the given address to a symbolic name.
- static void * [get_address_at_name](#) (const std::string &name)
Translate the given symbolic name to an address.
- static [JeodMemoryInterface](#) & [get_memory_interface](#) ()
Get the interface with the simulation memory model.
- static [SectionedInputStream](#) [get_checkpoint_reader](#) (const std::string §ion_id)
Get a reader of a section of the currently open checkpoint file.
- static [SectionedOutputStream](#) [get_checkpoint_writer](#) (const std::string §ion_id)
Get a writer to a section of the currently open checkpoint file.

Protected Member Functions

- virtual [JeodIntegratorInterface](#) * [create_integrator_internal](#) ()=0
Create an integration interface object.
- virtual double [get_job_cycle_internal](#) ()=0
Get the simulation cycle time of the currently executing function.
- virtual [JeodMemoryInterface](#) & [get_memory_interface_internal](#) ()=0
Get the interface with the simulation memory manager.
- virtual [SectionedInputStream](#) [get_checkpoint_reader_internal](#) (const std::string §ion_id)=0
Get a checkpoint section reader.
- virtual [SectionedOutputStream](#) [get_checkpoint_writer_internal](#) (const std::string §ion_id)=0
Get a checkpoint section writer.

Protected Attributes

- [Mode](#) [mode](#) {Construction}
The mode in which the simulation interface is operating.
- [Mode](#) [saved_mode](#) {Construction}
The mode prior to a checkpoint or restart process.

Static Protected Attributes

- static [JeodSimulationInterface](#) * [sim_interface](#) = nullptr

The singleton instance of a SimulationInterface object that must be created by a conforming JEOD simulation before any call can be made to one of the three static methods declared above.

Friends

- class [InputProcessor](#)
- void [init_attrjeod__JeodSimulationInterface](#) ()

8.9.1 Detailed Description

This abstract class defines the basis for the interface between JEOD and a simulation engine.

A compliant derived class must contain one instance each of a class that derives from MessageHandler and a class that derives from JeodMemoryManager. The MessageHandler object must be constructed before the Jeod↔MemoryManager object; destruction must be performed in reverse order.

Definition at line 129 of file simulation_interface.hh.

8.9.2 Member Enumeration Documentation

8.9.2.1 Mode

```
enum jeod::JeodSimulationInterface::Mode
```

Defines the states of the [JeodSimulationInterface](#) state machine.

Enumerator

Construction	
PreCheckpoint	
Checkpoint	
PostCheckpoint	
Restart	
Restore	
Initialization	
Operational	
Shutdown	
Dead	
NumModes	

Definition at line 137 of file simulation_interface.hh.

8.9.3 Constructor & Destructor Documentation

8.9.3.1 JeodSimulationInterface() [1/2]

```
jeod::JeodSimulationInterface::JeodSimulationInterface ( )
```

Construct a [JeodSimulationInterface](#) object.

Definition at line 65 of file `simulation_interface.cc`.

References `sim_interface`, and `jeod::SimInterfaceMessages::singleton_error`.

8.9.3.2 ~JeodSimulationInterface()

```
jeod::JeodSimulationInterface::~~JeodSimulationInterface ( ) [virtual]
```

Destruct a [JeodSimulationInterface](#) object.

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

References `sim_interface`.

8.9.3.3 JeodSimulationInterface() [2/2]

```
jeod::JeodSimulationInterface::JeodSimulationInterface (
    const JeodSimulationInterface & ) [delete]
```

8.9.4 Member Function Documentation

8.9.4.1 configure()

```
void jeod::JeodSimulationInterface::configure (
    const JeodSimulationInterfaceInit & config ) [virtual]
```

Configure a [JeodSimulationInterface](#) object.

Parameters

<code>in</code>	<code><i>config</i></code>	Configuration spec
-----------------	----------------------------	--------------------

Definition at line 95 of file simulation_interface.cc.

References [jeod::JeodSimulationInterfaceInit::memory_debug_level](#), [jeod::JeodSimulationInterfaceInit::message_↵_suppress_id](#), [jeod::JeodSimulationInterfaceInit::message_suppress_location](#), and [jeod::JeodSimulation↵InterfaceInit::message_suppression_level](#).

8.9.4.2 create_integrator_interface()

```
JeodIntegratorInterface * jeod::JeodSimulationInterface::create_integrator_interface ( ) [static]
```

Create a simulation integrator interface object.

Returns

Constructed IntegratorInterface object.

Definition at line 110 of file simulation_interface.cc.

References [create_integrator_internal\(\)](#), [sim_interface](#), and [jeod::SimInterfaceMessages::singleton_error](#).

8.9.4.3 create_integrator_internal()

```
virtual JeodIntegratorInterface* jeod::JeodSimulationInterface::create_integrator_internal ( )  
[protected], [pure virtual]
```

Create an integration interface object.

The calling object is responsible for destroying the created object.

Returns

Created integration interface object.

Implemented in [jeod::BasicJeodTrickSimInterface](#).

Referenced by [create_integrator_interface\(\)](#).

8.9.4.4 get_address_at_name()

```
void * jeod::JeodSimulationInterface::get_address_at_name (   
const std::string & name ) [static]
```

Translate the given symbolic name to an address.

Returns

Address

Parameters

in	<i>name</i>	Symbolic name
----	-------------	---------------

Definition at line 201 of file `simulation_interface.cc`.

References `jeod::JeodMemoryInterface::get_address_at_name()`, `get_memory_interface_internal()`, `sim_interface`, and `jeod::SimInterfaceMessages::singleton_error`.

Referenced by `jeod::JeodDynbodyIntegrationLoop::find_containing_sim_object()`.

8.9.4.5 `get_checkpoint_reader()`

```
SectionedInputStream jeod::JeodSimulationInterface::get_checkpoint_reader (
    const std::string & section_id ) [static]
```

Get a reader of a section of the currently open checkpoint file.

Returns

Checkpoint reader

Parameters

in	<i>section↔ _id</i>	Section ID
----	-------------------------	------------

Definition at line 225 of file `simulation_interface.cc`.

References `get_checkpoint_reader_internal()`, `sim_interface`, and `jeod::SimInterfaceMessages::singleton_error`.

Referenced by `jeod::JeodTrick10MemoryInterface::restore_allocations()`, and `jeod::JeodTrick10MemoryInterface↔::restore_containers()`.

8.9.4.6 `get_checkpoint_reader_internal()`

```
virtual SectionedInputStream jeod::JeodSimulationInterface::get_checkpoint_reader_internal (
    const std::string & section_id ) [protected], [pure virtual]
```

Get a checkpoint section reader.

Implemented in `jeod::BasicJeodTrickSimInterface`.

Referenced by `get_checkpoint_reader()`.

8.9.4.7 get_checkpoint_writer()

```
SectionedOutputStream jeod::JeodSimulationInterface::get_checkpoint_writer (
    const std::string & section_id ) [static]
```

Get a writer to a section of the currently open checkpoint file.

Returns

Checkpoint writer

Parameters

in	<i>section_id</i>	Section ID
----	-------------------	------------

Definition at line 245 of file simulation_interface.cc.

References `get_checkpoint_writer_internal()`, `sim_interface`, and `jeod::SimInterfaceMessages::singleton_error`.

Referenced by `jeod::JeodTrick10MemoryInterface::checkpoint_allocations()`, and `jeod::JeodTrick10MemoryInterface::checkpoint_containers()`.

8.9.4.8 get_checkpoint_writer_internal()

```
virtual SectionedOutputStream jeod::JeodSimulationInterface::get_checkpoint_writer_internal (
    const std::string & section_id ) [protected], [pure virtual]
```

Get a checkpoint section writer.

Implemented in `jeod::BasicJeodTrickSimInterface`.

Referenced by `get_checkpoint_writer()`.

8.9.4.9 get_job_cycle()

```
double jeod::JeodSimulationInterface::get_job_cycle ( ) [static]
```

Get the cycle time of the currently executing job.

Returns

Cycle time in simulation engine seconds of the currently executing job.
Units: s

Definition at line 133 of file simulation_interface.cc.

References `get_job_cycle_internal()`, `sim_interface`, and `jeod::SimInterfaceMessages::singleton_error`.

8.9.4.10 `get_job_cycle_internal()`

```
virtual double jeod::JeodSimulationInterface::get_job_cycle_internal ( ) [protected], [pure virtual]
```

Get the simulation cycle time of the currently executing function.

Returns

Cycle time in simulation engine seconds

Implemented in [jeod::BasicJeodTrickSimInterface](#).

Referenced by `get_job_cycle()`.

8.9.4.11 `get_memory_interface()`

```
JeodMemoryInterface & jeod::JeodSimulationInterface::get_memory_interface ( ) [static]
```

Get the interface with the simulation memory model.

Returns

Memory interface

Definition at line 156 of file `simulation_interface.cc`.

References `get_memory_interface_internal()`, `sim_interface`, and `jeod::SimInterfaceMessages::singleton_error`.

8.9.4.12 `get_memory_interface_internal()`

```
virtual JeodMemoryInterface& jeod::JeodSimulationInterface::get_memory_interface_internal ( ) [protected], [pure virtual]
```

Get the interface with the simulation memory manager.

Returns

JEOD/simulation engine memory interface.

Implemented in [jeod::BasicJeodTrickSimInterface](#).

Referenced by `get_address_at_name()`, `get_memory_interface()`, and `get_name_at_address()`.

8.9.4.13 get_mode()

```
Mode jeod::JeodSimulationInterface::get_mode ( ) const [inline]
```

Get the current mode.

Definition at line 235 of file simulation_interface.hh.

Referenced by jeod::BasicJeodTrickSimInterface::set_mode().

8.9.4.14 get_name_at_address()

```
std::string jeod::JeodSimulationInterface::get_name_at_address (
    const void * addr,
    const JeodMemoryTypeDescriptor * tdesc ) [static]
```

Translate the given address to a symbolic name.

Returns

Symbolic name

Parameters

in	<i>addr</i>	Address
in	<i>tdesc</i>	Descriptor

Definition at line 177 of file simulation_interface.cc.

References get_memory_interface_internal(), jeod::JeodMemoryInterface::get_name_at_address(), sim_interface, and jeod::SimInterfaceMessages::singleton_error.

Referenced by jeod::JeodDynbodyIntegrationLoop::find_containing_sim_object().

8.9.4.15 operator=()

```
JeodSimulationInterface& jeod::JeodSimulationInterface::operator= (
    const JeodSimulationInterface & ) [delete]
```

8.9.4.16 set_mode()

```
void jeod::JeodSimulationInterface::set_mode (
    Mode new_mode ) [virtual]
```

Set the mode, but only if allowed per the mode state transition diagram.

Assumptions and Limitations

- The standard JEODSys [Trick](#) sim object follows the correct state transition diagram. A similar sequence must be implemented when JEOD is used outside of the [Trick](#) environment. In a [Trick](#) environment, *nobody* should call this function except the [Trick](#) scheduler, and these calls must conform with the sequence in the standard JEODSys [Trick](#) sim object.

Parameters

in	<i>new_mode</i>	New mode
----	-----------------	----------

Reimplemented in [jeod::BasicJeodTrickSimInterface](#).

Definition at line 271 of file simulation_interface.cc.

References Checkpoint, Construction, Dead, jeod::SimInterfaceMessages::implementation_error, Initialization, mode, NumModes, Operational, jeod::SimInterfaceMessages::phasing_error, PostCheckpoint, PreCheckpoint, Restart, Restore, saved_mode, and Shutdown.

Referenced by jeod::BasicJeodTrickSimInterface::set_mode().

8.9.5 Friends And Related Function Documentation

8.9.5.1 init_attrjeod__JeodSimulationInterface

```
void init_attrjeod__JeodSimulationInterface ( ) [friend]
```

8.9.5.2 InputProcessor

```
friend class InputProcessor [friend]
```

Definition at line 131 of file simulation_interface.hh.

8.9.6 Field Documentation

8.9.6.1 mode

```
Mode jeod::JeodSimulationInterface::mode {Construction} [protected]
```

The mode in which the simulation interface is operating.

trick_units(—)

Definition at line 294 of file simulation_interface.hh.

Referenced by set_mode().

8.9.6.2 saved_mode

```
Mode jeod::JeodSimulationInterface::saved_mode {Construction} [protected]
```

The mode prior to a checkpoint or restart process.

set_mode(Restore) restores the mode to this saved value.trick_units(—)

Definition at line 300 of file simulation_interface.hh.

Referenced by set_mode().

8.9.6.3 sim_interface

```
JeodSimulationInterface * jeod::JeodSimulationInterface::sim_interface = nullptr [static],  
[protected]
```

The singleton instance of a SimulationInterface object that must be created by a conforming JEOD simulation before any call can be made to one of the three static methods declared above.

The first created instance of a class that derives from this base class becomes **the** SimulationInterface object used during the course of the simulation. Creation of more than one SimulationInterface objects is a non-fatal error. Attempts to allocate memory or generate a message prior creating a SimulationInterface object is a fatal error.trick_↵
_io(*o) trick_units(—)

Definition at line 256 of file simulation_interface.hh.

Referenced by create_integrator_interface(), get_address_at_name(), get_checkpoint_reader(), get_checkpoint_↵
_writer(), get_job_cycle(), get_memory_interface(), get_name_at_address(), JeodSimulationInterface(), and ~↵
JeodSimulationInterface().

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

- [simulation_interface.hh](#)
- [simulation_interface.cc](#)

8.10 jeod::JeodSimulationInterfaceInit Class Reference

Define configuration data needed to configure the dynamically-created message handler and memory manager.

```
#include <simulation_interface.hh>
```

Public Member Functions

- [JeodSimulationInterfaceInit](#) ()
Construct a [JeodSimulationInterfaceInit](#) object.

Data Fields

- unsigned int [message_suppression_level](#)
Specifies the message handler's message suppression level; see [MessageHandler::suppression_level](#) for details.
- bool [message_suppress_id](#) {}
Specifies the message handler's suppress_id flag; see [MessageHandler::suppression_id](#) for details.
- bool [message_suppress_location](#) {}
Specifies the message handler's suppress_location flag; see [MessageHandler::suppression_location](#) for details.
- unsigned int [memory_debug_level](#) {}
Specifies the memory manager's debug level; see [JeodMemoryManager::debug_level](#) for details.

8.10.1 Detailed Description

Define configuration data needed to configure the dynamically-created message handler and memory manager.

Definition at line 85 of file [simulation_interface.hh](#).

8.10.2 Constructor & Destructor Documentation

8.10.2.1 JeodSimulationInterfaceInit()

```
jeod::JeodSimulationInterfaceInit::JeodSimulationInterfaceInit ( )
```

Construct a [JeodSimulationInterfaceInit](#) object.

Definition at line 49 of file [simulation_interface.cc](#).

References [memory_debug_level](#), and [message_suppression_level](#).

8.10.3 Field Documentation

8.10.3.1 memory_debug_level

```
unsigned int jeod::JeodSimulationInterfaceInit::memory_debug_level {}
```

Specifies the memory manager's debug level; see JeodMemoryManager::debug_level for details.

trick_units(-)

Definition at line 119 of file simulation_interface.hh.

Referenced by jeod::JeodSimulationInterface::configure(), and JeodSimulationInterfaceInit().

8.10.3.2 message_suppress_id

```
bool jeod::JeodSimulationInterfaceInit::message_suppress_id {}
```

Specifies the message handler's suppress_id flag; see MessageHandler::suppression_id for details.

trick_units(-)

Definition at line 107 of file simulation_interface.hh.

Referenced by jeod::JeodSimulationInterface::configure().

8.10.3.3 message_suppress_location

```
bool jeod::JeodSimulationInterfaceInit::message_suppress_location {}
```

Specifies the message handler's suppress_location flag; see MessageHandler::suppression_location for details.

trick_units(-)

Definition at line 113 of file simulation_interface.hh.

Referenced by jeod::JeodSimulationInterface::configure().

8.10.3.4 message_suppression_level

```
unsigned int jeod::JeodSimulationInterfaceInit::message_suppression_level
```

Specifies the message handler's message suppression level; see MessageHandler::suppression_level for details.

trick_units(-)

Definition at line 101 of file simulation_interface.hh.

Referenced by jeod::JeodSimulationInterface::configure(), and JeodSimulationInterfaceInit().

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

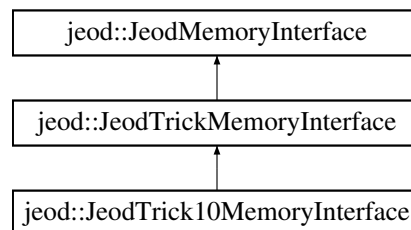
- [simulation_interface.hh](#)
- [simulation_interface.cc](#)

8.11 jeod::JeodTrick10MemoryInterface Class Reference

A TrickMemoryInterface implements the two required methods needed to register and deregister memory with the simulation engine, [Trick](#) in this case.

```
#include <trick10_memory_interface.hh>
```

Inheritance diagram for jeod::JeodTrick10MemoryInterface:



Public Member Functions

- [JeodTrick10MemoryInterface](#) ()
Construct a [JeodTrick10MemoryInterface](#) object.
- [~JeodTrick10MemoryInterface](#) () override=default
- [JeodTrick10MemoryInterface](#) (const [JeodTrick10MemoryInterface](#) &)=delete
- [JeodTrick10MemoryInterface](#) & operator= (const [JeodTrick10MemoryInterface](#) &)=delete
- void [register_container](#) (const void *owner, const JeodMemoryTypeDescriptor &owner_type, const std::string &elem_name, JeodCheckpointable &container) override
Register the checkpointable object with [Trick](#).
- void [deregister_container](#) (const void *owner, const JeodMemoryTypeDescriptor &owner_type, const std::string &elem_name, JeodCheckpointable &container) override
Revoke the registrations performed by register_container.
- const std::string [get_name_at_address](#) (const void *addr, const JeodMemoryTypeDescriptor *tdesc) const override
Get the simulation name, if any, associated with the address.
- void * [get_address_at_name](#) (const std::string &name) const override
Get the address, if any, that corresponds to the given name.
- bool [is_checkpoint_restart_supported](#) () const override
The Trick10 memory interface supports checkpoint/restart.
- const std::string [get_trick_checkpoint_file](#) (bool checkpoint) override
Get the name of the current [Trick](#) checkpoint file.
- void [checkpoint_containers](#) () override
Dump the checkpointable objects to the checkpoint file.
- void [restore_containers](#) () override
Restore the checkpointable objects from the checkpoint file.
- void [checkpoint_allocations](#) () override
Dump the allocation information to the checkpoint file.
- void [restore_allocations](#) (JeodMemoryManager &memory_manager) override
Restore the allocated data per the checkpoint file.

Protected Member Functions

- `std::string get_container_id (const ContainerListEntry &entry) const`
Construct the identifier for a checkpointable object.
- `std::string translate_addr_to_name (const void *addr, const ATTRIBUTES *attr) const`
Translate the given address to an address specification string, with the address interpreted in the context of the supplied attributes.
- `void * translate_name_to_addr (const std::string &spec) const`
Translate the given address specification string to an address.

Protected Attributes

- `Trick::ClassicCheckPointAgent * trick_checkpoint_agent`
Trick checkpoint agent.

Friends

- class `InputProcessor`
- void `init_attrjeod__JeodTrick10MemoryInterface ()`

Additional Inherited Members

8.11.1 Detailed Description

A TrickMemoryInterface implements the two required methods needed to register and deregister memory with the simulation engine, [Trick](#) in this case.

Definition at line 111 of file `trick10_memory_interface.hh`.

8.11.2 Constructor & Destructor Documentation

8.11.2.1 JeodTrick10MemoryInterface() [1/2]

```
jeod::JeodTrick10MemoryInterface::JeodTrick10MemoryInterface ( )
```

Construct a [JeodTrick10MemoryInterface](#) object.

Definition at line 60 of file `trick10_memory_interface.cc`.

References `jeod::SimInterfaceMessages::interface_error`, `trick_checkpoint_agent`, and `trick_MM`.

8.11.2.2 ~JeodTrick10MemoryInterface()

```
jeod::JeodTrick10MemoryInterface::~~JeodTrick10MemoryInterface ( ) [override], [default]
```

8.11.2.3 JeodTrick10MemoryInterface() [2/2]

```
jeod::JeodTrick10MemoryInterface::JeodTrick10MemoryInterface (
    const JeodTrick10MemoryInterface & ) [delete]
```

8.11.3 Member Function Documentation

8.11.3.1 checkpoint_allocations()

```
void jeod::JeodTrick10MemoryInterface::checkpoint_allocations ( ) [override], [virtual]
```

Dump the allocation information to the checkpoint file.

Reimplemented from [jeod::JeodTrickMemoryInterface](#).

Definition at line 422 of file `trick_memory_interface_chkpnt.cc`.

References [jeod::SectionedOutputStream::activate\(\)](#), [jeod::JeodTrickMemoryInterface::allocation_map](#), [jeod::JeodTrickMemoryInterface::construct_identifier\(\)](#), [jeod::JeodTrickMemoryInterface::container_list](#), [jeod::JeodSimulationInterface::get_checkpoint_writer\(\)](#), [jeod::SimInterfaceMessages::interface_error](#), [jeod::JeodTrickMemoryInterface::AllocationMapEntry::is_array](#), [jeod::JeodTrickMemoryInterface::AllocationMapEntry::nelements](#), and [jeod::JeodTrickMemoryInterface::AllocationMapEntry::typeid_info](#).

Referenced by [jeod::BasicJeodTrickSimInterface::checkpoint_allocations\(\)](#).

8.11.3.2 checkpoint_containers()

```
void jeod::JeodTrick10MemoryInterface::checkpoint_containers ( ) [override], [virtual]
```

Dump the checkpointable objects to the checkpoint file.

Reimplemented from [jeod::JeodTrickMemoryInterface](#).

Definition at line 236 of file `trick_memory_interface_chkpnt.cc`.

References [jeod::SectionedOutputStream::activate\(\)](#), [jeod::JeodTrickMemoryInterface::container_list](#), [jeod::SectionedOutputStream::deactivate\(\)](#), [jeod::JeodSimulationInterface::get_checkpoint_writer\(\)](#), [get_container_id\(\)](#), and [jeod::SimInterfaceMessages::interface_error](#).

Referenced by [jeod::BasicJeodTrickSimInterface::checkpoint_containers\(\)](#).

8.11.3.3 deregister_container()

```
void jeod::JeodTrick10MemoryInterface::deregister_container (
    const void * owner,
    const JeodMemoryTypeDescriptor & owner_type,
    const std::string & elem_name,
    JeodCheckpointable & container ) [override], [virtual]
```

Revoke the registrations performed by register_container.

This function is typically called at destruction time via JEOD_DEREGISTER_CHECKPOINTABLE.

Assumptions and Limitations

- The following unenforced assumptions are made:
 - A corresponding register_container was previously made.
 - [Trick](#) has been pre-initialized.

Enforcement of the above is the responsibility the simulation developer, the JEOD memory manager, and the simulation interface.

Parameters

in	<i>owner</i>	Owner of the container
in	<i>owner_type</i>	Owner type descriptor
in	<i>elem_name</i>	Container element
in, out	<i>container</i>	The container

Implements [jeod::JeodMemoryInterface](#).

Definition at line 140 of file trick_memory_interface_chkpnt.cc.

References [jeod::JeodTrickMemoryInterface::container_list](#), [jeod::JeodTrickMemoryInterface::ContainerListEntry](#), [jeod::SimInterfaceMessages::interface_error](#), [jeod::JeodTrickMemoryInterface::ContainerListEntry](#), [jeod::JeodTrickMemoryInterface::owner](#), and [jeod::JeodTrickMemoryInterface::ContainerListEntry::owner_type](#).

8.11.3.4 get_address_at_name()

```
void * jeod::JeodTrick10MemoryInterface::get_address_at_name (
    const std::string & name ) const [override], [virtual]
```

Get the address, if any, that corresponds to the given name.

Returns

Name of the address, if any

Parameters

in	<i>name</i>	of an address Units: Name
----	-------------	------------------------------

Implements [jeod::JeodMemoryInterface](#).

Definition at line 125 of file `trick_memory_interface_xlate.cc`.

References `translate_name_to_addr()`.

8.11.3.5 get_container_id()

```
std::string jeod::JeodTrick10MemoryInterface::get_container_id (
    const ContainerListEntry & entry ) const [protected]
```

Construct the identifier for a checkpointable object.

Returns

Container ID

Parameters

in	<i>entry</i>	Container list entry
----	--------------	----------------------

Definition at line 196 of file `trick_memory_interface_chkpnt.cc`.

References `jeod::JeodTrickMemoryInterface::ContainerListEntry::elem_name`, `jeod::JeodTrickMemoryInterface::↵`
`ContainerListEntry::owner`, `jeod::JeodTrickMemoryInterface::ContainerListEntry::owner_type`, `translate_addr_to_↵`
`name()`, and `translate_name_to_addr()`.

Referenced by `checkpoint_containers()`, `register_container()`, and `restore_containers()`.

8.11.3.6 get_name_at_address()

```
const std::string jeod::JeodTrick10MemoryInterface::get_name_at_address (
    const void * addr,
    const JeodMemoryTypeDescriptor * tdesc ) const [override], [virtual]
```

Get the simulation name, if any, associated with the address.

Returns

Name of the address, if any

Parameters

in	<i>addr</i>	Address of memory whose name is to be found
in	<i>tdesc</i>	How to interpret address

Implements [jeod::JeodMemoryInterface](#).

Definition at line 86 of file `trick_memory_interface_xlate.cc`.

References [jeod::SimInterfaceMessages::interface_error](#), [translate_addr_to_name\(\)](#), and [translate_name_to_addr\(\)](#).

8.11.3.7 `get_trick_checkpoint_file()`

```
const std::string jeod::JeodTrick10MemoryInterface::get_trick_checkpoint_file (
    bool checkpoint ) [override], [virtual]
```

Get the name of the current [Trick](#) checkpoint file.

Returns

Name of the current [Trick](#) checkpoint file.

Units:

Parameters

in	<i>checkpoint</i>	True for checkpoint, false for restart
----	-------------------	--

Reimplemented from [jeod::JeodTrickMemoryInterface](#).

Definition at line 66 of file `trick_memory_interface_xlate.cc`.

Referenced by [jeod::BasicJeodTrickSimInterface::open_checkpoint_file\(\)](#), and [jeod::BasicJeodTrickSimInterface::open_restart_file\(\)](#).

8.11.3.8 `is_checkpoint_restart_supported()`

```
bool jeod::JeodTrick10MemoryInterface::is_checkpoint_restart_supported ( ) const [inline],
[override], [virtual]
```

The Trick10 memory interface supports checkpoint/restart.

Implements [jeod::JeodMemoryInterface](#).

Definition at line 141 of file `trick10_memory_interface.hh`.

Referenced by [jeod::BasicJeodTrickSimInterface::checkpoint_allocations\(\)](#), [jeod::BasicJeodTrickSimInterface::checkpoint_containers\(\)](#), [jeod::BasicJeodTrickSimInterface::open_checkpoint_file\(\)](#), [jeod::BasicJeodTrickSimInterface::open_restart_file\(\)](#), [jeod::BasicJeodTrickSimInterface::restore_allocations\(\)](#), and [jeod::BasicJeodTrickSimInterface::restore_containers\(\)](#).

8.11.3.9 operator=()

```
JeodTrick10MemoryInterface& jeod::JeodTrick10MemoryInterface::operator= (
    const JeodTrick10MemoryInterface & ) [delete]
```

8.11.3.10 register_container()

```
void jeod::JeodTrick10MemoryInterface::register_container (
    const void * owner,
    const JeodMemoryTypeDescriptor & owner_type,
    const std::string & elem_name,
    JeodCheckpointable & container ) [override], [virtual]
```

Register the checkpointable object with [Trick](#).

This function is typically called at construction or initialization time via JEOD_REGISTER_CHECKPOINTABLE.

Assumptions and Limitations

- The following unenforced assumptions are made:
 - Sim objects have been constructed and registered with [Trick](#).
 - Checkpointable objects are unique.
 - [Trick](#) has been pre-initialized.
 - Not in shutdown mode.

Enforcement of the above is the responsibility the simulation developer, the JEOD memory manager, and the simulation interface.

Parameters

in	<i>owner</i>	Owner of the container
in	<i>owner_type</i>	Owner type descriptor
in	<i>elem_name</i>	Container element
in, out	<i>container</i>	The container

Implements [jeod::JeodMemoryInterface](#).

Definition at line 80 of file `trick_memory_interface_chkpnt.cc`.

References [jeod::JeodTrickMemoryInterface::container_list](#), [get_container_id\(\)](#), and [jeod::SimInterfaceMessages::interface_error](#).

8.11.3.11 restore_allocations()

```
void jeod::JeodTrick10MemoryInterface::restore_allocations (
    JeodMemoryManager & memory_manager ) [override], [virtual]
```

Restore the allocated data per the checkpoint file.

Parameters

<i>in, out</i>	<i>memory_manager</i>	JEOD memory manager
----------------	-----------------------	---------------------

Reimplemented from [jeod::JeodTrickMemoryInterface](#).

Definition at line 479 of file `trick_memory_interface_chkpnt.cc`.

References [jeod::SectionedInputStream::activate\(\)](#), [jeod::JeodTrickMemoryInterface::container_list](#), [jeod::JeodSimulationInterface::get_checkpoint_reader\(\)](#), and [jeod::SimInterfaceMessages::interface_error](#).

Referenced by [jeod::BasicJeodTrickSimInterface::restore_allocations\(\)](#).

8.11.3.12 `restore_containers()`

```
void jeod::JeodTrick10MemoryInterface::restore_containers ( ) [override], [virtual]
```

Restore the checkpointable objects from the checkpoint file.

Reimplemented from [jeod::JeodTrickMemoryInterface](#).

Definition at line 303 of file `trick_memory_interface_chkpnt.cc`.

References [jeod::SectionedInputStream::activate\(\)](#), [jeod::JeodTrickMemoryInterface::container_list](#), [jeod::SectionedInputStream::deactivate\(\)](#), [jeod::JeodSimulationInterface::get_checkpoint_reader\(\)](#), [get_container_id\(\)](#), and [jeod::SimInterfaceMessages::interface_error](#).

Referenced by [jeod::BasicJeodTrickSimInterface::restore_containers\(\)](#).

8.11.3.13 `translate_addr_to_name()`

```
std::string jeod::JeodTrick10MemoryInterface::translate_addr_to_name (
    const void * addr,
    const ATTRIBUTES * attr ) const [protected]
```

Translate the given address to an address specification string, with the address interpreted in the context of the supplied attributes.

It is the attributes structure that resolves the A versus A.B versus A.B.C ambiguity.

Note

The attributes structure must be that of a pointer type.

Parameters

<i>addr</i>	The address to be translated.
<i>attr</i>	The context in which to interpret the address.

Returns

Address specification string, e.g., &foo.bar.baz[42]

Definition at line 145 of file trick_memory_interface_xlate.cc.

References jeod::SimInterfaceMessages::interface_error, jeod::JeodTrickMemoryInterface::pointer_attributes(), and trick_checkpoint_agent.

Referenced by get_container_id(), and get_name_at_address().

8.11.3.14 translate_name_to_addr()

```
void * jeod::JeodTrick10MemoryInterface::translate_name_to_addr (
    const std::string & spec ) const [protected]
```

Translate the given address specification string to an address.

This is the inverse of translate_addr_to_name.

Parameters

<i>spec</i>	The address specification to be interpreted.
-------------	--

Returns

Address corresponding to the address specification.

Definition at line 181 of file trick_memory_interface_xlate.cc.

References jeod::SimInterfaceMessages::interface_error.

Referenced by get_address_at_name(), get_container_id(), and get_name_at_address().

8.11.4 Friends And Related Function Documentation**8.11.4.1 init_attrjeod__JeodTrick10MemoryInterface**

```
void init_attrjeod__JeodTrick10MemoryInterface ( ) [friend]
```

8.11.4.2 InputProcessor

```
friend class InputProcessor [friend]
```

Definition at line 113 of file trick10_memory_interface.hh.

8.11.5 Field Documentation

8.11.5.1 trick_checkpoint_agent

```
Trick::ClassicCheckPointAgent* jeod::JeodTrick10MemoryInterface::trick_checkpoint_agent [protected]
```

[Trick](#) checkpoint agent.

```
trick_io(**)
```

Definition at line 174 of file trick10_memory_interface.hh.

Referenced by JeodTrick10MemoryInterface(), and translate_addr_to_name().

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

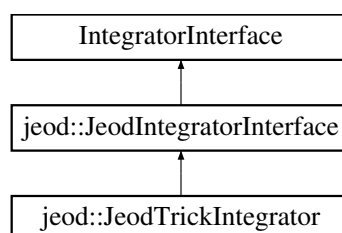
- [trick10_memory_interface.hh](#)
- [trick10_memory_interface.cc](#)
- [trick_memory_interface_chkpnt.cc](#)
- [trick_memory_interface_xlate.cc](#)

8.12 jeod::JeodTrickIntegrator Class Reference

A [JeodTrickIntegrator](#) specializes the [JeodIntegratorInterface](#) for use with [Trick](#) as the simulation engine.

```
#include <jeod_trick_integrator.hh>
```

Inheritance diagram for jeod::JeodTrickIntegrator:



Public Member Functions

- [JeodTrickIntegrator](#) ()
Default constructor.
- [~JeodTrickIntegrator](#) () override=default
Destructor.
- `er7_utils::Integration::Technique` [interpret_integration_type](#) (int integ_technique) const override
Interpret the integration technique.
- `Trick::Integrator *` [get_integrator](#) () override
Get the simulation engine's integrator.
- `double` [get_dt](#) () const override
Get the integration cycle time step.
- `bool` [get_first_step_derivs_flag](#) () const override
Get the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle.
- `void` [set_first_step_derivs_flag](#) (bool value) override
Set the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle.
- `void` [reset_first_step_derivs_flag](#) () override
Reset the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle.
- `void` [restore_first_step_derivs_flag](#) () override
Restore the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle to it's value prior to the most recent call to reset_first_step_derivs_flag.
- `void` [set_step_number](#) (unsigned int stepno) override
Set the step number within an integration cycle.
- `void` [set_time](#) (double sim_time) override
Update the time model given the simulation time.
- [JeodTrickIntegrator](#) (const [JeodTrickIntegrator](#) &)=delete
- [JeodTrickIntegrator](#) & [operator=](#) (const [JeodTrickIntegrator](#) &)=delete

Private Attributes

- `TrickJeodIntegrator` [trick_integrator](#)
Trick integration structure.
- `bool` [default_first_step_deriv](#) {}
Default value of trick_integrator.first_step_deriv.

Friends

- `class` [InputProcessor](#)
- `void` [init_attrjeod__JeodTrickIntegrator](#) ()

8.12.1 Detailed Description

A [JeodTrickIntegrator](#) specializes the [JeodIntegratorInterface](#) for use with [Trick](#) as the simulation engine.

Definition at line 118 of file `jeod_trick_integrator.hh`.

8.12.2 Constructor & Destructor Documentation

8.12.2.1 JeodTrickIntegrator() [1/2]

```
jeod::JeodTrickIntegrator::JeodTrickIntegrator ( ) [inline]
```

Default constructor.

Definition at line 126 of file jeod_trick_integrator.hh.

8.12.2.2 ~JeodTrickIntegrator()

```
jeod::JeodTrickIntegrator::~~JeodTrickIntegrator ( ) [override], [default]
```

Destructor.

8.12.2.3 JeodTrickIntegrator() [2/2]

```
jeod::JeodTrickIntegrator::JeodTrickIntegrator (
    const JeodTrickIntegrator & ) [delete]
```

8.12.3 Member Function Documentation

8.12.3.1 get_dt()

```
double jeod::JeodTrickIntegrator::get_dt ( ) const [inline], [override]
```

Get the integration cycle time step.

Returns

Simulation time delta t, in seconds

Definition at line 159 of file jeod_trick_integrator.hh.

8.12.3.2 get_first_step_derivs_flag()

```
bool jeod::JeodTrickIntegrator::get_first_step_derivs_flag ( ) const [inline], [override]
```

Get the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle.

Returns

Value of the first step derivatives flag

Definition at line 169 of file jeod_trick_integrator.hh.

8.12.3.3 `get_integrator()`

```
Trick::Integrator* jeod::JeodTrickIntegrator::get_integrator ( ) [inline], [override], [virtual]
```

Get the simulation engine's integrator.

Returns

Pointer to the simulation engine's integrator.

Implements [jeod::JeodIntegratorInterface](#).

Definition at line 150 of file `jeod_trick_integrator.hh`.

8.12.3.4 `interpret_integration_type()`

```
er7_utils::Integration::Technique jeod::JeodTrickIntegrator::interpret_integration_type (
    int integ_technique ) const [inline], [override], [virtual]
```

Interpret the integration technique.

Implements [jeod::JeodIntegratorInterface](#).

Definition at line 141 of file `jeod_trick_integrator.hh`.

8.12.3.5 `operator=()`

```
JeodTrickIntegrator& jeod::JeodTrickIntegrator::operator= (
    const JeodTrickIntegrator & ) [delete]
```

8.12.3.6 `reset_first_step_derivs_flag()`

```
void jeod::JeodTrickIntegrator::reset_first_step_derivs_flag ( ) [inline], [override]
```

Reset the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle.

Derivatives are always needed just after a reset. The behavior should revert to nominal after the reset has been performed.

Definition at line 190 of file `jeod_trick_integrator.hh`.

8.12.3.7 `restore_first_step_derivs_flag()`

```
void jeod::JeodTrickIntegrator::restore_first_step_derivs_flag ( ) [inline], [override]
```

Restore the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle to it's value prior to the most recent call to `reset_first_step_derivs_flag`.

Definition at line 201 of file `jeod_trick_integrator.hh`.

8.12.3.8 `set_first_step_derivs_flag()`

```
void jeod::JeodTrickIntegrator::set_first_step_derivs_flag (
    bool value ) [inline], [override]
```

Set the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle.

Parameters

in	<i>value</i>	Value of the first step derivatives flag
----	--------------	--

Definition at line 179 of file jeod_trick_integrator.hh.

8.12.3.9 set_step_number()

```
void jeod::JeodTrickIntegrator::set_step_number (
    unsigned int stepno ) [inline], [override]
```

Set the step number within an integration cycle.

Parameters

in	<i>stepno</i>	Step number
----	---------------	-------------

Definition at line 210 of file jeod_trick_integrator.hh.

8.12.3.10 set_time()

```
void jeod::JeodTrickIntegrator::set_time (
    double sim_time ) [inline], [override]
```

Update the time model given the simulation time.

Parameters

in	<i>sim_time</i>	Simulation time
----	-----------------	-----------------

Definition at line 219 of file jeod_trick_integrator.hh.

8.12.4 Friends And Related Function Documentation

8.12.4.1 init_attrjeod__JeodTrickIntegrator

```
void init_attrjeod__JeodTrickIntegrator ( ) [friend]
```

8.12.4.2 InputProcessor

```
friend class InputProcessor [friend]
```

Definition at line 120 of file jeod_trick_integrator.hh.

8.12.5 Field Documentation

8.12.5.1 default_first_step_deriv

```
bool jeod::JeodTrickIntegrator::default_first_step_deriv {} [private]
```

Default value of trick_integrator.first_step_deriv.

trick_units(-)

Definition at line 239 of file jeod_trick_integrator.hh.

8.12.5.2 trick_integrator

```
TrickJeodIntegrator jeod::JeodTrickIntegrator::trick_integrator [private]
```

Trick integration structure.

trick_units(-)

Definition at line 234 of file jeod_trick_integrator.hh.

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

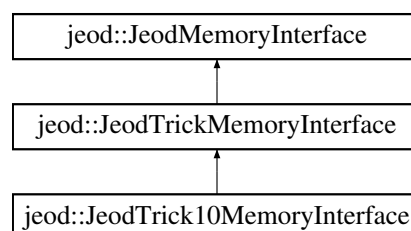
- [jeod_trick_integrator.hh](#)

8.13 jeod::JeodTrickMemoryInterface Class Reference

A TrickMemoryInterface implements the two required methods needed to register and deregister memory with the simulation engine, [Trick](#) in this case.

```
#include <trick_memory_interface.hh>
```

Inheritance diagram for jeod::JeodTrickMemoryInterface:



Data Structures

- struct [AllocationMapEntry](#)
Describes a chunk of JEOD-allocated memory.
- struct [ContainerListEntry](#)
Describes a Checkpointable object.

Public Member Functions

- [JeodTrickMemoryInterface](#) ()
JeodTrickMemoryInterface default constructor.
- [~JeodTrickMemoryInterface](#) () override
JeodTrickMemoryInterface destructor.
- [JeodTrickMemoryInterface](#) (const [JeodTrickMemoryInterface](#) &)=delete
- [JeodTrickMemoryInterface](#) & operator= (const [JeodTrickMemoryInterface](#) &)=delete
- void [set_mode](#) ([JeodSimulationInterface::Mode](#) new_mode)
Set the mode and perform mode transitions.
- std::string [construct_identifier](#) (uint32_t unique_id_number)
Construct an identifier for a chunk of JEOD-allocated memory.
- const struct ATTRIBUTES_tag * [find_attributes](#) (const std::string &type_name) const override
Find the attributes for a class in the symbol table.
- const struct ATTRIBUTES_tag * [find_attributes](#) (const std::type_info &data_type) const override
Find the attributes for a class in the symbol table.
- struct ATTRIBUTES_tag [primitive_attributes](#) (const std::type_info &data_type) const override
Create an attributes structure that represents a primitive type.
- struct ATTRIBUTES_tag [pointer_attributes](#) (const struct ATTRIBUTES_tag &target_attr) const override
Create an attributes structure that represents a pointer type.
- struct ATTRIBUTES_tag [void_pointer_attributes](#) () const override
Create an attributes structure that represents a void pointer.*
- struct ATTRIBUTES_tag [structure_attributes](#) (const struct ATTRIBUTES_tag *target_attr, std::size_t target←_size) const override
Create an attributes structure that represents a structured type.
- bool [register_allocation](#) (const void *addr, const [JeodMemoryItem](#) &item, const [JeodMemoryTypeDescriptor](#) &tdesc, const char *file, unsigned int line) override
Register newly allocated memory with Trick.
- void [deregister_allocation](#) (const void *addr, const [JeodMemoryItem](#) &item, const [JeodMemoryType](#)←Descriptor &tdesc, const char *file, unsigned int line) override
Delete Trick information about some pointer – but not the pointer itself.
- void [register_container](#) (const void *owner, const [JeodMemoryTypeDescriptor](#) &owner_type, const std::string &elem_name, [JeodCheckpointable](#) &container) override
Register the checkpointable object with Trick.
- void [deregister_container](#) (const void *owner, const [JeodMemoryTypeDescriptor](#) &owner_type, const std::←string &elem_name, [JeodCheckpointable](#) &container) override
Revoke the registrations performed by register_container.
- const std::string [get_name_at_address](#) (const void *addr, const [JeodMemoryTypeDescriptor](#) *tdesc) const override
Stubbed-out implementation of get_name_at_address for Trick implementations that do not fully support JEOD check-point/restart requirements.
- void * [get_address_at_name](#) (const std::string &name) const override
Stubbed-out implementation of get_address_at_name for Trick implementations that do not fully support JEOD check-point/restart requirements.
- bool [is_checkpoint_restart_supported](#) () const override

The generic [Trick](#) memory interface does not support checkpoint/restart.

- virtual const std::string [get_trick_checkpoint_file](#) (bool checkpoint)
Get the name of the current [Trick](#) checkpoint file.
- virtual void [checkpoint_containers](#) ()
Dump the container checkpointable objects to the checkpoint file.
- virtual void [restore_containers](#) ()
Restore the container checkpointables objects from the checkpoint file.
- virtual void [checkpoint_allocations](#) ()
Dump the allocation information to the checkpoint file.
- virtual void [restore_allocations](#) (JeodMemoryManager &memory_manager)
Restore the allocated data per the checkpoint file.

Protected Types

- using [AllocationMap](#) = std::map< uint32_t, [AllocationMapEntry](#) >
Maps JEOD-allocated data names to (type, size) pairs.
- using [ContainerList](#) = std::list< [ContainerListEntry](#) >
Container of a list of [ContainerListEntry](#) objects.

Protected Attributes

- void * [dlhandle](#) {}
dlhandle, from dlopen.
- [AllocationMap](#) [allocation_map](#)
Map of allocated names to type info.
- [ContainerList](#) [container_list](#)
List of container checkpointables.
- const std::string [id_prefix](#) {"jeod_alloc_"}
- const uint32_t [id_length](#) {6}
Number of digits in the numeric part of the unique identifier.
- [JeodSimulationInterface::Mode](#) [mode](#) {[JeodSimulationInterface::Construction](#)}
Simulation interface mode.

Friends

- class [InputProcessor](#)
- void [init_attrjeod__JeodTrickMemoryInterface](#) ()

8.13.1 Detailed Description

A [TrickMemoryInterface](#) implements the two required methods needed to register and deregister memory with the simulation engine, [Trick](#) in this case.

Definition at line 97 of file [trick_memory_interface.hh](#).

8.13.2 Member Typedef Documentation

8.13.2.1 AllocationMap

```
using jeod::JeodTrickMemoryInterface::AllocationMap = std::map<uint32_t, AllocationMapEntry>
[protected]
```

Maps JEOD-allocated data names to (type, size) pairs.

Definition at line 303 of file trick_memory_interface.hh.

8.13.2.2 ContainerList

```
using jeod::JeodTrickMemoryInterface::ContainerList = std::list<ContainerListEntry> [protected]
```

Container of a list of [ContainerListEntry](#) objects.

Definition at line 308 of file trick_memory_interface.hh.

8.13.3 Constructor & Destructor Documentation

8.13.3.1 JeodTrickMemoryInterface() [1/2]

```
jeod::JeodTrickMemoryInterface::JeodTrickMemoryInterface ( )
```

[JeodTrickMemoryInterface](#) default constructor.

Definition at line 53 of file trick_memory_interface.cc.

References [dlhandle](#), and [jeod::SimInterfaceMessages::implementation_error](#).

8.13.3.2 ~JeodTrickMemoryInterface()

```
jeod::JeodTrickMemoryInterface::~JeodTrickMemoryInterface ( ) [override]
```

[JeodTrickMemoryInterface](#) destructor.

Definition at line 70 of file trick_memory_interface.cc.

References [dlhandle](#).

8.13.3.3 JeodTrickMemoryInterface() [2/2]

```
jeod::JeodTrickMemoryInterface::JeodTrickMemoryInterface (
    const JeodTrickMemoryInterface & ) [delete]
```

8.13.4 Member Function Documentation

8.13.4.1 checkpoint_allocations()

```
virtual void jeod::JeodTrickMemoryInterface::checkpoint_allocations ( ) [inline], [virtual]
```

Dump the allocation information to the checkpoint file.

Note

The default implementation does nothing; checkpoint/restart is not supported by default.

Reimplemented in [jeod::JeodTrick10MemoryInterface](#).

Definition at line 208 of file `trick_memory_interface.hh`.

8.13.4.2 checkpoint_containers()

```
virtual void jeod::JeodTrickMemoryInterface::checkpoint_containers ( ) [inline], [virtual]
```

Dump the container checkpointable objects to the checkpoint file.

Note

The default implementation does nothing; checkpoint/restart is not supported by default.

Reimplemented in [jeod::JeodTrick10MemoryInterface](#).

Definition at line 192 of file `trick_memory_interface.hh`.

8.13.4.3 construct_identifier()

```
std::string jeod::JeodTrickMemoryInterface::construct_identifier (
    uint32_t unique_id_number )
```

Construct an identifier for a chunk of JEOD-allocated memory.

Returns

Identifier string

Parameters

<i>unique_id_number</i>	Identifier number
-------------------------	-------------------

Definition at line 93 of file `trick_memory_interface.cc`.

References `id_length`, and `id_prefix`.

Referenced by `jeod::JeodTrick10MemoryInterface::checkpoint_allocations()`, and `register_allocation()`.

8.13.4.4 deregister_allocation()

```
void jeod::JeodTrickMemoryInterface::deregister_allocation (
    const void * addr,
    const JeodMemoryItem & item,
    const JeodMemoryTypeDescriptor & tdesc,
    const char * file,
    unsigned int line ) [override], [virtual]
```

Delete [Trick](#) information about some pointer – but not the pointer itself.

Assumptions and Limitations

- Some other agent must freeing the memory at the input address itself. This function merely deletes [Trick](#)'s knowledge of that pointer.

Parameters

in	<i>addr</i>	Allocated memory
in	<i>item</i>	Description of the memory
in	<i>tdesc</i>	Description of the type
in	<i>file</i>	Source file containing JEOD_ALLOC
in	<i>line</i>	Line number containing JEOD_ALLOC

Implements [jeod::JeodMemoryInterface](#).

Definition at line 128 of file `trick_memory_interface_alloc.cc`.

References `allocation_map`, `jeod::SimInterfaceMessages::interface_error`, and `trick_MM`.

8.13.4.5 deregister_container()

```
void jeod::JeodTrickMemoryInterface::deregister_container (
    const void * owner,
    const JeodMemoryTypeDescriptor & owner_type,
```

```
const std::string & elem_name,
JeodCheckpointable & container ) [override], [virtual]
```

Revoke the registrations performed by register_container.

This function is typically called at destruction time via JEOD_DEREGISTER_CHECKPOINTABLE. This default implementation does nothing.

Parameters

in	<i>owner</i>	Owner of the container
in	<i>owner_type</i>	Owner type descriptor
in	<i>elem_name</i>	Container element
in, out	<i>container</i>	The container

Implements [jeod::JeodMemoryInterface](#).

Definition at line 130 of file trick_memory_interface.cc.

8.13.4.6 find_attributes() [1/2]

```
const struct ATTRIBUTES_tag * jeod::JeodTrickMemoryInterface::find_attributes (
    const std::string & type_name ) const [override], [virtual]
```

Find the attributes for a class in the symbol table.

Returns

Found attributes

Parameters

in	<i>type_name</i>	Demangled type name
----	------------------	---------------------

Implements [jeod::JeodMemoryInterface](#).

Definition at line 53 of file trick_memory_interface_attr.cc.

References dlhandle, and jeod::SimInterfaceMessages::interface_error.

Referenced by find_attributes().

8.13.4.7 find_attributes() [2/2]

```
const struct ATTRIBUTES_tag * jeod::JeodTrickMemoryInterface::find_attributes (
    const std::type_info & data_type ) const [override], [virtual]
```

Find the attributes for a class in the symbol table.

Returns

Found attributes

Parameters

in	<i>data_type</i>	Data type descriptor
----	------------------	----------------------

Implements [jeod::JeodMemoryInterface](#).

Definition at line 81 of file `trick_memory_interface_attrib.cc`.

References `find_attributes()`.

8.13.4.8 get_address_at_name()

```
void * jeod::JeodTrickMemoryInterface::get_address_at_name (
    const std::string & name ) const [override], [virtual]
```

Stubbed-out implementation of `get_address_at_name` for [Trick](#) implementations that do not fully support JEOD checkpoint/restart requirements.

Returns

Address of named item in memory

Parameters

<i>name</i>	Name of item to be found
-------------	--------------------------

Implements [jeod::JeodMemoryInterface](#).

Definition at line 158 of file `trick_memory_interface.cc`.

8.13.4.9 get_name_at_address()

```
const std::string jeod::JeodTrickMemoryInterface::get_name_at_address (
    const void * addr,
    const JeodMemoryTypeDescriptor * tdesc ) const [override], [virtual]
```

Stubbed-out implementation of `get_name_at_address` for [Trick](#) implementations that do not fully support JEOD checkpoint/restart requirements.

Returns

Name of the address, if any.

Parameters

<i>addr</i>	Address of memory whose name is to be found
<i>tdesc</i>	How to interpret address

Implements [jeod::JeodMemoryInterface](#).

Definition at line 145 of file `trick_memory_interface.cc`.

8.13.4.10 `get_trick_checkpoint_file()`

```
virtual const std::string jeod::JeodTrickMemoryInterface::get_trick_checkpoint_file (
    bool checkpoint ) [inline], [virtual]
```

Get the name of the current [Trick](#) checkpoint file.

Parameters

in	<i>checkpoint</i>	True for checkpoint, false for restart
----	-------------------	--

Returns

Current checkpoint file, or the empty string.

Note

The default implementation always returns the empty string; checkpoint/restart is not supported by default.

Reimplemented in [jeod::JeodTrick10MemoryInterface](#).

Definition at line 181 of file `trick_memory_interface.hh`.

8.13.4.11 `is_checkpoint_restart_supported()`

```
bool jeod::JeodTrickMemoryInterface::is_checkpoint_restart_supported ( ) const [inline],
[override], [virtual]
```

The generic [Trick](#) memory interface does not support checkpoint/restart.

Implements [jeod::JeodMemoryInterface](#).

Definition at line 168 of file `trick_memory_interface.hh`.

8.13.4.12 operator=()

```
JeodTrickMemoryInterface& jeod::JeodTrickMemoryInterface::operator= (
    const JeodTrickMemoryInterface & ) [delete]
```

8.13.4.13 pointer_attributes()

```
struct ATTRIBUTES_tag jeod::JeodTrickMemoryInterface::pointer_attributes (
    const struct ATTRIBUTES_tag & target_attr ) const [override], [virtual]
```

Create an attributes structure that represents a pointer type.

Returns

Constructed pointer attributes.

Parameters

in	<i>target_attr</i>	Pointed-to type attributes.
----	--------------------	-----------------------------

Implements [jeod::JeodMemoryInterface](#).

Definition at line 213 of file `trick_memory_interface_attr.cc`.

Referenced by `jeod::JeodTrick10MemoryInterface::translate_addr_to_name()`.

8.13.4.14 primitive_attributes()

```
struct ATTRIBUTES_tag jeod::JeodTrickMemoryInterface::primitive_attributes (
    const std::type_info & data_type ) const [override], [virtual]
```

Create an attributes structure that represents a primitive type.

Returns

Constructed attributes.

Parameters

in	<i>data_type</i>	Data type descriptor
----	------------------	----------------------

Implements [jeod::JeodMemoryInterface](#).

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

References [jeod::SimInterfaceMessages::interface_error](#).

8.13.4.15 register_allocation()

```
bool jeod::JeodTrickMemoryInterface::register_allocation (
    const void * addr,
    const JeodMemoryItem & item,
    const JeodMemoryTypeDescriptor & tdesc,
    const char * file,
    unsigned int line ) [override], [virtual]
```

Register newly allocated memory with [Trick](#).

Assumptions and Limitations

- Memory was indeed allocated.
- The input address is not null.
- The number of elements is positive.

Returns

True if registered

Parameters

in	<i>addr</i>	Allocated memory
in	<i>item</i>	Description of the memory
in	<i>tdesc</i>	Description of the type
in	<i>file</i>	Source file containing JEOD_ALLOC
in	<i>line</i>	Line number containing JEOD_ALLOC

Implements [jeod::JeodMemoryInterface](#).

Definition at line 73 of file `trick_memory_interface_alloc.cc`.

References [allocation_map](#), [construct_identifier\(\)](#), [jeod::SimInterfaceMessages::interface_error](#), and [trick_MM](#).

8.13.4.16 register_container()

```
void jeod::JeodTrickMemoryInterface::register_container (
    const void * owner,
    const JeodMemoryTypeDescriptor & owner_type,
    const std::string & elem_name,
    JeodCheckpointable & container ) [override], [virtual]
```

Register the checkpointable object with [Trick](#).

This function is typically called at construction or initialization time via `JEOD_REGISTER_CHECKPOINTABLE`. This default implementation does nothing.

Parameters

in	<i>owner</i>	Owner of the container
in	<i>owner_type</i>	Owner type descriptor
in	<i>elem_name</i>	Container element
in, out	<i>container</i>	The container

Implements [jeod::JeodMemoryInterface](#).

Definition at line 112 of file `trick_memory_interface.cc`.

8.13.4.17 `restore_allocations()`

```
virtual void jeod::JeodTrickMemoryInterface::restore_allocations (
    JeodMemoryManager & memory_manager ) [inline], [virtual]
```

Restore the allocated data per the checkpoint file.

Parameters

<i>memory_manager</i>	JEOD memory manager
-----------------------	---------------------

Note

The default implementation does nothing; checkpoint/restart is not supported by default.

Reimplemented in [jeod::JeodTrick10MemoryInterface](#).

Definition at line 217 of file `trick_memory_interface.hh`.

8.13.4.18 `restore_containers()`

```
virtual void jeod::JeodTrickMemoryInterface::restore_containers ( ) [inline], [virtual]
```

Restore the container checkpointables objects from the checkpoint file.

Note

The default implementation does nothing; checkpoint/restart is not supported by default.

Reimplemented in [jeod::JeodTrick10MemoryInterface](#).

Definition at line 200 of file `trick_memory_interface.hh`.

8.13.4.19 `set_mode()`

```
void jeod::JeodTrickMemoryInterface::set_mode (
    JeodSimulationInterface::Mode new_mode )
```

Set the mode and perform mode transitions.

Parameters

<i>new_mode</i>	New mode
-----------------	----------

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

References mode.

Referenced by `jeod::BasicJeodTrickSimInterface::set_mode()`.

8.13.4.20 structure_attributes()

```
struct ATTRIBUTES_tag jeod::JeodTrickMemoryInterface::structure_attributes (
    const struct ATTRIBUTES_tag * target_attr,
    std::size_t target_size ) const [override], [virtual]
```

Create an attributes structure that represents a structured type.

Returns

Constructed structure attributes.

Parameters

in	<i>target_attr</i>	Return value from <code>find_attributes</code> .
in	<i>target_size</i>	Structure size.

Implements [jeod::JeodMemoryInterface](#).

Definition at line 285 of file `trick_memory_interface_attr.cc`.

8.13.4.21 void_pointer_attributes()

```
struct ATTRIBUTES_tag jeod::JeodTrickMemoryInterface::void_pointer_attributes ( ) const [override],
[virtual]
```

Create an attributes structure that represents a `void*` pointer.

Returns

Constructed pointer attributes.

Implements [jeod::JeodMemoryInterface](#).

Definition at line 263 of file `trick_memory_interface_attr.cc`.

8.13.5 Friends And Related Function Documentation

8.13.5.1 init_attrjeod__JeodTrickMemoryInterface

```
void init_attrjeod__JeodTrickMemoryInterface ( ) [friend]
```

8.13.5.2 InputProcessor

```
friend class InputProcessor [friend]
```

Definition at line 99 of file `trick_memory_interface.hh`.

8.13.6 Field Documentation

8.13.6.1 allocation_map

```
AllocationMap jeod::JeodTrickMemoryInterface::allocation_map [protected]
```

Map of allocated names to type info.

```
trick_io(**)
```

Definition at line 320 of file `trick_memory_interface.hh`.

Referenced by `jeod::JeodTrick10MemoryInterface::checkpoint_allocations()`, `deregister_allocation()`, and `register_allocation()`.

8.13.6.2 container_list

```
ContainerList jeod::JeodTrickMemoryInterface::container_list [protected]
```

List of container checkpointables.

```
trick_io(**)
```

Definition at line 325 of file `trick_memory_interface.hh`.

Referenced by `jeod::JeodTrick10MemoryInterface::checkpoint_allocations()`, `jeod::JeodTrick10MemoryInterface::checkpoint_containers()`, `jeod::JeodTrick10MemoryInterface::deregister_container()`, `jeod::JeodTrick10MemoryInterface::register_container()`, `jeod::JeodTrick10MemoryInterface::restore_allocations()`, and `jeod::JeodTrick10MemoryInterface::restore_containers()`.

8.13.6.3 dlhandle

```
void* jeod::JeodTrickMemoryInterface::dlhandle {} [protected]
```

dlhandle, from dlopen.

trick_io(**)

Definition at line 315 of file trick_memory_interface.hh.

Referenced by find_attributes(), JeodTrickMemoryInterface(), and ~JeodTrickMemoryInterface().

8.13.6.4 id_length

```
const uint32_t jeod::JeodTrickMemoryInterface::id_length {6} [protected]
```

Number of digits in the numeric part of the unique identifier.

trick_io(*o) trick_units(-)

Definition at line 335 of file trick_memory_interface.hh.

Referenced by construct_identifier().

8.13.6.5 id_prefix

```
const std::string jeod::JeodTrickMemoryInterface::id_prefix {"jeod_alloc_"} [protected]
```

Prefix used for constructing a unique name for JEOD-allocated memory.

trick_io(*o) trick_units(-)

Definition at line 330 of file trick_memory_interface.hh.

Referenced by construct_identifier().

8.13.6.6 mode

```
JeodSimulationInterface::Mode jeod::JeodTrickMemoryInterface::mode {JeodSimulationInterface::Construction}  
[protected]
```

Simulation interface mode.

trick_units(-)

Definition at line 340 of file trick_memory_interface.hh.

Referenced by set_mode().

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

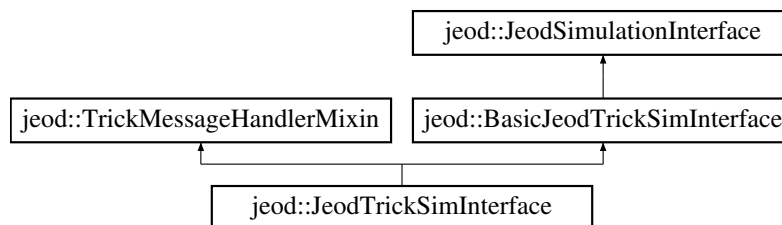
- [trick_memory_interface.hh](#)
- [trick_memory_interface.cc](#)
- [trick_memory_interface_alloc.cc](#)
- [trick_memory_interface_attrib.cc](#)

8.14 jeod::JeodTrickSimInterface Class Reference

A JEODTrickSimInterface implements the required capabilities of the generic [JeodSimulationInterface](#) in a [Trick](#) simulation environment.

```
#include <trick_sim_interface.hh>
```

Inheritance diagram for jeod::JeodTrickSimInterface:



Public Member Functions

- [JeodTrickSimInterface](#) ()
Non-default constructor.
- [~JeodTrickSimInterface](#) () override=default
Destructor.
- [JeodTrickSimInterface](#) (const [JeodTrickSimInterface](#) &)=delete
- [JeodTrickSimInterface](#) & operator= (const [JeodTrickSimInterface](#) &)=delete

Friends

- class [InputProcessor](#)
- void [init_attrjeod__JeodTrickSimInterface](#) ()

Additional Inherited Members

8.14.1 Detailed Description

A JEODTrickSimInterface implements the required capabilities of the generic [JeodSimulationInterface](#) in a [Trick](#) simulation environment.

By virtue of member data ownership, the class creates the requisite MessageHandler and MemoryManager and does so in the correct order.

Definition at line 259 of file [trick_sim_interface.hh](#).

8.14.2 Constructor & Destructor Documentation

8.14.2.1 JeodTrickSimInterface() [1/2]

```
jeod::JeodTrickSimInterface::JeodTrickSimInterface ( ) [inline], [explicit]
```

Non-default constructor.

Definition at line 266 of file trick_sim_interface.hh.

8.14.2.2 ~JeodTrickSimInterface()

```
jeod::JeodTrickSimInterface::~~JeodTrickSimInterface ( ) [override], [default]
```

Destructor.

8.14.2.3 JeodTrickSimInterface() [2/2]

```
jeod::JeodTrickSimInterface::JeodTrickSimInterface (
    const JeodTrickSimInterface & ) [delete]
```

8.14.3 Member Function Documentation**8.14.3.1 operator=()**

```
JeodTrickSimInterface& jeod::JeodTrickSimInterface::operator= (
    const JeodTrickSimInterface & ) [delete]
```

8.14.4 Friends And Related Function Documentation**8.14.4.1 init_attrjeod__JeodTrickSimInterface**

```
void init_attrjeod__JeodTrickSimInterface ( ) [friend]
```


8.14.4.2 InputProcessor

```
friend class InputProcessor [friend]
```

Definition at line 262 of file `trick_sim_interface.hh`.

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

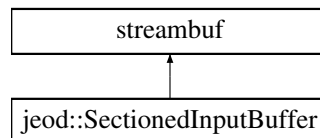
- [trick_sim_interface.hh](#)

8.15 jeod::SectionedInputBuffer Class Reference

A [SectionedInputBuffer](#) is a `std::streambuf` that reads from a section in a checkpoint file.

```
#include <checkpoint_input_manager.hh>
```

Inheritance diagram for `jeod::SectionedInputBuffer`:



Public Member Functions

- [~SectionedInputBuffer](#) () override=default

Destructor.

- [SectionedInputBuffer](#) (const [SectionedInputBuffer](#) &)=delete
- [SectionedInputBuffer](#) & operator= (const [SectionedInputBuffer](#) &)=delete
- bool [operator!](#) () const

Conversion to boolean.

Private Member Functions

- [SectionedInputBuffer](#) ()

Default constructor.

- void [activate](#) (std::ifstream &stream, std::size_t spos, std::size_t epos)

Activate the object.

- void [deactivate](#) ()

Deactivate the object.

- std::streambuf::int_type [underflow](#) () override

Get a character in the case of depletion of the read buffer.

Private Attributes

- `std::filebuf * file_buf {}`
The file buffer that reads from the checkpoint file.
- `size_t start_pos {}`
The position of the start of the contents of the checkpoint file section being read by this object.
- `size_t end_pos {}`
The position just after the end of the contents of the checkpoint file section being read by this object.
- `size_t curr_pos {}`
The current position of the file_buf reader.
- `bool at_eof {true}`
At EOF in the file or in the section?
- `char buf {}`
Input buffer.

Friends

- class `SectionedInputStream`

8.15.1 Detailed Description

A `SectionedInputBuffer` is a `std::streambuf` that reads from a section in a checkpoint file.

This class will indicate EOF when the input pointer in the checkpoint file file buffer goes beyond the end of the section.

This is a barebones implementation. It does not provide buffering, it does not support seek and tell, and it does not support putback or unget.

Note that with the exception of the destructor and the inherited members from `std::streambuf`, *everything* in this class is private. This class is not extensible.

Definition at line 86 of file `checkpoint_input_manager.hh`.

8.15.2 Constructor & Destructor Documentation

8.15.2.1 `~SectionedInputBuffer()`

```
jeod::SectionedInputBuffer::~SectionedInputBuffer ( ) [override], [default]
```

Destructor.

For now, this does nothing.

8.15.2.2 SectionedInputBuffer() [1/2]

```
jeod::SectionedInputBuffer::SectionedInputBuffer (
    const SectionedInputBuffer & ) [delete]
```

8.15.2.3 SectionedInputBuffer() [2/2]

```
jeod::SectionedInputBuffer::SectionedInputBuffer ( ) [private]
```

Default constructor.

This constructor creates an empty [SectionedInputBuffer](#) – one that will return EOF on the first read attempt. An empty [SectionedInputBuffer](#) has two purposes:

- As the basis for a copy constructor of a containing stream, and
- As a graceful means of handling of erroneous conditions.

Definition at line 43 of file `checkpoint_input_manager.cc`.

8.15.3 Member Function Documentation

8.15.3.1 activate()

```
void jeod::SectionedInputBuffer::activate (
    std::ifstream & stream,
    std::size_t spos,
    std::size_t epos ) [private]
```

Activate the object.

Note

Using the object for reading prior to activation will result in EOF.

Parameters

in	<i>stream</i>	Checkpoint file input file stream
in	<i>spos</i>	Section data start position
in	<i>epos</i>	Section data end position

Definition at line 56 of file `checkpoint_input_manager.cc`.

References `at_eof`, `curr_pos`, `end_pos`, `file_buf`, and `start_pos`.

Referenced by `jeod::SectionedInputStream::activate()`.

8.15.3.2 deactivate()

```
void jeod::SectionedInputBuffer::deactivate ( ) [inline], [private]
```

Deactivate the object.

Used to force a badly behaving stream to disconnect.

Definition at line 124 of file `checkpoint_input_manager.hh`.

References `at_eof`, and `file_buf`.

Referenced by `jeod::SectionedInputStream::deactivate()`.

8.15.3.3 operator!()

```
bool jeod::SectionedInputBuffer::operator! ( ) const [inline]
```

Conversion to boolean.

Returns

False if object is OK.

Definition at line 105 of file `checkpoint_input_manager.hh`.

References `file_buf`.

8.15.3.4 operator=()

```
SectionedInputBuffer& jeod::SectionedInputBuffer::operator= (
    const SectionedInputBuffer & ) [delete]
```

8.15.3.5 underflow()

```
std::streambuf::int_type jeod::SectionedInputBuffer::underflow ( ) [override], [private]
```

Get a character in the case of depletion of the read buffer.

For now, the buffer is always depleted.

Returns

Character read from the underlying file.

Definition at line 69 of file checkpoint_input_manager.cc.

References at_eof, buf, curr_pos, end_pos, and file_buf.

8.15.4 Friends And Related Function Documentation

8.15.4.1 SectionedInputStream

```
friend class SectionedInputStream [friend]
```

Definition at line 88 of file checkpoint_input_manager.hh.

8.15.5 Field Documentation

8.15.5.1 at_eof

```
bool jeod::SectionedInputBuffer::at_eof {true} [private]
```

At EOF in the file or in the section?

trick_io(**)

Definition at line 160 of file checkpoint_input_manager.hh.

Referenced by activate(), deactivate(), and underflow().

8.15.5.2 buf

```
char jeod::SectionedInputBuffer::buf {} [private]
```

Input buffer.

trick_io(**)

Definition at line 165 of file checkpoint_input_manager.hh.

Referenced by underflow().

8.15.5.3 curr_pos

```
size_t jeod::SectionedInputBuffer::curr_pos {} [private]
```

The current position of the file_buf reader.

trick_io(**)

Definition at line 155 of file checkpoint_input_manager.hh.

Referenced by activate(), and underflow().

8.15.5.4 end_pos

```
size_t jeod::SectionedInputBuffer::end_pos {} [private]
```

The position just after the end of the contents of the checkpoint file section being read by this object.

trick_io(**)

Definition at line 150 of file checkpoint_input_manager.hh.

Referenced by activate(), and underflow().

8.15.5.5 file_buf

```
std::filebuf* jeod::SectionedInputBuffer::file_buf {} [private]
```

The file buffer that reads from the checkpoint file.

trick_io(**)

Definition at line 138 of file checkpoint_input_manager.hh.

Referenced by activate(), deactivate(), operator!(), and underflow().

8.15.5.6 start_pos

```
size_t jeod::SectionedInputBuffer::start_pos {} [private]
```

The position of the start of the contents of the checkpoint file section being read by this object.

```
trick_io(**)
```

Definition at line 144 of file checkpoint_input_manager.hh.

Referenced by activate().

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

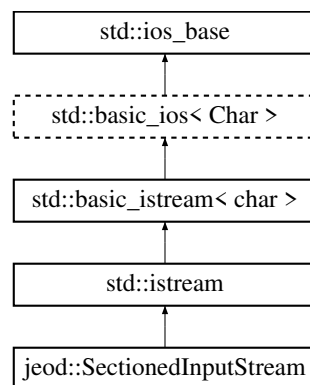
- [checkpoint_input_manager.hh](#)
- [checkpoint_input_manager.cc](#)

8.16 jeod::SectionedInputStream Class Reference

A [SectionedInputStream](#) is a `std::istream` that reads from a section in a checkpoint file.

```
#include <checkpoint_input_manager.hh>
```

Inheritance diagram for `jeod::SectionedInputStream`:



Public Member Functions

- [SectionedInputStream](#) ()
Construct a [SectionedInputStream](#) object.
- [SectionedInputStream](#) (const [SectionedInputStream](#) &)
Construct a [SectionedInputStream](#) object by copying from another.
- [~SectionedInputStream](#) () override
Destruct a [SectionedInputStream](#) object.
- [SectionedInputStream](#) & operator= (const [SectionedInputStream](#) &)=delete
- bool [is_activatable](#) () const
Determine if the stream is able to be activated.
- bool [activate](#) ()
Activate the object.
- void [deactivate](#) ()
Deactivate the object.
- bool [operator!](#) () const
Conversion to boolean.
- [operator void *](#) () const
Conversion to void.*

Private Member Functions

- [SectionedInputStream](#) ([CheckPointInputManager](#) *mngr, std::ifstream &fstream, std::size_t spos, std::size_t epos)

Construct a [SectionedInputStream](#) object that is connected to a file stream and to a [CheckPointInputManager](#).

Private Attributes

- [SectionedInputBuffer](#) sectbuf

The std::streambuf that does the reading from the file.

- [CheckPointInputManager](#) * manager {}

The input manager that created this object.

- std::ifstream * stream {}

The C++ file stream that reads from the checkpoint file.

- size_t start_pos {}

The position of the start of the contents of the checkpoint file section being read by this object.

- size_t end_pos {}

The position just after the end of the contents of the checkpoint file section being read by this object.

- bool is_copy {}

Is this a copy of some other [SectionedInputStream](#)? Copies of copies are verboten.

- bool is_active {}

Is this an active object? In the end, there can be only one.

Friends

- class [CheckPointInputManager](#)

8.16.1 Detailed Description

A [SectionedInputStream](#) is a std::istream that reads from a section in a checkpoint file.

This class will indicate EOF when the input pointer in the checkpoint file file buffer goes beyond the end of the section.

This is a barebones implementation. It does not provide buffering, it does not support seek and tell, and it does not support putback or unget.

Usage

A [SectionedInputStream](#) object is used in a preload_checkpoint or restart job to read and then act on contents stored in a checkpoint file.


```

return_type function_name (
    SomeStructureType & stuff_to_restore)
{
    std::string section_name;
    double number;
    char c_style_line[256];
    std::string cpp_line;
    char character;
    int char_as_int;

    std::string section_name;
    // Set to name of the checkpoint section

    // Construct a checkpoint input stream.
    // Notes:
    // - This object must go out of scope by the end of the job.
    // - DO NOT make a copy of this object.
    // - DO NOT save a pointer to this object in a permanent structure.
    // - The code below assumes that function_name is called as a
    //   preload_checkpoint or a restart job.
    SectionedInputStream reader (
        JeodSimulationInterface::get_checkpoint_reader(
            section_name));

    // Activate the reader.
    // Fail to do so and you'll get EOF on the first read.
    reader.activate();

    // You can use the C++ operator >> to read various kinds of data ...
    reader >> number;

    // ... even data structures if the structure has a deserializer.
    reader >> stuff_to_restore;

    // Lines can be read with the getline member or std::getline global.
    reader.getline (c_style_line, 255);
    std::getline (reader, cpp_line);

    // Individual characters can be read in a variety of ways.
    reader >> std::noskipws >> character;
    reader.get (character);
    char_as_int = reader.rdbuf()->sbumpc();

    // A bunch of numbers can be read using operator >>:
    while (!! (reader >> number)) {
        stuff_to_restore.add_number (number);
    }

    // An alternative is to implicitly use operator void*:
    while (reader >> number) {
        stuff_to_restore.add_number (number);
    }

    // The file can be scanned via getline, here using the bang-bang trick:
    while (!! std::getline (reader, cpp_string)) {
        process_line (cpp_string);
    }

    // Same as the above, but implicitly using operator void*:
    while (std::getline (reader, cpp_string)) {
        process_line (cpp_string);
    }

    // The file can be processed a character at a time.
    // Once again, either the bang-bang trick or operator void* can be
    // used to check for EOF.
    while (!! std::get (reader, character)) {
        stuff_to_restore.add_char (character);
    }

    // Yet another alternative is to test for EOF using sbumpc:
    while ((char_as_int = rdbuf->sbumpc()) != EOF) {
        stuff_to_restore.add_char ((char)char_as_int);
    }

    // Or use sgetc/sbumpc if the above grates too much:
    while (reader.rdbuf->sgetc() != EOF) {
        stuff_to_restore.add_char ((char)reader.rdbuf()->sbumpc());
    }
}

```

Diagnosing problems

- Nothing is being read. This can be caused by several problems, described below.

- Is the JEOD checkpoint file open for input?
Checkpoint file sections can only be read from a JEOD checkpoint file that is open for input. In a [Trick](#) context, the checkpoint file is only open for preload_checkpoint and restart jobs. Reading from a checkpoint file in other contexts won't work.
- Are multiple threads trying to read from the same checkpoint file?
Don't do that. This package is not thread-safe.
- Have you cached some another active checkpoint reader somewhere?
Don't do that, either. Only one reader can be active at a time.
- Is the checkpoint file section in the checkpoint file?
You will get a diagnostic message if the section doesn't exist.
- Is the checkpoint reader viable?
The above problems will result in a non-viable checkpoint reader. The method [is_activatable\(\)](#) can be called prior to calling [activate\(\)](#) to check whether the stream is viable.
- Did you call reader.activate()?
Whether compilers make two different objects in the construction of the [SectionedInputStream](#) or just one object depends on the compiler and on the optimization level. Making the package robustly handle the complexities of RVO (return value optimization) was too much for the author of the package. The call to reader.activate() is essential.
- Did the call to reader.activate() work?
The method [activate\(\)](#) returns true or false to indicate success or failure. While the above code did not check status, doing so is a good idea.
- Did you call reader.deactivate()?
Don't do that until you are done reading. The call to [deactivate\(\)](#) is irreversible.
- Did you mix scanned input with line reading?
As with any other stream, operator >> will mark the stream as failed if the operator fails to parse.

Definition at line 295 of file checkpoint_input_manager.hh.

8.16.2 Constructor & Destructor Documentation

8.16.2.1 SectionedInputStream() [1/3]

```
jeod::SectionedInputStream::SectionedInputStream ( )
```

Construct a [SectionedInputStream](#) object.

Note

This default constructor creates a disconnected and hence unusable stream. Usable streams are created by the non-default constructor.

Definition at line 118 of file checkpoint_input_manager.cc.

8.16.2.2 SectionedInputStream() [2/3]

```
jeod::SectionedInputStream::SectionedInputStream (
    const SectionedInputStream & source )
```

Construct a [SectionedInputStream](#) object by copying from another.

Parameters

in	<i>source</i>	Source object
----	---------------	---------------

Definition at line 147 of file `checkpoint_input_manager.cc`.

References `jeod::SimInterfaceMessages::implementation_error`, `is_active`, `is_copy`, `manager`, and `stream`.

8.16.2.3 ~SectionedInputStream()

```
jeod::SectionedInputStream::~~SectionedInputStream ( ) [override]
```

Destruct a [SectionedInputStream](#) object.

Definition at line 172 of file `checkpoint_input_manager.cc`.

References `jeod::CheckPointInputManager::deregister_reader()`, `is_active`, and `manager`.

8.16.2.4 SectionedInputStream() [3/3]

```
jeod::SectionedInputStream::SectionedInputStream (
    CheckPointInputManager * mngr,
    std::ifstream & ifstream,
    std::size_t spos,
    std::size_t epos ) [private]
```

Construct a [SectionedInputStream](#) object that is connected to a file stream and to a [CheckPointInputManager](#).

Parameters

in	<i>mngr</i>	The stream manager
in	<i>ifstream</i>	The input file stream
in	<i>spos</i>	Start position of section data
in	<i>epos</i>	End position of section data

Definition at line 131 of file `checkpoint_input_manager.cc`.

8.16.3 Member Function Documentation

8.16.3.1 activate()

```
bool jeod::SectionedInputStream::activate ( )
```

Activate the object.

Note

Using the object for reading prior to activation will result in EOF.

Returns

True if activated.

Definition at line 206 of file checkpoint_input_manager.cc.

References jeod::SectionedInputBuffer::activate(), end_pos, jeod::SimInterfaceMessages::implementation_error, is_active, manager, jeod::CheckPointInputManager::register_reader(), sectbuf, start_pos, and stream.

Referenced by jeod::JeodTrick10MemoryInterface::restore_allocations(), and jeod::JeodTrick10MemoryInterface↵::restore_containers().

8.16.3.2 deactivate()

```
void jeod::SectionedInputStream::deactivate ( )
```

Deactivate the object.

Note

Deactivation is undoable.

Definition at line 254 of file checkpoint_input_manager.cc.

References jeod::SectionedInputBuffer::deactivate(), jeod::CheckPointInputManager::deregister_reader(), is↵active, manager, sectbuf, and stream.

Referenced by jeod::CheckPointInputManager::create_trick_section_reader(), and jeod::JeodTrick10Memory↵Interface::restore_containers().

8.16.3.3 is_activatable()

```
bool jeod::SectionedInputStream::is_activatable ( ) const
```

Determine if the stream is able to be activated.

Returns

True if object can be activated.

Definition at line 184 of file checkpoint_input_manager.cc.

References jeod::CheckPointInputManager::have_active_reader(), is_active, manager, and stream.

8.16.3.4 operator void *()

```
jeod::SectionedInputStream::operator void * ( ) const [inline]
```

Conversion to void*.

This method provides an alternative to the bang-bang trick to determine if the object is OK.

Returns

this pointer (cast to void*) if object is OK, NULL otherwise.

Definition at line 337 of file checkpoint_input_manager.hh.

8.16.3.5 operator!()

```
bool jeod::SectionedInputStream::operator! ( ) const [inline]
```

Conversion to boolean.

Use the bang-bang trick to determine if the object is OK.

Returns

False if object is OK, true if something is wrong.

Definition at line 326 of file checkpoint_input_manager.hh.

References `is_active`, `sectbuf`, and `stream`.

8.16.3.6 operator=()

```
SectionedInputStream& jeod::SectionedInputStream::operator= (
    const SectionedInputStream & ) [delete]
```

8.16.4 Friends And Related Function Documentation**8.16.4.1 CheckPointInputManager**

```
friend class CheckPointInputManager [friend]
```

Definition at line 297 of file checkpoint_input_manager.hh.

8.16.5 Field Documentation

8.16.5.1 end_pos

```
size_t jeod::SectionedInputStream::end_pos {} [private]
```

The position just after the end of the contents of the checkpoint file section being read by this object.

trick_io(**)

Definition at line 381 of file checkpoint_input_manager.hh.

Referenced by activate().

8.16.5.2 is_active

```
bool jeod::SectionedInputStream::is_active {} [private]
```

Is this an active object? In the end, there can be only one.

trick_io(**)

Definition at line 393 of file checkpoint_input_manager.hh.

Referenced by activate(), deactivate(), is_activatable(), operator!(), SectionedInputStream(), and ~SectionedInputStream().

8.16.5.3 is_copy

```
bool jeod::SectionedInputStream::is_copy {} [private]
```

Is this a copy of some other [SectionedInputStream](#)? Copies of copies are verboten.

trick_io(**)

Definition at line 387 of file checkpoint_input_manager.hh.

Referenced by SectionedInputStream().

8.16.5.4 manager

```
CheckpointInputManager* jeod::SectionedInputStream::manager {} [private]
```

The input manager that created this object.

trick_io(**)

Definition at line 364 of file checkpoint_input_manager.hh.

Referenced by activate(), deactivate(), is_activatable(), SectionedInputStream(), and ~SectionedInputStream().

8.16.5.5 sectbuf

```
SectionedInputBuffer jeod::SectionedInputStream::sectbuf [private]
```

The std::streambuf that does the reading from the file.

trick_io(**)

Definition at line 359 of file checkpoint_input_manager.hh.

Referenced by activate(), deactivate(), and operator!().

8.16.5.6 start_pos

```
size_t jeod::SectionedInputStream::start_pos {} [private]
```

The position of the start of the contents of the checkpoint file section being read by this object.

trick_io(**)

Definition at line 375 of file checkpoint_input_manager.hh.

Referenced by activate().

8.16.5.7 stream

```
std::ifstream* jeod::SectionedInputStream::stream {} [private]
```

The C++ file stream that reads from the checkpoint file.

trick_io(**)

Definition at line 369 of file checkpoint_input_manager.hh.

Referenced by activate(), deactivate(), is_activatable(), operator!(), and SectionedInputStream().

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

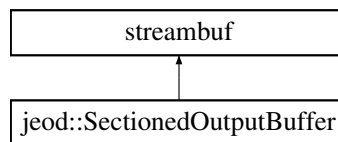
- [checkpoint_input_manager.hh](#)
- [checkpoint_input_manager.cc](#)

8.17 jeod::SectionedOutputBuffer Class Reference

A [SectionedOutputBuffer](#) is a `std::streambuf` that writes a section of a checkpoint file.

```
#include <checkpoint_output_manager.hh>
```

Inheritance diagram for `jeod::SectionedOutputBuffer`:



Public Member Functions

- [~SectionedOutputBuffer](#) () override=default

Destructor.

- [SectionedOutputBuffer](#) (const [SectionedOutputBuffer](#) &)=delete
- [SectionedOutputBuffer](#) & [operator=](#) (const [SectionedOutputBuffer](#) &)=delete
- bool [operator!](#) () const

Conversion to boolean.

Private Member Functions

- [SectionedOutputBuffer](#) ()

Default constructor.

- [SectionedOutputBuffer](#) (std::ofstream *stream)
- void [activate](#) (std::ofstream &stream)

Activate the object.

- void [deactivate](#) ()

Deactivate the object.

- std::streambuf::int_type [overflow](#) (std::streambuf::int_type c) override

Write a character in the case of overflow of the write buffer.

Private Attributes

- std::filebuf * [file_buf](#) {}

The file buffer that writes to the checkpoint file.

Friends

- class [SectionedOutputStream](#)

8.17.1 Detailed Description

A [SectionedOutputBuffer](#) is a `std::streambuf` that writes a section of a checkpoint file.

This is a barebones implementation. It does not provide buffering, and it does not support seek and tell.

Note that with the exception of the destructor and the inherited members from `std::streambuf`, *everything* in this class is private. This class is not extensible.

Definition at line 84 of file `checkpoint_output_manager.hh`.

8.17.2 Constructor & Destructor Documentation

8.17.2.1 `~SectionedOutputBuffer()`

```
jeod::SectionedOutputBuffer::~~SectionedOutputBuffer ( ) [override], [default]
```

Destructor.

For now, this does nothing.

8.17.2.2 `SectionedOutputBuffer()` [1/3]

```
jeod::SectionedOutputBuffer::SectionedOutputBuffer (
    const SectionedOutputBuffer & ) [delete]
```

8.17.2.3 `SectionedOutputBuffer()` [2/3]

```
jeod::SectionedOutputBuffer::SectionedOutputBuffer ( ) [private]
```

Default constructor.

This constructor creates an empty [SectionedOutputBuffer](#) – one that will return EOF on the first write attempt. An empty [SectionedOutputBuffer](#) has two purposes:

- As the basis for a copy constructor of a containing stream, and
- As a graceful means of handling of erroneous conditions.

Definition at line 44 of file `checkpoint_output_manager.cc`.

8.17.2.4 SectionedOutputBuffer() [3/3]

```
jeod::SectionedOutputBuffer::SectionedOutputBuffer (
    std::ofstream * stream ) [explicit], [private]
```

8.17.3 Member Function Documentation

8.17.3.1 activate()

```
void jeod::SectionedOutputBuffer::activate (
    std::ofstream & stream ) [private]
```

Activate the object.

Note

Using the object for writing prior to activation will result in EOF.

Parameters

in	<i>stream</i>	Output file stream
----	---------------	--------------------

Definition at line 56 of file checkpoint_output_manager.cc.

References file_buf.

Referenced by jeod::SectionedOutputStream::activate().

8.17.3.2 deactivate()

```
void jeod::SectionedOutputBuffer::deactivate ( ) [inline], [private]
```

Deactivate the object.

Used to disconnect the buffer when the stream is done, sometimes by force.

Definition at line 123 of file checkpoint_output_manager.hh.

References file_buf.

Referenced by jeod::SectionedOutputStream::deactivate().

8.17.3.3 operator!()

```
bool jeod::SectionedOutputBuffer::operator! ( ) const [inline]
```

Conversion to boolean.

Returns

False if object is OK.

Definition at line 103 of file checkpoint_output_manager.hh.

References file_buf.

8.17.3.4 operator=()

```
SectionedOutputBuffer& jeod::SectionedOutputBuffer::operator= (
    const SectionedOutputBuffer & ) [delete]
```

8.17.3.5 overflow()

```
std::streambuf::int_type jeod::SectionedOutputBuffer::overflow (
    std::streambuf::int_type ch ) [override], [private]
```

Write a character in the case of overflow of the write buffer.

For now, the buffer always overflows.

Returns

Status: EOF => failed

Parameters

in	ch	Character to be written
----	----	-------------------------

Definition at line 68 of file checkpoint_output_manager.cc.

References file_buf.

8.17.4 Friends And Related Function Documentation

8.17.4.1 SectionedOutputStream

```
friend class SectionedOutputStream [friend]
```

Definition at line 86 of file checkpoint_output_manager.hh.

8.17.5 Field Documentation

8.17.5.1 file_buf

```
std::filebuf* jeod::SectionedOutputBuffer::file_buf {} [private]
```

The file buffer that writes to the checkpoint file.

```
trick_io(**)
```

Definition at line 140 of file checkpoint_output_manager.hh.

Referenced by activate(), deactivate(), operator!(), and overflow().

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

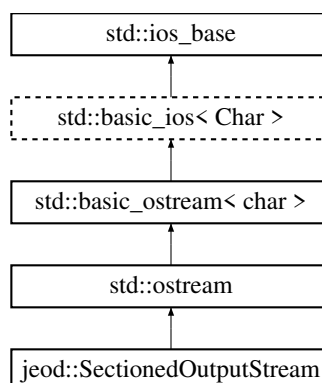
- [checkpoint_output_manager.hh](#)
- [checkpoint_output_manager.cc](#)

8.18 jeod::SectionedOutputStream Class Reference

A [SectionedOutputStream](#) is a `std::ostream` that writes a section of a checkpoint file.

```
#include <checkpoint_output_manager.hh>
```

Inheritance diagram for `jeod::SectionedOutputStream`:



Public Member Functions

- [SectionedOutputStream](#) ()
Construct a [SectionedOutputStream](#) object.
- [SectionedOutputStream](#) (const [SectionedOutputStream](#) &)
Construct a [SectionedOutputStream](#) object by copying from another.
- [~SectionedOutputStream](#) () override
Destruct a [SectionedOutputStream](#) object.
- [SectionedOutputStream](#) & operator= (const [SectionedOutputStream](#) &)=delete
- bool [is_activatable](#) () const
Determine if the stream is able to be activated.
- bool [activate](#) ()
Activate the object.
- void [deactivate](#) ()
Deactivate the object.
- bool [operator!](#) () const

Conversion to boolean.
- [operator void *](#) () const

Conversion to void.*

Private Member Functions

- [SectionedOutputStream](#) ([CheckPointOutputManager](#) *mngr, std::ofstream &ofstream, const std::string &start_marker, const std::string &end_marker, const std::string §ion_name)
Construct a [SectionedOutputStream](#) object that is connected to a file stream and to a [CheckPointOutputManager](#).

Private Attributes

- [SectionedOutputBuffer](#) sectbuf
The std::streambuf that does the writing to the file.
- [CheckPointOutputManager](#) * [manager](#) {}
The input manager that created this object.
- std::ofstream * [stream](#) {}
The C++ file stream that writes to the checkpoint file.
- const std::string * [section_start](#) {}
The string that indicates the start of a checkpoint file section.
- const std::string * [section_end](#) {}
The string that indicates the start of a checkpoint file section.
- const std::string [tag](#) {}
The name of the checkpoint file section.
- bool [is_copy](#) {}
Is this a copy of some other [SectionedOutputStream](#)? Copies of copies are verboten.
- bool [is_active](#) {}
Is this an active object? In the end, there can be only one.

Friends

- class [CheckPointOutputManager](#)

8.18.1 Detailed Description

A [SectionedOutputStream](#) is a `std::ostream` that writes a section of a checkpoint file.

This class automatically writes the start and end markers. Standard C++ output mechanisms can be used to write the contents of the section.

This is a barebones implementation. It does not provide buffering, it does not support seek and tell, and it does not support putback or unget.

Note that most of the content of this class is private. This class is not extensible and is intended to be used within the context of a [CheckPointOutputManager](#).

Definition at line 156 of file `checkpoint_output_manager.hh`.

8.18.2 Constructor & Destructor Documentation

8.18.2.1 SectionedOutputStream() [1/3]

```
jeod::SectionedOutputStream::SectionedOutputStream ( )
```

Construct a [SectionedOutputStream](#) object.

Note

This default constructor creates a disconnected and hence unusable stream. Usable streams are created by the non-default constructor.

Definition at line 113 of file `checkpoint_output_manager.cc`.

8.18.2.2 SectionedOutputStream() [2/3]

```
jeod::SectionedOutputStream::SectionedOutputStream (
    const SectionedOutputStream & source )
```

Construct a [SectionedOutputStream](#) object by copying from another.

Parameters

in	<i>source</i>	Source object
----	---------------	---------------

Definition at line 145 of file `checkpoint_output_manager.cc`.

References `jeod::SimInterfaceMessages::implementation_error`, `is_active`, `is_copy`, `manager`, and `stream`.

8.18.2.3 ~SectionedOutputStream()

```
jeod::SectionedOutputStream::~~SectionedOutputStream ( ) [override]
```

Destruct a [SectionedOutputStream](#) object.

Definition at line 171 of file `checkpoint_output_manager.cc`.

References `deactivate()`.

8.18.2.4 SectionedOutputStream() [3/3]

```
jeod::SectionedOutputStream::SectionedOutputStream (
    CheckPointOutputManager * mngr,
    std::ofstream & ostream,
    const std::string & start_marker,
    const std::string & end_marker,
    const std::string & section_name ) [private]
```

Construct a [SectionedOutputStream](#) object that is connected to a file stream and to a [CheckPointOutputManager](#).

Parameters

in	<i>mngr</i>	The stream manager
in	<i>ostream</i>	The output file stream
in	<i>start_marker</i>	Start of section marker
in	<i>end_marker</i>	End of section marker
in	<i>section_name</i>	Name of the section

Definition at line 127 of file `checkpoint_output_manager.cc`.

8.18.3 Member Function Documentation

8.18.3.1 activate()

```
bool jeod::SectionedOutputStream::activate ( )
```

Activate the object.

Note

Using the object for writing prior to activation will write nothing.

Returns

True if activated.

Definition at line 203 of file checkpoint_output_manager.cc.

References jeod::SectionedOutputBuffer::activate(), jeod::SimInterfaceMessages::implementation_error, is_active, manager, jeod::CheckPointOutputManager::register_writer(), sectbuf, section_start, stream, and tag.

Referenced by jeod::JeodTrick10MemoryInterface::checkpoint_allocations(), and jeod::JeodTrick10MemoryInterface::checkpoint_containers().

8.18.3.2 deactivate()

```
void jeod::SectionedOutputStream::deactivate ( )
```

Deactivate the object.

Note

Deactivation is undoable.

Definition at line 260 of file checkpoint_output_manager.cc.

References jeod::SectionedOutputBuffer::deactivate(), jeod::CheckPointOutputManager::deregister_writer(), is_active, manager, sectbuf, section_end, stream, and tag.

Referenced by jeod::JeodTrick10MemoryInterface::checkpoint_containers(), jeod::CheckPointOutputManager::create_trick_section_writer(), and ~SectionedOutputStream().

8.18.3.3 is_activatable()

```
bool jeod::SectionedOutputStream::is_activatable ( ) const
```

Determine if the stream is able to be activated.

Returns

True if object can be activated.

Definition at line 181 of file checkpoint_output_manager.cc.

References jeod::CheckPointOutputManager::have_active_writer(), is_active, manager, and stream.

8.18.3.4 operator void *()

```
jeod::SectionedOutputStream::operator void * ( ) const [inline]
```

Conversion to void*.

This method provides an alternative to the bang-bang trick to determine if the object is OK.

Returns

this pointer (cast to void*) if object is OK, NULL otherwise.

Definition at line 197 of file checkpoint_output_manager.hh.

8.18.3.5 operator!()

```
bool jeod::SectionedOutputStream::operator! ( ) const [inline]
```

Conversion to boolean.

Returns

False if object is OK.

Definition at line 186 of file checkpoint_output_manager.hh.

References `is_active`, `sectbuf`, and `stream`.

8.18.3.6 operator=()

```
SectionedOutputStream& jeod::SectionedOutputStream::operator= (
    const SectionedOutputStream & ) [delete]
```

8.18.4 Friends And Related Function Documentation

8.18.4.1 CheckPointOutputManager

```
friend class CheckPointOutputManager [friend]
```

Definition at line 158 of file checkpoint_output_manager.hh.

8.18.5 Field Documentation

8.18.5.1 `is_active`

```
bool jeod::SectionedOutputStream::is_active {} [private]
```

Is this an active object? In the end, there can be only one.

`trick_io(**)`

Definition at line 260 of file `checkpoint_output_manager.hh`.

Referenced by `activate()`, `deactivate()`, `is_activatable()`, `operator!()`, and `SectionedOutputStream()`.

8.18.5.2 `is_copy`

```
bool jeod::SectionedOutputStream::is_copy {} [private]
```

Is this a copy of some other [SectionedOutputStream](#)? Copies of copies are verboten.

`trick_io(**)`

Definition at line 254 of file `checkpoint_output_manager.hh`.

Referenced by `SectionedOutputStream()`.

8.18.5.3 `manager`

```
CheckpointOutputManager\* jeod::SectionedOutputStream::manager {} [private]
```

The input manager that created this object.

`trick_io(**)`

Definition at line 228 of file `checkpoint_output_manager.hh`.

Referenced by `activate()`, `deactivate()`, `is_activatable()`, and `SectionedOutputStream()`.

8.18.5.4 sectbuf

```
SectionedOutputBuffer jeod::SectionedOutputStream::sectbuf [private]
```

The std::streambuf that does the writing to the file.

trick_io(**)

Definition at line 223 of file checkpoint_output_manager.hh.

Referenced by activate(), deactivate(), and operator!().

8.18.5.5 section_end

```
const std::string* jeod::SectionedOutputStream::section_end {} [private]
```

The string that indicates the start of a checkpoint file section.

Definition at line 243 of file checkpoint_output_manager.hh.

Referenced by deactivate().

8.18.5.6 section_start

```
const std::string* jeod::SectionedOutputStream::section_start {} [private]
```

The string that indicates the start of a checkpoint file section.

Definition at line 238 of file checkpoint_output_manager.hh.

Referenced by activate().

8.18.5.7 stream

```
std::ofstream* jeod::SectionedOutputStream::stream {} [private]
```

The C++ file stream that writes to the checkpoint file.

trick_io(**)

Definition at line 233 of file checkpoint_output_manager.hh.

Referenced by activate(), deactivate(), is_activatable(), operator!(), and SectionedOutputStream().

8.18.5.8 tag

```
const std::string jeod::SectionedOutputStream::tag {""} [private]
```

The name of the checkpoint file section.

Definition at line 248 of file `checkpoint_output_manager.hh`.

Referenced by `activate()`, and `deactivate()`.

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

- [checkpoint_output_manager.hh](#)
- [checkpoint_output_manager.cc](#)

8.19 jeod::CheckPointInputManager::SectionInfo Struct Reference

A [SectionInfo](#) contains the start and end positions of a checkpoint file section.

Public Member Functions

- [SectionInfo](#) (std::size_t start, std::size_t end)
Non-default constructor.

Data Fields

- size_t [start_pos](#)
Position of the first readable character of a section.
- size_t [end_pos](#)
Position of the first unreadable character after a section.

8.19.1 Detailed Description

A [SectionInfo](#) contains the start and end positions of a checkpoint file section.

Definition at line 461 of file `checkpoint_input_manager.hh`.

8.19.2 Constructor & Destructor Documentation

8.19.2.1 SectionInfo()

```
jeod::CheckPointInputManager::SectionInfo::SectionInfo (
    std::size_t start,
    std::size_t end ) [inline]
```

Non-default constructor.

Parameters

in	<i>start</i>	Start position
in	<i>end</i>	End position

Definition at line 478 of file checkpoint_input_manager.hh.

8.19.3 Field Documentation

8.19.3.1 end_pos

```
size_t jeod::CheckpointInputManager::SectionInfo::end_pos
```

Position of the first unreadable character after a section.

```
trick_io(**)
```

Definition at line 471 of file checkpoint_input_manager.hh.

Referenced by jeod::CheckpointInputManager::create_section_reader().

8.19.3.2 start_pos

```
size_t jeod::CheckpointInputManager::SectionInfo::start_pos
```

Position of the first readable character of a section.

```
trick_io(**)
```

Definition at line 466 of file checkpoint_input_manager.hh.

Referenced by jeod::CheckpointInputManager::create_section_reader().

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

- [checkpoint_input_manager.hh](#)

8.20 jeod::SimInterfaceMessages Class Reference

Specifies the message IDs used in the sim_interface model.

```
#include <sim_interface_messages.hh>
```

Public Member Functions

- `SimInterfaceMessages()`=delete
- `SimInterfaceMessages(const SimInterfaceMessages &)=delete`
- `SimInterfaceMessages & operator=(const SimInterfaceMessages &)=delete`

Static Public Attributes

- static const char * `singleton_error` = "utils/sim_interface/" "singleton_error"
Message issued when multiple instance of a class that should be a singleton are created or when no such instance exists (but should).
- static const char * `interface_error` = "utils/sim_interface/" "interface_error"
Message issued when issues arise from interacting with the sim engine.
- static const char * `phasing_error` = "utils/sim_interface/" "phasing_error"
Message issued when things happen out of order.
- static const char * `integration_error` = "utils/sim_interface/" "integration_error"
Message issued when something goes awry with integration.
- static const char * `implementation_error` = "utils/sim_interface/" "implementation_error"
Message issued when something went wrong with the implementation.

8.20.1 Detailed Description

Specifies the message IDs used in the `sim_interface` model.

Definition at line 77 of file `sim_interface_messages.hh`.

8.20.2 Constructor & Destructor Documentation

8.20.2.1 SimInterfaceMessages() [1/2]

```
jeod::SimInterfaceMessages::SimInterfaceMessages ( ) [delete]
```

8.20.2.2 SimInterfaceMessages() [2/2]

```
jeod::SimInterfaceMessages::SimInterfaceMessages (
    const SimInterfaceMessages & ) [delete]
```

8.20.3 Member Function Documentation

8.20.3.1 operator=()

```
SimInterfaceMessages& jeod::SimInterfaceMessages::operator= (
    const SimInterfaceMessages & ) [delete]
```

8.20.4 Field Documentation

8.20.4.1 implementation_error

```
char const * jeod::SimInterfaceMessages::implementation_error = "utils/sim_interface/" "implementation↵
_error" [static]
```

Message issued when something went wrong with the implementation.

trick_units(—)

Definition at line 107 of file sim_interface_messages.hh.

Referenced by jeod::SectionedOutputStream::activate(), jeod::SectionedInputStream::activate(), jeod::Check↵
PointInputManager::CheckPointInputManager(), jeod::CheckPointOutputManager::CheckPointOutputManager(),
jeod::CheckPointInputManager::create_section_reader(), jeod::CheckPointOutputManager::create_section_↵
writer(), jeod::CheckPointInputManager::initialize(), jeod::JeodTrickMemoryInterface::JeodTrickMemoryInterface(),
jeod::SectionedInputStream::SectionedInputStream(), jeod::SectionedOutputStream::SectionedOutputStream(),
and jeod::JeodSimulationInterface::set_mode().

8.20.4.2 integration_error

```
char const * jeod::SimInterfaceMessages::integration_error = "utils/sim_interface/" "integration↵
_error" [static]
```

Message issued when something goes awry with integration.

trick_units(—)

Definition at line 102 of file sim_interface_messages.hh.

Referenced by jeod::JeodDynbodyIntegrationLoop::initialize_integ_loop(), jeod::JeodDynbodyIntegrationLoop↵
::integrate_dt(), jeod::JeodDynbodyIntegrationLoop::JeodDynbodyIntegrationLoop(), and jeod::JeodDynbody↵
IntegrationLoop::update_integration_group().

8.20.4.3 interface_error

```
char const * jeod::SimInterfaceMessages::interface_error = "utils/sim_interface/" "interface_↵
error" [static]
```

Message issued when issues arise from interacting with the sim engine.

trick_units(–)

Definition at line 92 of file sim_interface_messages.hh.

Referenced by jeod::JeodTrick10MemoryInterface::checkpoint_allocations(), jeod::JeodTrick10MemoryInterface↵
::checkpoint_containers(), jeod::JeodTrickMemoryInterface::deregister_allocation(), jeod::JeodTrick10Memory↵
Interface::deregister_container(), jeod::JeodTrickMemoryInterface::find_attributes(), jeod::JeodTrick10Memory↵
Interface::get_name_at_address(), jeod::JeodTrick10MemoryInterface::JeodTrick10MemoryInterface(), jeod::↵
JeodTrickMemoryInterface::primitive_attributes(), jeod::JeodTrickMemoryInterface::register_allocation(), jeod↵
::JeodTrick10MemoryInterface::register_container(), jeod::JeodTrick10MemoryInterface::restore_allocations(),
jeod::JeodTrick10MemoryInterface::restore_containers(), jeod::JeodTrick10MemoryInterface::translate_addr_↵
to_name(), and jeod::JeodTrick10MemoryInterface::translate_name_to_addr().

8.20.4.4 phasing_error

```
char const * jeod::SimInterfaceMessages::phasing_error = "utils/sim_interface/" "phasing_↵
error" [static]
```

Message issued when things happen out of order.

trick_units(–)

Definition at line 97 of file sim_interface_messages.hh.

Referenced by jeod::BasicJeodTrickSimInterface::get_checkpoint_reader_internal(), jeod::BasicJeodTrickSim↵
Interface::get_checkpoint_writer_internal(), and jeod::JeodSimulationInterface::set_mode().

8.20.4.5 singleton_error

```
char const * jeod::SimInterfaceMessages::singleton_error = "utils/sim_interface/" "singleton_↵
error" [static]
```

Message issued when multiple instance of a class that should be a singleton are created or when no such instance exists (but should).

trick_units(–)

Definition at line 87 of file sim_interface_messages.hh.

Referenced by jeod::JeodSimulationInterface::create_integrator_interface(), jeod::JeodSimulationInterface::get↵
_address_at_name(), jeod::JeodSimulationInterface::get_checkpoint_reader(), jeod::JeodSimulationInterface↵
::get_checkpoint_writer(), jeod::JeodSimulationInterface::get_job_cycle(), jeod::JeodSimulationInterface::get↵
memory_interface(), jeod::JeodSimulationInterface::get_name_at_address(), and jeod::JeodSimulationInterface↵
::JeodSimulationInterface().

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

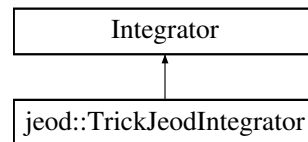
- [sim_interface_messages.hh](#)
- [sim_interface_messages.cc](#)

8.21 jeod::TrickJeodIntegrator Class Reference

A [TrickJeodIntegrator](#) specializes the `Trick::Integrator` to provide the [Trick](#) side of the integration interface between [Trick](#) and JEOD.

```
#include <jeod_trick_integrator.hh>
```

Inheritance diagram for `jeod::TrickJeodIntegrator`:



Public Member Functions

- [~TrickJeodIntegrator](#) () override=default
Destructor.
- int [integrate](#) () override
Does nothing.
- void [initialize](#) (int, double) override
Does nothing.

8.21.1 Detailed Description

A [TrickJeodIntegrator](#) specializes the `Trick::Integrator` to provide the [Trick](#) side of the integration interface between [Trick](#) and JEOD.

Definition at line 82 of file `jeod_trick_integrator.hh`.

8.21.2 Constructor & Destructor Documentation

8.21.2.1 ~TrickJeodIntegrator()

```
jeod::TrickJeodIntegrator::~~TrickJeodIntegrator ( ) [override], [default]
```

Destructor.

8.21.3 Member Function Documentation

8.21.3.1 initialize()

```
void jeod::TrickJeodIntegrator::initialize (
    int ,
    double ) [inline], [override]
```

Does nothing.

Definition at line 111 of file jeod_trick_integrator.hh.

8.21.3.2 integrate()

```
int jeod::TrickJeodIntegrator::integrate ( ) [inline], [override]
```

Does nothing.

Definition at line 103 of file jeod_trick_integrator.hh.

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

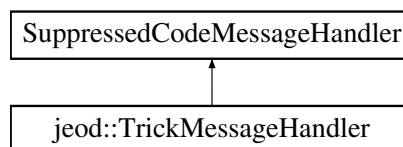
- [jeod_trick_integrator.hh](#)

8.22 jeod::TrickMessageHandler Class Reference

The MessageHandler class for designed for use in Trick-based simulations.

```
#include <trick_message_handler.hh>
```

Inheritance diagram for jeod::TrickMessageHandler:



Public Member Functions

- [TrickMessageHandler](#) ()=default
- [~TrickMessageHandler](#) () override=default
- [TrickMessageHandler](#) (const [TrickMessageHandler](#) &)=delete
- [TrickMessageHandler](#) & operator= (const [TrickMessageHandler](#) &)=delete
- void [register_contents](#) () override

Register the [TrickMessageHandler](#)'s checkpointable contents.

Protected Member Functions

- void [process_message](#) (int severity, const char *prefix, const char *file, unsigned int line, const char *msg↵_code, const char *format, va_list args) const override
Handle a message.

Friends

- class [InputProcessor](#)
- void [init_attrjeod__TrickMessageHandler](#) ()

8.22.1 Detailed Description

The MessageHandler class for designed for use in Trick-based simulations.

Definition at line 90 of file `trick_message_handler.hh`.

8.22.2 Constructor & Destructor Documentation

8.22.2.1 TrickMessageHandler() [1/2]

```
jeod::TrickMessageHandler::TrickMessageHandler ( ) [default]
```

8.22.2.2 ~TrickMessageHandler()

```
jeod::TrickMessageHandler::~~TrickMessageHandler ( ) [override], [default]
```

8.22.2.3 TrickMessageHandler() [2/2]

```
jeod::TrickMessageHandler::TrickMessageHandler (
    const TrickMessageHandler & ) [delete]
```

8.22.3 Member Function Documentation

8.22.3.1 operator=()

```
TrickMessageHandler& jeod::TrickMessageHandler::operator= (
    const TrickMessageHandler & ) [delete]
```

8.22.3.2 process_message()

```
void jeod::TrickMessageHandler::process_message (
    int severity,
    const char * prefix,
    const char * file,
    unsigned int line,
    const char * msg_code,
    const char * format,
    va_list args ) const [override], [protected]
```

Handle a message.

All calls to the message-generating MessageHandler methods eventually result in a call to thisTrickMessageHandler::process_message method. This method uses the [Trick](#) function exec_terminate to process fatal errors. The [Trick](#) function send_hs is used for all non-fatal messages, but only if the message severity is at or below the message suppression level.

Parameters

in	<i>severity</i>	Severity level
in	<i>prefix</i>	Message prefix (e.g., Error)
in	<i>file</i>	Typically FILE
in	<i>line</i>	Typically LINE
in	<i>msg_code</i>	Message code
in	<i>format</i>	sprintf format
in	<i>args</i>	Arguments

Definition at line 81 of file trick_message_handler.cc.

References MAX_MSG_SIZE.

8.22.3.3 register_contents()

```
void jeod::TrickMessageHandler::register_contents ( ) [override]
```

Register the [TrickMessageHandler](#)'s checkpointable contents.

Definition at line 60 of file trick_message_handler.cc.

8.22.4 Friends And Related Function Documentation

8.22.4.1 init_attrjeod__TrickMessageHandler

```
void init_attrjeod__TrickMessageHandler ( ) [friend]
```

8.22.4.2 InputProcessor

```
friend class InputProcessor [friend]
```

Definition at line 92 of file `trick_message_handler.hh`.

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

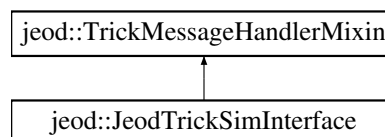
- [trick_message_handler.hh](#)
- [trick_message_handler.cc](#)

8.23 jeod::TrickMessageHandlerMixin Class Reference

The [TrickMessageHandlerMixin](#) implements the required capabilities of the generic [JeodSimulationInterface](#) in a [Trick](#) simulation environment.

```
#include <trick_sim_interface.hh>
```

Inheritance diagram for `jeod::TrickMessageHandlerMixin`:



Public Member Functions

- [TrickMessageHandlerMixin](#) ()=default
Default constructor.
- virtual [~TrickMessageHandlerMixin](#) ()=default
Destructor.
- [TrickMessageHandlerMixin](#) (const [TrickMessageHandlerMixin](#) &)=delete
- [TrickMessageHandlerMixin](#) & operator= (const [TrickMessageHandlerMixin](#) &)=delete

Protected Attributes

- [TrickMessageHandler](#) `message_handler`

The global MessageHandler.

Friends

- class [InputProcessor](#)
- void [init_attrjeod__TrickMessageHandlerMixin](#) ()

8.23.1 Detailed Description

The [TrickMessageHandlerMixin](#) implements the required capabilities of the generic [JeodSimulationInterface](#) in a [Trick](#) simulation environment.

By virtue of member data ownership, the class creates the requisite MessageHandler and MemoryManager and does so in the correct order.

Definition at line 226 of file `trick_sim_interface.hh`.

8.23.2 Constructor & Destructor Documentation

8.23.2.1 [TrickMessageHandlerMixin](#)() [1/2]

```
jeod::TrickMessageHandlerMixin::TrickMessageHandlerMixin ( ) [default]
```

Default constructor.

8.23.2.2 [~TrickMessageHandlerMixin](#)()

```
virtual jeod::TrickMessageHandlerMixin::~~TrickMessageHandlerMixin ( ) [virtual], [default]
```

Destructor.

8.23.2.3 [TrickMessageHandlerMixin](#)() [2/2]

```
jeod::TrickMessageHandlerMixin::TrickMessageHandlerMixin (
    const TrickMessageHandlerMixin & ) [delete]
```

8.23.3 Member Function Documentation

8.23.3.1 operator=()

```
TrickMessageHandlerMixin& jeod::TrickMessageHandlerMixin::operator= (
    const TrickMessageHandlerMixin & ) [delete]
```

8.23.4 Friends And Related Function Documentation

8.23.4.1 init_attrjeod__TrickMessageHandlerMixin

```
void init_attrjeod__TrickMessageHandlerMixin ( ) [friend]
```

8.23.4.2 InputProcessor

```
friend class InputProcessor [friend]
```

Definition at line 228 of file trick_sim_interface.hh.

8.23.5 Field Documentation

8.23.5.1 message_handler

```
TrickMessageHandler jeod::TrickMessageHandlerMixin::message_handler [protected]
```

The global MessageHandler.

trick_units(-)

Definition at line 250 of file trick_sim_interface.hh.

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

- [trick_sim_interface.hh](#)

Chapter 9

File Documentation

9.1 checkpoint_input_manager.cc File Reference

Define CheckPointInputManager member functions and of related classes.

```
#include <cstdint>
#include <cstring>
#include <iostream>
#include "utils/message/include/message_handler.hh"
#include "../include/checkpoint_input_manager.hh"
#include "../include/sim_interface_messages.hh"
```

Namespaces

- [jeod](#)

Namespace jeod.

9.1.1 Detailed Description

Define CheckPointInputManager member functions and of related classes.

9.2 checkpoint_input_manager.hh File Reference

Define class CheckPointInputManager and related classes.

```
#include "utils/sim_interface/include/config.hh"
#include "utils/sim_interface/include/jeod_class.hh"
#include <cstdint>
#include <fstream>
#include <istream>
#include <map>
#include <string>
```

Data Structures

- class [jeod::SectionedInputBuffer](#)

A [SectionedInputBuffer](#) is a `std::streambuf` that reads from a section in a checkpoint file.

- class [jeod::SectionedInputStream](#)

A [SectionedInputStream](#) is a `std::istream` that reads from a section in a checkpoint file.

- class [jeod::CheckpointInputManager](#)

A [CheckpointInputManager](#) provides tools for reading a checkpoint file.

- struct [jeod::CheckpointInputManager::SectionInfo](#)

A [SectionInfo](#) contains the start and end positions of a checkpoint file section.

Namespaces

- [jeod](#)

Namespace [jeod](#).

9.2.1 Detailed Description

Define class `CheckpointInputManager` and related classes.

9.3 checkpoint_output_manager.cc File Reference

Define `CheckpointOutputManager` member functions and of related classes.

```
#include <cstdint>
#include <cstring>
#include <iostream>
#include <utility>
#include "utils/message/include/message_handler.hh"
#include "../include/checkpoint_output_manager.hh"
#include "../include/sim_interface_messages.hh"
```

Namespaces

- [jeod](#)

Namespace [jeod](#).

9.3.1 Detailed Description

Define `CheckpointOutputManager` member functions and of related classes.

9.4 checkpoint_output_manager.hh File Reference

Define class CheckPointOutputManager and related classes.

```
#include "utils/sim_interface/include/jeod_class.hh"
#include <fstream>
#include <map>
#include <ostream>
#include <string>
```

Data Structures

- class [jeod::SectionedOutputBuffer](#)

A [SectionedOutputBuffer](#) is a `std::streambuf` that writes a section of a checkpoint file.

- class [jeod::SectionedOutputStream](#)

A [SectionedOutputStream](#) is a `std::ostream` that writes a section of a checkpoint file.

- class [jeod::CheckPointOutputManager](#)

A [CheckPointOutputManager](#) provides the basic tools for writing a checkpoint file.

Namespaces

- [jeod](#)

Namespace *jeod*.

9.4.1 Detailed Description

Define class CheckPointOutputManager and related classes.

9.5 class_declarations.hh File Reference

Forward declarations of classes defined in the utils/sim_interface model.

Namespaces

- [jeod](#)

Namespace *jeod*.

9.5.1 Detailed Description

Forward declarations of classes defined in the utils/sim_interface model.

9.6 config.hh File Reference

Configure JEOD for use by some simulation engine.

```
#include "config_trick10.hh"
```

Macros

- `#define JEOD_UNUSED`
- `#define ER7_UTILS_UNUSED`
- `#define ER7_UTILS_RESTRICT`
- `#define ER7_UTILS_ALWAYS_INLINE`

9.6.1 Detailed Description

Configure JEOD for use by some simulation engine.

9.7 config_test_harness.hh File Reference

Configure JEOD for use in standalone test mode.

Macros

- `#define JEOD_ATTRIBUTES_TYPE int`
- `#define JEOD_ATTRIBUTES_POINTER_TYPE void *`
- `#define JEOD_SIM_INTEGRATOR_POINTER_TYPE void *`

9.7.1 Detailed Description

Configure JEOD for use in standalone test mode.

9.8 config_trick10.hh File Reference

Configure JEOD for use in a Trick10 environment.

```
#include "jeod_va_macro_utility.hh"
```

Macros

- `#define JEOD_SIZE_T` `size_t`
- `#define JEOD_PTRDIFF_T` `long int`
- `#define JEOD_INTPTR_T` `long int`
- `#define JEOD_UINTPTR_T` `unsigned long int`
- `#define JEOD_CLASS_ESTABLISH_FRIENDS3`(ns1, ns2, class_name)
- `#define JEOD_CLASS_ESTABLISH_FRIENDS2`(ns, class_name)
- `#define JEOD_CLASS_ESTABLISH_FRIENDS1`(class_name)
- `#define JEOD_CLASS_ESTABLISH_FRIENDS(...)` `JEODVMACRO(JEOD_CLASS_ESTABLISH_FRIENDS1, __VA_ARGS__)`
- `#define JEOD_ATTRIBUTES_SIM_ENGINE_HEADER` `"sim_services/MemoryManager/include/attributes.h"`
- `#define JEOD_ATTRIBUTES_TYPE` `struct ATTRIBUTES_tag`
- `#define JEOD_ATTRIBUTES_POINTER_TYPE` `JEOD_ATTRIBUTES_TYPE *`
- `#define JEOD_SIM_INTEGRATOR_SIM_ENGINE_HEADER` `"sim_services/Integrator/include/Integrator.hh"`
- `#define JEOD_SIM_INTEGRATOR_FORWARD`
- `#define JEOD_SIM_INTEGRATOR_POINTER_TYPE` `Trick::Integrator *`
- `#define JEOD_SIM_INTEGRATOR_ENUM` `Integrator_type`

9.8.1 Detailed Description

Configure JEOD for use in a Trick10 environment.

9.9 jeod_class.hh File Reference

Define the JEOD class declaration macros `JEOD_MAKE_SIM_INTERFACES` and `JEOD_DECLARE_SIM_INTERFACES`.

```
#include "config.hh"
```

Macros

- `#define JEOD_MAKE_SIM_INTERFACES(...)` `JEOD_CLASS_ESTABLISH_FRIENDS(__VA_ARGS__); JEOD_MAKE_SIM_INTERFACES(class_name)` *Defines friends of the given class.*
- `#define JEOD_DECLARE_SIM_INTERFACES(class_name)` `JEOD_DECLARE_SIM_INTERFACES(class_name)` *Forward declare classes and external functions needed to make the `JEOD_MAKE_SIM_INTERFACES(class_name)` expansion compilable.*

9.9.1 Detailed Description

Define the JEOD class declaration macros `JEOD_MAKE_SIM_INTERFACES` and `JEOD_DECLARE_SIM_INTERFACES`.

All JEOD class definitions must invoke `JEOD_MAKE_SIM_INTERFACES` within the body of the class. Corresponding invocations of `JEOD_DECLARE_SIM_INTERFACES` Are made at file scope and in the context of the global namespace.

In a [Trick](#) environment, these macros gives the [Trick](#) input processor, the [Trick](#) checkpoint / checkpoint-restart facility, and the ICG-generated `io_src` file for the header full visibility of the class's contents. The intent is to provide the same capability outside the [Trick](#).

9.10 jeod_integrator_interface.hh File Reference

Define the interface for accessing / updating elements of a simulation engine's integrator object.

```
#include "utils/sim_interface/include/config.hh"
#include "utils/sim_interface/include/jeod_class.hh"
#include "er7_utils/integration/core/include/integration_technique.hh"
#include "er7_utils/integration/core/include/integrator_interface.hh"
```

Data Structures

- class [jeod::JeodIntegratorInterface](#)

A [JeodIntegratorInterface](#) extends the [ER7 IntegratorInterface](#) with the concept of a pointer to the simulation engine's integration object.

Namespaces

- [Trick](#)

Namespace [Trick](#) furnishes several standard functions for use in the [Trick](#) environment.

- [jeod](#)

Namespace [jeod](#).

9.10.1 Detailed Description

Define the interface for accessing / updating elements of a simulation engine's integrator object.

9.11 jeod_trick_integrator.hh File Reference

Define the interface for accessing / updating elements of a [Trick](#) simulation integrator object.

```
#include "sim_services/Integrator/include/Integrator.hh"
#include "er7_utils/trick/integration/include/translate_trick_integ_type.↵
hh"
#include "jeod_class.hh"
#include "jeod_integrator_interface.hh"
```

Data Structures

- class [jeod::TrickJeodIntegrator](#)

A [TrickJeodIntegrator](#) specializes the [Trick::Integrator](#) to provide the [Trick](#) side of the integration interface between [Trick](#) and [JEOD](#).

- class [jeod::JeodTrickIntegrator](#)

A [JeodTrickIntegrator](#) specializes the [JeodIntegratorInterface](#) for use with [Trick](#) as the simulation engine.

Namespaces

- [jeod](#)

Namespace jeod.

9.11.1 Detailed Description

Define the interface for accessing / updating elements of a [Trick](#) simulation integrator object.

9.12 jeod_va_macro_utility.hh File Reference

Support header for variable argument macro functions.

Macros

- #define [JEODVMACRO_VA_SIZE](#)(...) [JEODVMACRO_GET_COUNT](#)(__VA_ARGS__, [JEODVMACRO_REVSEQ_COUNT](#)())
- #define [JEODVMACRO_GET_COUNT](#)(...) [JEODVMACRO_SEQ_COUNT](#)(__VA_ARGS__)
- #define [JEODVMACRO_SEQ_COUNT](#)(_1, _2, _3, _4, _5, _6, _7, _8, _9, _10, _11, _12, _13, _14, _15, _16, _17, _18, _19, _20, _21, _22, _23, _24, _25, _26, _27, _28, _29, _30, _31, _32, _33, _34, _35, _36, _37, _38, _39, _40, _41, _42, _43, _44, _45, _46, _47, _48, _49, _50, _51, _52, _53, _54, _55, _56, _57, _58, _59, _60, _61, _62, _63, N, ...) N
- #define [JEODVMACRO_REVSEQ_COUNT](#)()
- #define [JEODVMACRO_SELECT_FUNC_CAT](#)(name, n) name##n
- #define [JEODVMACRO_SELECT_FUNC](#)(name, n) [JEODVMACRO_SELECT_FUNC_CAT](#)(name, n)
- #define [JEODVMACRO](#)(func, ...) [JEODVMACRO_SELECT_FUNC](#)(func, [JEODVMACRO_VA_SIZE](#)(__VA_ARGS__)(__VA_ARGS__))

9.12.1 Detailed Description

Support header for variable argument macro functions.

9.13 memory_attributes.hh File Reference

Define JEOD memory interface macros.

```
#include "config.hh"
#include "sim_services/MemoryManager/include/attributes.h"
```

Namespaces

- [jeod](#)

Namespace jeod.

Macros

- `#define JEOD_DECLARE_ATTRIBUTES(class_name)`
`JEOD_DECLARE_ATTRIBUTES(class_name)` *This macro is obsolete.*
- `#define JEOD_ATTRIBUTES(type) JeodSimulationInterface::get_memory_interface().find_attributes(#type)`
Get a pointer to or construct the name of the attributes for the type.

9.13.1 Detailed Description

Define JEOD memory interface macros.

- Most of the memory interface between JEOD and the simulation engine is handled by the JeodMemory↔Interface.
- The macros defined in this file represent the functionality that cannot be solved using c++ classes.
- The macros prefixed with JEOD_DECLARE are used in model files that use the memory model to allocate memory.
- The remaining macros are used internally by the JEOD memory model and should not be used in model files.

9.13.2 Macro Definition Documentation

9.13.2.1 JEOD_ATTRIBUTES

```
#define JEOD_ATTRIBUTES(  
    type ) JeodSimulationInterface::get_memory_interface().find_attributes(#type)
```

Get a pointer to or construct the name of the attributes for the type.

Note

This is a primitive macro. Do not use it in model files.

Parameters

<i>type</i>	Data type.
-------------	------------

Returns

Pointer to or symbolic name of the attributes for the type.

Definition at line 108 of file memory_attributes.hh.

9.13.2.2 JEOD_DECLARE_ATTRIBUTES

```
#define JEOD_DECLARE_ATTRIBUTES(  
    class_name )
```

[JEOD_DECLARE_ATTRIBUTES\(class_name\)](#) This macro is obsolete.

Definition at line 99 of file memory_attributes.hh.

9.14 memory_interface.hh File Reference

Define the MemoryInterface class, which abstractly defines the interface between the memory manager and the simulation engine.

```
#include <cstddef>  
#include <string>  
#include <typeinfo>  
#include "utils/sim_interface/include/jeod_class.hh"  
#include "memory_attributes.hh"
```

Data Structures

- class [jeod::JeodMemoryInterface](#)

Abstract interface between the JEOD memory manager and the simulation engine.

Namespaces

- [jeod](#)

Namespace jeod.

9.14.1 Detailed Description

Define the MemoryInterface class, which abstractly defines the interface between the memory manager and the simulation engine.

9.15 sim_interface_messages.cc File Reference

Implement the class SimInterfaceMessages.

```
#include "../include/sim_interface_messages.hh"
```

Namespaces

- [jeod](#)

Namespace jeod.

Macros

- `#define PATH "utils/sim_interface/"`
- `#define CLASS SimInterfaceMessages`
- `#define MAKE_MESSAGE_CODE(id) char const * CLASS::id = PATH #id`

9.15.1 Detailed Description

Implement the class `SimInterfaceMessages`.

9.16 `sim_interface_messages.hh` File Reference

Define the class `SimInterfaceMessages`, the class that specifies the message IDs used in the sim interface model.

```
#include "jeod_class.hh"
```

Data Structures

- class `jeod::SimInterfaceMessages`
Specifies the message IDs used in the sim_interface model.

Namespaces

- `jeod`
Namespace jeod.

9.16.1 Detailed Description

Define the class `SimInterfaceMessages`, the class that specifies the message IDs used in the sim interface model.

9.17 `simulation_interface.cc` File Reference

Implement `SimulationInterface` methods.

```
#include <cstdint>
#include "utils/memory/include/memory_manager.hh"
#include "utils/message/include/message_handler.hh"
#include "../include/sim_interface_messages.hh"
#include "../include/simulation_interface.hh"
```

Namespaces

- [jeod](#)
Namespace jeod.

9.17.1 Detailed Description

Implement SimulationInterface methods.

9.18 simulation_interface.hh File Reference

Define the abstract class JeodSimulationInterface.

```
#include <string>
#include "checkpoint_input_manager.hh"
#include "checkpoint_output_manager.hh"
#include "class_declarations.hh"
#include "jeod_class.hh"
#include "jeod_integrator_interface.hh"
```

Data Structures

- class [jeod::JeodSimulationInterfaceInit](#)
Define configuration data needed to configure the dynamically-created message handler and memory manager.
- class [jeod::JeodSimulationInterface](#)
This abstract class defines the basis for the interface between JEOD and a simulation engine.

Namespaces

- [jeod](#)
Namespace jeod.

9.18.1 Detailed Description

Define the abstract class JeodSimulationInterface.

9.19 trick10_memory_interface.cc File Reference

Define JeodTrickMemoryInterface methods.

```
#include <cstddef>
#include <dlfcn.h>
#include <iomanip>
#include <sstream>
#include <iosfwd>
#include "sim_services/CheckPointAgent/include/ClassicCheckPointAgent.hh"
#include "sim_services/MemoryManager/include/MemoryManager.hh"
#include "utils/message/include/message_handler.hh"
#include "../include/sim_interface_messages.hh"
#include "../include/simulation_interface.hh"
#include "../include/trick10_memory_interface.hh"
```

Namespaces

- [jeod](#)

Namespace jeod.

Variables

- Trick::MemoryManager * [trick_MM](#)

9.19.1 Detailed Description

Define JeodTrickMemoryInterface methods.

9.20 trick10_memory_interface.hh File Reference

Define the interface for registering / deregistering memory with [Trick](#).

```
#include <cstddef>
#include <cstdint>
#include <cstring>
#include <list>
#include <map>
#include <string>
#include "jeod_class.hh"
#include "memory_attributes.hh"
#include "memory_interface.hh"
#include "simulation_interface.hh"
#include "trick_memory_interface.hh"
```

Data Structures

- class [jeod::JeodTrick10MemoryInterface](#)

A TrickMemoryInterface implements the two required methods needed to register and deregister memory with the simulation engine, [Trick](#) in this case.

Namespaces

- [jeod](#)

Namespace jeod.

- [Trick](#)

Namespace [Trick](#) furnishes several standard functions for use in the [Trick](#) environment.

9.20.1 Detailed Description

Define the interface for registering / deregistering memory with [Trick](#).

9.21 trick_dynbody_integ_loop.cc File Reference

Define JeodDynbodyIntegrationLoop methods.

```
#include "../include/trick_dynbody_integ_loop.hh"
#include "../include/sim_interface_messages.hh"
#include "dynamics/dyn_body/include/dyn_body.hh"
#include "dynamics/dyn_manager/include/dyn_manager.hh"
#include "environment/time/include/time_manager.hh"
#include "utils/message/include/message_handler.hh"
#include "sim_services/Executive/include/exec_proto.h"
#include <cstdlib>
#include <string>
#include <vector>
```

Namespaces

- [jeod](#)
Namespace jeod.

Variables

- Trick::Integrator * [trick_curr_integ](#)

9.21.1 Detailed Description

Define JeodDynbodyIntegrationLoop methods.

9.22 trick_dynbody_integ_loop.hh File Reference

Define the class IntegrationGroupIntegLoopScheduler, which replaces the base [Trick](#) integration loop for multi-rate JEOD-based simulations.

```
#include "jeod_trick_integrator.hh"
#include "dynamics/dyn_manager/include/dynamics_integration_group.hh"
#include "utils/integration/include/jeod_integration_group.hh"
#include "utils/sim_interface/include/jeod_class.hh"
#include "sim_services/Integrator/include/IntegLoopScheduler.hh"
```

Data Structures

- class [jeod::JeodDynbodyIntegrationLoop](#)
A Trick::IntegLoopScheduler that provides the ability to integrate a collection of Trick::SimObject instances over time, with the sim objects capable of being moved from one integration loop to another during run time.

Namespaces

- [jeod](#)
Namespace [jeod](#).
- [Trick](#)
Namespace [Trick](#) furnishes several standard functions for use in the [Trick](#) environment.
- [er7_utils](#)
Namespace [er7_utils](#) contains the state integration models used by JEOD.

9.22.1 Detailed Description

Define the class `IntegrationGroupIntegLoopScheduler`, which replaces the base [Trick](#) integration loop for multi-rate JEOD-based simulations.

9.23 `trick_memory_interface.cc` File Reference

Define `JeodTrickMemoryInterface` methods.

```
#include <cstddef>
#include <dlfcn.h>
#include <iomanip>
#include <sstream>
#include "utils/message/include/message_handler.hh"
#include "../include/sim_interface_messages.hh"
#include "../include/simulation_interface.hh"
#include "../include/trick_memory_interface.hh"
```

Namespaces

- [jeod](#)
Namespace [jeod](#).

9.23.1 Detailed Description

Define `JeodTrickMemoryInterface` methods.

9.24 `trick_memory_interface.hh` File Reference

Define the interface for registering / deregistering memory with [Trick](#).

```
#include <cstddef>
#include <stdint>
#include <cstring>
#include <list>
#include <map>
#include <string>
#include <utility>
#include "jeod_class.hh"
#include "memory_attributes.hh"
#include "memory_interface.hh"
#include "simulation_interface.hh"
```

Data Structures

- class [jeod::JeodTrickMemoryInterface](#)
A TrickMemoryInterface implements the two required methods needed to register and deregister memory with the simulation engine, [Trick](#) in this case.
- struct [jeod::JeodTrickMemoryInterface::ContainerListEntry](#)
Describes a Checkpointable object.
- struct [jeod::JeodTrickMemoryInterface::AllocationMapEntry](#)
Describes a chunk of JEOD-allocated memory.

Namespaces

- [jeod](#)
Namespace jeod.

9.24.1 Detailed Description

Define the interface for registering / deregistering memory with [Trick](#).

9.25 trick_memory_interface_alloc.cc File Reference

Define JeodTrickMemoryInterface methods related to allocation/deallocation.

```
#include <cstdint>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <dlfcn.h>
#include <iomanip>
#include <sstream>
#include <typeinfo>
#include "sim_services/MemoryManager/include/ADefParseContext.hh"
#include "sim_services/MemoryManager/include/MemoryManager.hh"
#include "sim_services/MemoryManager/include/attributes.h"
#include "utils/memory/include/memory_item.hh"
#include "utils/memory/include/memory_manager.hh"
#include "utils/memory/include/memory_type.hh"
#include "utils/message/include/message_handler.hh"
#include "../include/sim_interface_messages.hh"
#include "../include/simulation_interface.hh"
#include "../include/trick_memory_interface.hh"
```

Namespaces

- [jeod](#)
Namespace jeod.

Variables

- Trick::MemoryManager * [trick_MM](#)

9.25.1 Detailed Description

Define JeodTrickMemoryInterface methods related to allocation/deallocation.

9.26 `trick_memory_interface_attrb.cc` File Reference

Define JeodTrickMemoryInterface methods related to attributes.

```
#include <cstdint>
#include <cstring>
#include <dlfcn.h>
#include "sim_services/MemoryManager/include/attributes.h"
#include "utils/memory/include/memory_type.hh"
#include "utils/message/include/message_handler.hh"
#include "utils/named_item/include/named_item.hh"
#include "../include/sim_interface_messages.hh"
#include "../include/trick_memory_interface.hh"
```

Namespaces

- [jeod](#)
Namespace jeod.

9.26.1 Detailed Description

Define JeodTrickMemoryInterface methods related to attributes.

9.27 `trick_memory_interface_chkpnt.cc` File Reference

Define JeodTrick10MemoryInterface methods related to checkpoint/restart.

```
#include <cstdint>
#include <stdio>
#include <stdlib>
#include <cstring>
#include <dlfcn.h>
#include <iomanip>
#include <sstream>
#include <typeinfo>
#include "sim_services/MemoryManager/include/ADefParseContext.hh"
#include "sim_services/MemoryManager/include/MemoryManager.hh"
#include "sim_services/MemoryManager/include/attributes.h"
#include "utils/container/include/checkpointable.hh"
#include "utils/memory/include/memory_item.hh"
#include "utils/memory/include/memory_manager.hh"
#include "utils/memory/include/memory_type.hh"
#include "utils/message/include/message_handler.hh"
#include "../include/sim_interface_messages.hh"
#include "../include/simulation_interface.hh"
#include "../include/trick10_memory_interface.hh"
```


Namespaces

- [jeod](#)

Namespace jeod.

Variables

- Trick::MemoryManager * [trick_MM](#)

9.27.1 Detailed Description

Define JeodTrick10MemoryInterface methods related to checkpoint/restart.

9.28 trick_memory_interface_xlate.cc File Reference

Define JeodTrickMemoryInterface methods related to name translation.

```
#include <cstddef>
#include <cstdlib>
#include <string>
#include <iosfwd>
#include "sim_services/CheckPointAgent/include/ClassicCheckPointAgent.hh"
#include "sim_services/MemoryManager/include/MemoryManager.hh"
#include "sim_services/MemoryManager/include/attributes.h"
#include "sim_services/MemoryManager/include/memorymanager_c_intf.h"
#include "sim_services/CheckPointRestart/include/CheckPointRestart_c_intf.↵
hh"
#include "utils/memory/include/memory_type.hh"
#include "../include/sim_interface_messages.hh"
#include "../include/simulation_interface.hh"
#include "../include/trick10_memory_interface.hh"
```

Namespaces

- [jeod](#)

Namespace jeod.

Variables

- Trick::MemoryManager * [trick_MM](#)

9.28.1 Detailed Description

Define JeodTrickMemoryInterface methods related to name translation.

9.29 trick_message_handler.cc File Reference

Define member functions for the class TrickMessageHandler.

```
#include <cstdlib>
#include <cstdio>
#include "sim_services/Executive/include/exec_proto.h"
#include "sim_services/Message/include/message_proto.h"
#include "utils/memory/include/jeod_alloc.hh"
#include "../include/trick_message_handler.hh"
```

Namespaces

- [jeod](#)
Namespace jeod.

Macros

- `#define` [MAX_MSG_SIZE](#) 4096

9.29.1 Detailed Description

Define member functions for the class TrickMessageHandler.

9.30 trick_message_handler.hh File Reference

Define the class TrickMessageHandler, the message handler designed for use in Trick-based simulations.

```
#include <cstdlib>
#include <string>
#include "utils/container/include/primitive_set.hh"
#include "utils/message/include/suppressed_code_message_handler.hh"
#include "utils/sim_interface/include/jeod_class.hh"
#include "class_declarations.hh"
```

Data Structures

- class [jeod::TrickMessageHandler](#)
The MessageHandler class for designed for use in Trick-based simulations.

Namespaces

- [jeod](#)
Namespace jeod.

9.30.1 Detailed Description

Define the class TrickMessageHandler, the message handler designed for use in Trick-based simulations.

9.31 trick_sim_interface.cc File Reference

Implement TrickSimInterface methods.

```
#include "sim_services/CommandLineArguments/include/command_line_protos.h"
#include "sim_services/Executive/include/exec_proto.h"
#include "sim_services/Message/include/message_proto.h"
#include "utils/memory/include/jeod_alloc.hh"
#include "utils/message/include/message_handler.hh"
#include "../include/jeod_trick_integrator.hh"
#include "../include/sim_interface_messages.hh"
#include "../include/trick_sim_interface.hh"
#include "../include/checkpoint_input_manager.hh"
#include "../include/checkpoint_output_manager.hh"
```

Namespaces

- [jeod](#)
Namespace *jeod*.

9.31.1 Detailed Description

Implement TrickSimInterface methods.

9.32 trick_sim_interface.hh File Reference

Define the class JeodTrickSimInterface.

```
#include "utils/memory/include/memory_manager.hh"
#include "jeod_class.hh"
#include "simulation_interface.hh"
#include "trick10_memory_interface.hh"
#include "trick_memory_interface.hh"
#include "trick_message_handler.hh"
#include "utils/sim_interface/include/jeod_trick_integrator.hh"
```

Data Structures

- class [jeod::BasicJeodTrickSimInterface](#)
The BasicJeodTrickSimInterface implements the required capabilities of the generic JeodSimulationInterface in a Trick simulation environment.
- class [jeod::TrickMessageHandlerMixin](#)
The TrickMessageHandlerMixin implements the required capabilities of the generic JeodSimulationInterface in a Trick simulation environment.
- class [jeod::JeodTrickSimInterface](#)
A JEODTrickSimInterface implements the required capabilities of the generic JeodSimulationInterface in a Trick simulation environment.

Namespaces

- [jeod](#)

Namespace jeod.

9.32.1 Detailed Description

Define the class JeodTrickSimInterface.

Index

- ~BasicJeodTrickSimInterface
 - jeod::BasicJeodTrickSimInterface, 31
- ~JeodDynbodyIntegrationLoop
 - jeod::JeodDynbodyIntegrationLoop, 59
- ~JeodIntegratorInterface
 - jeod::JeodIntegratorInterface, 70
- ~JeodMemoryInterface
 - jeod::JeodMemoryInterface, 73
- ~JeodSimulationInterface
 - jeod::JeodSimulationInterface, 82
- ~JeodTrick10MemoryInterface
 - jeod::JeodTrick10MemoryInterface, 93
- ~JeodTrickIntegrator
 - jeod::JeodTrickIntegrator, 103
- ~JeodTrickMemoryInterface
 - jeod::JeodTrickMemoryInterface, 109
- ~JeodTrickSimInterface
 - jeod::JeodTrickSimInterface, 122
- ~SectionedInputBuffer
 - jeod::SectionedInputBuffer, 124
- ~SectionedInputStream
 - jeod::SectionedInputStream, 133
- ~SectionedOutputBuffer
 - jeod::SectionedOutputBuffer, 139
- ~SectionedOutputStream
 - jeod::SectionedOutputStream, 144
- ~TrickJeodIntegrator
 - jeod::TrickJeodIntegrator, 155
- ~TrickMessageHandler
 - jeod::TrickMessageHandler, 157
- ~TrickMessageHandlerMixin
 - jeod::TrickMessageHandlerMixin, 160
- activate
 - jeod::SectionedInputBuffer, 125
 - jeod::SectionedInputStream, 133
 - jeod::SectionedOutputBuffer, 140
 - jeod::SectionedOutputStream, 145
- add_integrable_object
 - jeod::JeodDynbodyIntegrationLoop, 60
- add_sim_object
 - jeod::JeodDynbodyIntegrationLoop, 60
- add_sim_object_bodies
 - jeod::JeodDynbodyIntegrationLoop, 60, 62
- allocation_map
 - jeod::JeodTrickMemoryInterface, 119
- AllocationMap
 - jeod::JeodTrickMemoryInterface, 109
- AllocationMapEntry
 - jeod::JeodTrickMemoryInterface::AllocationMapEntry, 27
- at_eof
 - jeod::SectionedInputBuffer, 127
- BasicJeodTrickSimInterface
 - jeod::BasicJeodTrickSimInterface, 31
- buf
 - jeod::SectionedInputBuffer, 127
- CLASS
 - SimInterface, 15
- CheckPointInputManager
 - jeod::CheckPointInputManager, 41
 - jeod::SectionedInputStream, 135
- CheckPointOutputManager
 - jeod::CheckPointOutputManager, 48
 - jeod::SectionedOutputStream, 147
- checkpoint_allocations
 - jeod::BasicJeodTrickSimInterface, 31
 - jeod::JeodTrick10MemoryInterface, 94
 - jeod::JeodTrickMemoryInterface, 110
- checkpoint_containers
 - jeod::BasicJeodTrickSimInterface, 32
 - jeod::JeodTrick10MemoryInterface, 94
 - jeod::JeodTrickMemoryInterface, 110
- checkpoint_file_name
 - jeod::BasicJeodTrickSimInterface, 37
- checkpoint_input_manager.cc, 163
- checkpoint_input_manager.hh, 163
- checkpoint_output_manager.cc, 164
- checkpoint_output_manager.hh, 165
- checkpoint_reader
 - jeod::BasicJeodTrickSimInterface, 37
- checkpoint_writer
 - jeod::BasicJeodTrickSimInterface, 37
- class_declarations.hh, 165
- close_checkpoint_file
 - jeod::BasicJeodTrickSimInterface, 32
- close_restart_file
 - jeod::BasicJeodTrickSimInterface, 32
- collect_derivatives
 - jeod::JeodDynbodyIntegrationLoop, 62
- config.hh, 166
- config_test_harness.hh, 166
- config_trick10.hh, 166
- configure
 - jeod::JeodSimulationInterface, 82
- construct_identifier
 - jeod::JeodTrickMemoryInterface, 110

- container
 - jeod::JeodTrickMemoryInterface::ContainerList↔
Entry, 55
- container_list
 - jeod::JeodTrickMemoryInterface, 119
- ContainerList
 - jeod::JeodTrickMemoryInterface, 109
- ContainerListEntry
 - jeod::JeodTrickMemoryInterface::ContainerList↔
Entry, 54
- create_integrator_interface
 - jeod::JeodSimulationInterface, 83
- create_integrator_internal
 - jeod::BasicJeodTrickSimInterface, 32
 - jeod::JeodSimulationInterface, 83
- create_section_reader
 - jeod::CheckPointInputManager, 41, 42
- create_section_writer
 - jeod::CheckPointOutputManager, 48, 49
- create_trick_section_reader
 - jeod::CheckPointInputManager, 43
- create_trick_section_writer
 - jeod::CheckPointOutputManager, 49
- curr_pos
 - jeod::SectionedInputBuffer, 128
- current_reader
 - jeod::CheckPointInputManager, 45
- current_writer
 - jeod::CheckPointOutputManager, 52
- deactivate
 - jeod::SectionedInputBuffer, 126
 - jeod::SectionedInputStream, 134
 - jeod::SectionedOutputBuffer, 140
 - jeod::SectionedOutputStream, 146
- default_first_step_deriv
 - jeod::JeodTrickIntegrator, 106
- deregister_allocation
 - jeod::JeodMemoryInterface, 73
 - jeod::JeodTrickMemoryInterface, 111
- deregister_container
 - jeod::JeodMemoryInterface, 73
 - jeod::JeodTrick10MemoryInterface, 94
 - jeod::JeodTrickMemoryInterface, 111
- deregister_reader
 - jeod::CheckPointInputManager, 43
- deregister_writer
 - jeod::CheckPointOutputManager, 50
- deriv_ephem_update
 - jeod::JeodDynbodyIntegrationLoop, 67
- dlhandle
 - jeod::JeodTrickMemoryInterface, 119
- dyn_manager
 - jeod::JeodDynbodyIntegrationLoop, 67
- ER7_UTILS_ALWAYS_INLINE
 - SimInterface, 15
- ER7_UTILS_RESTRICT
 - SimInterface, 16
- ER7_UTILS_UNUSED
 - SimInterface, 16
- elem_name
 - jeod::JeodTrickMemoryInterface::ContainerList↔
Entry, 55
- end_pos
 - jeod::CheckPointInputManager::SectionInfo, 151
 - jeod::SectionedInputBuffer, 128
 - jeod::SectionedInputStream, 136
- er7_utils, 25
- file_buf
 - jeod::SectionedInputBuffer, 128
 - jeod::SectionedOutputBuffer, 142
- filename
 - jeod::CheckPointInputManager, 45
 - jeod::CheckPointOutputManager, 52
- find_attributes
 - jeod::JeodMemoryInterface, 74
 - jeod::JeodTrickMemoryInterface, 112
- find_containing_sim_object
 - jeod::JeodDynbodyIntegrationLoop, 62
- generic_message_handler
 - jeod::BasicJeodTrickSimInterface, 38
- get_address_at_name
 - jeod::JeodMemoryInterface, 75
 - jeod::JeodSimulationInterface, 83
 - jeod::JeodTrick10MemoryInterface, 95
 - jeod::JeodTrickMemoryInterface, 113
- get_checkpoint_file_name
 - jeod::BasicJeodTrickSimInterface, 33
- get_checkpoint_reader
 - jeod::JeodSimulationInterface, 84
- get_checkpoint_reader_internal
 - jeod::BasicJeodTrickSimInterface, 33
 - jeod::JeodSimulationInterface, 84
- get_checkpoint_writer
 - jeod::JeodSimulationInterface, 84
- get_checkpoint_writer_internal
 - jeod::BasicJeodTrickSimInterface, 33
 - jeod::JeodSimulationInterface, 85
- get_container_id
 - jeod::JeodTrick10MemoryInterface, 96
- get_dt
 - jeod::JeodTrickIntegrator, 103
- get_first_step_derivs_flag
 - jeod::JeodTrickIntegrator, 103
- get_integrator
 - jeod::JeodIntegratorInterface, 70
 - jeod::JeodTrickIntegrator, 103
- get_job_cycle
 - jeod::JeodSimulationInterface, 85
- get_job_cycle_internal
 - jeod::BasicJeodTrickSimInterface, 34
 - jeod::JeodSimulationInterface, 85
- get_memory_interface
 - jeod::JeodSimulationInterface, 86
- get_memory_interface_internal

- jeod::BasicJeodTrickSimInterface, 34
- jeod::JeodSimulationInterface, 86
- get_mode
 - jeod::JeodSimulationInterface, 86
- get_name_at_address
 - jeod::JeodMemoryInterface, 75
 - jeod::JeodSimulationInterface, 87
 - jeod::JeodTrick10MemoryInterface, 96
 - jeod::JeodTrickMemoryInterface, 113
- get_trick_checkpoint_file
 - jeod::JeodTrick10MemoryInterface, 97
 - jeod::JeodTrickMemoryInterface, 114
- gravitation
 - jeod::JeodDynbodyIntegrationLoop, 63
- gravity_manager
 - jeod::JeodDynbodyIntegrationLoop, 67
- have_active_reader
 - jeod::CheckPointInputManager, 43
- have_active_writer
 - jeod::CheckPointOutputManager, 50
- id_length
 - jeod::JeodTrickMemoryInterface, 120
- id_prefix
 - jeod::JeodTrickMemoryInterface, 120
- implementation_error
 - jeod::SimInterfaceMessages, 153
- init_attrjeod__BasicJeodTrickSimInterface
 - jeod::BasicJeodTrickSimInterface, 36
- init_attrjeod__JeodDynbodyIntegrationLoop
 - jeod::JeodDynbodyIntegrationLoop, 66
- init_attrjeod__JeodIntegratorInterface
 - jeod::JeodIntegratorInterface, 71
- init_attrjeod__JeodMemoryInterface
 - jeod::JeodMemoryInterface, 79
- init_attrjeod__JeodSimulationInterface
 - jeod::JeodSimulationInterface, 88
- init_attrjeod__JeodTrick10MemoryInterface
 - jeod::JeodTrick10MemoryInterface, 100
- init_attrjeod__JeodTrickIntegrator
 - jeod::JeodTrickIntegrator, 105
- init_attrjeod__JeodTrickMemoryInterface
 - jeod::JeodTrickMemoryInterface, 119
- init_attrjeod__JeodTrickSimInterface
 - jeod::JeodTrickSimInterface, 122
- init_attrjeod__TrickMessageHandler
 - jeod::TrickMessageHandler, 159
- init_attrjeod__TrickMessageHandlerMixin
 - jeod::TrickMessageHandlerMixin, 161
- initialize
 - jeod::CheckPointInputManager, 44
 - jeod::TrickJeodIntegrator, 155
- initialize_integ_loop
 - jeod::JeodDynbodyIntegrationLoop, 63
- InputProcessor
 - jeod::BasicJeodTrickSimInterface, 37
 - jeod::JeodDynbodyIntegrationLoop, 66
 - jeod::JeodIntegratorInterface, 71
 - jeod::JeodMemoryInterface, 79
 - jeod::JeodSimulationInterface, 88
 - jeod::JeodTrick10MemoryInterface, 100
 - jeod::JeodTrickIntegrator, 105
 - jeod::JeodTrickMemoryInterface, 119
 - jeod::JeodTrickSimInterface, 122
 - jeod::TrickMessageHandler, 159
 - jeod::TrickMessageHandlerMixin, 161
- integ_constructor
 - jeod::JeodDynbodyIntegrationLoop, 67
- integ_group
 - jeod::JeodDynbodyIntegrationLoop, 68
- integ_group_factory
 - jeod::JeodDynbodyIntegrationLoop, 68
- integ_interface
 - jeod::JeodDynbodyIntegrationLoop, 68
- integrate
 - jeod::TrickJeodIntegrator, 156
- integrate_dt
 - jeod::JeodDynbodyIntegrationLoop, 63
- integration_error
 - jeod::SimInterfaceMessages, 153
- interface_error
 - jeod::SimInterfaceMessages, 153
- interpret_integration_type
 - jeod::JeodIntegratorInterface, 70
 - jeod::JeodTrickIntegrator, 104
- is_activatable
 - jeod::SectionedInputStream, 134
 - jeod::SectionedOutputStream, 146
- is_active
 - jeod::SectionedInputStream, 136
 - jeod::SectionedOutputStream, 148
- is_array
 - jeod::JeodTrickMemoryInterface::AllocationMap←
Entry, 28
- is_checkpoint_restart_supported
 - jeod::JeodMemoryInterface, 76
 - jeod::JeodTrick10MemoryInterface, 97
 - jeod::JeodTrickMemoryInterface, 114
- is_copy
 - jeod::SectionedInputStream, 136
 - jeod::SectionedOutputStream, 148
- is_open
 - jeod::CheckPointInputManager, 45
 - jeod::CheckPointOutputManager, 52
- JEOD_ATTRIBUTES_POINTER_TYPE
 - SimInterface, 16
- JEOD_ATTRIBUTES_SIM_ENGINE_HEADER
 - SimInterface, 16
- JEOD_ATTRIBUTES_TYPE
 - SimInterface, 16, 17
- JEOD_ATTRIBUTES
 - memory_attributes.hh, 170
- JEOD_CLASS_ESTABLISH_FRIENDS1
 - SimInterface, 17
- JEOD_CLASS_ESTABLISH_FRIENDS2
 - SimInterface, 17

- JEOD_CLASS_ESTABLISH_FRIENDS3
 - SimInterface, [17](#)
- JEOD_CLASS_ESTABLISH_FRIENDS
 - SimInterface, [17](#)
- JEOD_DECLARE_ATTRIBUTES
 - memory_attributes.hh, [170](#)
- JEOD_DECLARE_SIM_INTERFACES
 - SimInterface, [18](#)
- JEOD_INTPTR_T
 - SimInterface, [18](#)
- JEOD_MAKE_SIM_INTERFACES
 - SimInterface, [18](#)
- JEOD_PTRDIFF_T
 - SimInterface, [19](#)
- JEOD_SIM_INTEGRATOR_ENUM
 - SimInterface, [19](#)
- JEOD_SIM_INTEGRATOR_FORWARD
 - SimInterface, [19](#)
- JEOD_SIM_INTEGRATOR_POINTER_TYPE
 - SimInterface, [19](#)
- JEOD_SIM_INTEGRATOR_SIM_ENGINE_HEADER
 - SimInterface, [20](#)
- JEOD_SIZE_T
 - SimInterface, [20](#)
- JEOD_UINTPTR_T
 - SimInterface, [20](#)
- JEOD_UNUSED
 - SimInterface, [20](#)
- JEODVMACRO_SEQ_COUNT
 - SimInterface, [20](#)
- JEODVMACRO_GET_COUNT
 - SimInterface, [22](#)
- JEODVMACRO_REVSEQ_COUNT
 - SimInterface, [22](#)
- JEODVMACRO_SELECT_FUNC_CAT
 - SimInterface, [23](#)
- JEODVMACRO_SELECT_FUNC
 - SimInterface, [22](#)
- JEODVMACRO_VA_SIZE
 - SimInterface, [23](#)
- JEODVMACRO
 - SimInterface, [22](#)
- jeod, [25](#)
- jeod::BasicJeodTrickSimInterface, [29](#)
 - ~BasicJeodTrickSimInterface, [31](#)
 - BasicJeodTrickSimInterface, [31](#)
 - checkpoint_allocations, [31](#)
 - checkpoint_containers, [32](#)
 - checkpoint_file_name, [37](#)
 - checkpoint_reader, [37](#)
 - checkpoint_writer, [37](#)
 - close_checkpoint_file, [32](#)
 - close_restart_file, [32](#)
 - create_integrator_internal, [32](#)
 - generic_message_handler, [38](#)
 - get_checkpoint_file_name, [33](#)
 - get_checkpoint_reader_internal, [33](#)
 - get_checkpoint_writer_internal, [33](#)
 - get_job_cycle_internal, [34](#)
 - get_memory_interface_internal, [34](#)
 - init_attrjeod__BasicJeodTrickSimInterface, [36](#)
 - InputProcessor, [37](#)
 - memory_manager, [38](#)
 - open_checkpoint_file, [34](#)
 - open_restart_file, [35](#)
 - operator=, [35](#)
 - restore_allocations, [35](#)
 - restore_containers, [35](#)
 - section_end, [38](#)
 - section_start, [38](#)
 - set_checkpoint_file_name, [36](#)
 - set_mode, [36](#)
 - trick_memory_interface, [39](#)
- jeod::CheckPointInputManager, [39](#)
 - CheckPointInputManager, [41](#)
 - create_section_reader, [41, 42](#)
 - create_trick_section_reader, [43](#)
 - current_reader, [45](#)
 - deregister_reader, [43](#)
 - filename, [45](#)
 - have_active_reader, [43](#)
 - initialize, [44](#)
 - is_open, [45](#)
 - operator!, [44](#)
 - operator=, [44](#)
 - register_reader, [44](#)
 - section_end, [45](#)
 - section_start, [46](#)
 - sections, [46](#)
 - stream, [46](#)
- jeod::CheckPointInputManager::SectionInfo, [150](#)
 - end_pos, [151](#)
 - SectionInfo, [150](#)
 - start_pos, [151](#)
- jeod::CheckPointOutputManager, [47](#)
 - CheckPointOutputManager, [48](#)
 - create_section_writer, [48, 49](#)
 - create_trick_section_writer, [49](#)
 - current_writer, [52](#)
 - deregister_writer, [50](#)
 - filename, [52](#)
 - have_active_writer, [50](#)
 - is_open, [52](#)
 - MemoryManagerWrapper, [52](#)
 - operator!, [51](#)
 - operator=, [51](#)
 - register_writer, [51](#)
 - section_end, [53](#)
 - section_start, [53](#)
 - stream, [53](#)
- jeod::JeodDynbodyIntegrationLoop, [56](#)
 - ~JeodDynbodyIntegrationLoop, [59](#)
 - add_integrable_object, [60](#)
 - add_sim_object, [60](#)
 - add_sim_object_bodies, [60, 62](#)
 - collect_derivatives, [62](#)

- deriv_ephem_update, 67
- dyn_manager, 67
- find_containing_sim_object, 62
- gravitation, 63
- gravity_manager, 67
- init_attrjeod__JeodDynbodyIntegrationLoop, 66
- initialize_integ_loop, 63
- InputProcessor, 66
- integ_constructor, 67
- integ_group, 68
- integ_group_factory, 68
- integ_interface, 68
- integrate_dt, 63
- JeodDynbodyIntegrationLoop, 58, 59
- loop_sim_object, 68
- operator=, 64
- remove_integrable_object, 64
- remove_sim_object, 64
- remove_sim_object_bodies, 65
- set_deriv_ephem_update, 65
- set_time_to_loop_start, 65
- time_manager, 69
- update_integration_group, 66
- jeod::JeodIntegratorInterface, 69
 - ~JeodIntegratorInterface, 70
 - get_integrator, 70
 - init_attrjeod__JeodIntegratorInterface, 71
 - InputProcessor, 71
 - interpret_integration_type, 70
- jeod::JeodMemoryInterface, 71
 - ~JeodMemoryInterface, 73
 - deregister_allocation, 73
 - deregister_container, 73
 - find_attributes, 74
 - get_address_at_name, 75
 - get_name_at_address, 75
 - init_attrjeod__JeodMemoryInterface, 79
 - InputProcessor, 79
 - is_checkpoint_restart_supported, 76
 - JeodMemoryInterface, 73
 - operator=, 76
 - pointer_attributes, 76
 - primitive_attributes, 77
 - register_allocation, 77
 - register_container, 78
 - structure_attributes, 78
 - void_pointer_attributes, 78
- jeod::JeodSimulationInterface, 79
 - ~JeodSimulationInterface, 82
 - configure, 82
 - create_integrator_interface, 83
 - create_integrator_internal, 83
 - get_address_at_name, 83
 - get_checkpoint_reader, 84
 - get_checkpoint_reader_internal, 84
 - get_checkpoint_writer, 84
 - get_checkpoint_writer_internal, 85
 - get_job_cycle, 85
 - get_job_cycle_internal, 85
 - get_memory_interface, 86
 - get_memory_interface_internal, 86
 - get_mode, 86
 - get_name_at_address, 87
 - init_attrjeod__JeodSimulationInterface, 88
 - InputProcessor, 88
 - JeodSimulationInterface, 82
 - Mode, 81
 - mode, 88
 - operator=, 87
 - saved_mode, 89
 - set_mode, 87
 - sim_interface, 89
- jeod::JeodSimulationInterfaceInit, 90
 - JeodSimulationInterfaceInit, 90
 - memory_debug_level, 90
 - message_suppress_id, 91
 - message_suppress_location, 91
 - message_suppression_level, 91
- jeod::JeodTrick10MemoryInterface, 92
 - ~JeodTrick10MemoryInterface, 93
 - checkpoint_allocations, 94
 - checkpoint_containers, 94
 - deregister_container, 94
 - get_address_at_name, 95
 - get_container_id, 96
 - get_name_at_address, 96
 - get_trick_checkpoint_file, 97
 - init_attrjeod__JeodTrick10MemoryInterface, 100
 - InputProcessor, 100
 - is_checkpoint_restart_supported, 97
 - JeodTrick10MemoryInterface, 93, 94
 - operator=, 97
 - register_container, 98
 - restore_allocations, 98
 - restore_containers, 99
 - translate_addr_to_name, 99
 - translate_name_to_addr, 100
 - trick_checkpoint_agent, 101
- jeod::JeodTrickIntegrator, 101
 - ~JeodTrickIntegrator, 103
 - default_first_step_deriv, 106
 - get_dt, 103
 - get_first_step_derivs_flag, 103
 - get_integrator, 103
 - init_attrjeod__JeodTrickIntegrator, 105
 - InputProcessor, 105
 - interpret_integration_type, 104
 - JeodTrickIntegrator, 102, 103
 - operator=, 104
 - reset_first_step_derivs_flag, 104
 - restore_first_step_derivs_flag, 104
 - set_first_step_derivs_flag, 104
 - set_step_number, 105
 - set_time, 105
 - trick_integrator, 106
- jeod::JeodTrickMemoryInterface, 106

- ~JeodTrickMemoryInterface, 109
- allocation_map, 119
- AllocationMap, 109
- checkpoint_allocations, 110
- checkpoint_containers, 110
- construct_identifier, 110
- container_list, 119
- ContainerList, 109
- deregister_allocation, 111
- deregister_container, 111
- dlhandle, 119
- find_attributes, 112
- get_address_at_name, 113
- get_name_at_address, 113
- get_trick_checkpoint_file, 114
- id_length, 120
- id_prefix, 120
- init_attrjeod__JeodTrickMemoryInterface, 119
- InputProcessor, 119
- is_checkpoint_restart_supported, 114
- JeodTrickMemoryInterface, 109
- mode, 120
- operator=, 114
- pointer_attributes, 115
- primitive_attributes, 115
- register_allocation, 116
- register_container, 116
- restore_allocations, 117
- restore_containers, 117
- set_mode, 117
- structure_attributes, 118
- void_pointer_attributes, 118
- jeod::JeodTrickMemoryInterface::AllocationMapEntry, 27
 - AllocationMapEntry, 27
 - is_array, 28
 - nelements, 28
 - typeid_info, 28
- jeod::JeodTrickMemoryInterface::ContainerListEntry, 54
 - container, 55
 - ContainerListEntry, 54
 - elem_name, 55
 - owner, 55
 - owner_type, 55
- jeod::JeodTrickSimInterface, 121
 - ~JeodTrickSimInterface, 122
 - init_attrjeod__JeodTrickSimInterface, 122
 - InputProcessor, 122
 - JeodTrickSimInterface, 121, 122
 - operator=, 122
- jeod::SectionedInputBuffer, 123
 - ~SectionedInputBuffer, 124
 - activate, 125
 - at_eof, 127
 - buf, 127
 - curr_pos, 128
 - deactivate, 126
 - end_pos, 128
 - file_buf, 128
 - operator!, 126
 - operator=, 126
 - SectionedInputBuffer, 124, 125
 - SectionedInputStream, 127
 - start_pos, 128
 - underflow, 126
- jeod::SectionedInputStream, 129
 - ~SectionedInputStream, 133
 - activate, 133
 - CheckPointInputManager, 135
 - deactivate, 134
 - end_pos, 136
 - is_activatable, 134
 - is_active, 136
 - is_copy, 136
 - manager, 136
 - operator void *, 134
 - operator!, 135
 - operator=, 135
 - sectbuf, 137
 - SectionedInputStream, 132, 133
 - start_pos, 137
 - stream, 137
- jeod::SectionedOutputBuffer, 138
 - ~SectionedOutputBuffer, 139
 - activate, 140
 - deactivate, 140
 - file_buf, 142
 - operator!, 140
 - operator=, 141
 - overflow, 141
 - SectionedOutputBuffer, 139
 - SectionedOutputStream, 141
- jeod::SectionedOutputStream, 142
 - ~SectionedOutputStream, 144
 - activate, 145
 - CheckPointOutputManager, 147
 - deactivate, 146
 - is_activatable, 146
 - is_active, 148
 - is_copy, 148
 - manager, 148
 - operator void *, 146
 - operator!, 147
 - operator=, 147
 - sectbuf, 148
 - section_end, 149
 - section_start, 149
 - SectionedOutputStream, 144, 145
 - stream, 149
 - tag, 149
- jeod::SimInterfaceMessages, 151
 - implementation_error, 153
 - integration_error, 153
 - interface_error, 153
 - operator=, 152
 - phasing_error, 154

- SimInterfaceMessages, 152
 - singleton_error, 154
- jeod::TrickJeodIntegrator, 155
 - ~TrickJeodIntegrator, 155
 - initialize, 155
 - integrate, 156
- jeod::TrickMessageHandler, 156
 - ~TrickMessageHandler, 157
 - init_attrjeod__TrickMessageHandler, 159
 - InputProcessor, 159
 - operator=, 157
 - process_message, 158
 - register_contents, 158
 - TrickMessageHandler, 157
- jeod::TrickMessageHandlerMixin, 159
 - ~TrickMessageHandlerMixin, 160
 - init_attrjeod__TrickMessageHandlerMixin, 161
 - InputProcessor, 161
 - message_handler, 161
 - operator=, 161
 - TrickMessageHandlerMixin, 160
- jeod_class.hh, 167
- jeod_integrator_interface.hh, 168
- jeod_trick_integrator.hh, 168
- jeod_va_macro_utility.hh, 169
- JeodDynbodyIntegrationLoop
 - jeod::JeodDynbodyIntegrationLoop, 58, 59
- JeodMemoryInterface
 - jeod::JeodMemoryInterface, 73
- JeodSimulationInterface
 - jeod::JeodSimulationInterface, 82
- JeodSimulationInterfaceInit
 - jeod::JeodSimulationInterfaceInit, 90
- JeodTrick10MemoryInterface
 - jeod::JeodTrick10MemoryInterface, 93, 94
- JeodTrickIntegrator
 - jeod::JeodTrickIntegrator, 102, 103
- JeodTrickMemoryInterface
 - jeod::JeodTrickMemoryInterface, 109
- JeodTrickSimInterface
 - jeod::JeodTrickSimInterface, 121, 122
- loop_sim_object
 - jeod::JeodDynbodyIntegrationLoop, 68
- MAKE_MESSAGE_CODE
 - SimInterface, 23
- MAX_MSG_SIZE
 - SimInterface, 23
- manager
 - jeod::SectionedInputStream, 136
 - jeod::SectionedOutputStream, 148
- memory_attributes.hh, 169
 - JEOD_ATTRIBUTES, 170
 - JEOD_DECLARE_ATTRIBUTES, 170
- memory_debug_level
 - jeod::JeodSimulationInterfaceInit, 90
- memory_interface.hh, 171
- memory_manager
 - jeod::BasicJeodTrickSimInterface, 38
- MemoryManagerWrapper
 - jeod::CheckPointOutputManager, 52
- message_handler
 - jeod::TrickMessageHandlerMixin, 161
- message_suppress_id
 - jeod::JeodSimulationInterfaceInit, 91
- message_suppress_location
 - jeod::JeodSimulationInterfaceInit, 91
- message_suppression_level
 - jeod::JeodSimulationInterfaceInit, 91
- Mode
 - jeod::JeodSimulationInterface, 81
- mode
 - jeod::JeodSimulationInterface, 88
 - jeod::JeodTrickMemoryInterface, 120
- Models, 11
- nelements
 - jeod::JeodTrickMemoryInterface::AllocationMap↔
Entry, 28
- open_checkpoint_file
 - jeod::BasicJeodTrickSimInterface, 34
- open_restart_file
 - jeod::BasicJeodTrickSimInterface, 35
- operator void *
 - jeod::SectionedInputStream, 134
 - jeod::SectionedOutputStream, 146
- operator!
 - jeod::CheckPointInputManager, 44
 - jeod::CheckPointOutputManager, 51
 - jeod::SectionedInputBuffer, 126
 - jeod::SectionedInputStream, 135
 - jeod::SectionedOutputBuffer, 140
 - jeod::SectionedOutputStream, 147
- operator=
 - jeod::BasicJeodTrickSimInterface, 35
 - jeod::CheckPointInputManager, 44
 - jeod::CheckPointOutputManager, 51
 - jeod::JeodDynbodyIntegrationLoop, 64
 - jeod::JeodMemoryInterface, 76
 - jeod::JeodSimulationInterface, 87
 - jeod::JeodTrick10MemoryInterface, 97
 - jeod::JeodTrickIntegrator, 104
 - jeod::JeodTrickMemoryInterface, 114
 - jeod::JeodTrickSimInterface, 122
 - jeod::SectionedInputBuffer, 126
 - jeod::SectionedInputStream, 135
 - jeod::SectionedOutputBuffer, 141
 - jeod::SectionedOutputStream, 147
 - jeod::SimInterfaceMessages, 152
 - jeod::TrickMessageHandler, 157
 - jeod::TrickMessageHandlerMixin, 161
- overflow
 - jeod::SectionedOutputBuffer, 141
- owner
 - jeod::JeodTrickMemoryInterface::ContainerList↔
Entry, 55

- owner_type
 - jeod::JeodTrickMemoryInterface::ContainerList↔
Entry, 55
- PATH
 - SimInterface, 23
- phasing_error
 - jeod::SimInterfaceMessages, 154
- pointer_attributes
 - jeod::JeodMemoryInterface, 76
 - jeod::JeodTrickMemoryInterface, 115
- primitive_attributes
 - jeod::JeodMemoryInterface, 77
 - jeod::JeodTrickMemoryInterface, 115
- process_message
 - jeod::TrickMessageHandler, 158
- register_allocation
 - jeod::JeodMemoryInterface, 77
 - jeod::JeodTrickMemoryInterface, 116
- register_container
 - jeod::JeodMemoryInterface, 78
 - jeod::JeodTrick10MemoryInterface, 98
 - jeod::JeodTrickMemoryInterface, 116
- register_contents
 - jeod::TrickMessageHandler, 158
- register_reader
 - jeod::CheckPointInputManager, 44
- register_writer
 - jeod::CheckPointOutputManager, 51
- remove_integrable_object
 - jeod::JeodDynbodyIntegrationLoop, 64
- remove_sim_object
 - jeod::JeodDynbodyIntegrationLoop, 64
- remove_sim_object_bodies
 - jeod::JeodDynbodyIntegrationLoop, 65
- reset_first_step_derivs_flag
 - jeod::JeodTrickIntegrator, 104
- restore_allocations
 - jeod::BasicJeodTrickSimInterface, 35
 - jeod::JeodTrick10MemoryInterface, 98
 - jeod::JeodTrickMemoryInterface, 117
- restore_containers
 - jeod::BasicJeodTrickSimInterface, 35
 - jeod::JeodTrick10MemoryInterface, 99
 - jeod::JeodTrickMemoryInterface, 117
- restore_first_step_derivs_flag
 - jeod::JeodTrickIntegrator, 104
- saved_mode
 - jeod::JeodSimulationInterface, 89
- sectbuf
 - jeod::SectionedInputStream, 137
 - jeod::SectionedOutputStream, 148
- section_end
 - jeod::BasicJeodTrickSimInterface, 38
 - jeod::CheckPointInputManager, 45
 - jeod::CheckPointOutputManager, 53
 - jeod::SectionedOutputStream, 149
- section_start
 - jeod::BasicJeodTrickSimInterface, 38
 - jeod::CheckPointInputManager, 46
 - jeod::CheckPointOutputManager, 53
 - jeod::SectionedOutputStream, 149
- SectionInfo
 - jeod::CheckPointInputManager::SectionInfo, 150
- SectionedInputBuffer
 - jeod::SectionedInputBuffer, 124, 125
- SectionedInputStream
 - jeod::SectionedInputBuffer, 127
 - jeod::SectionedInputStream, 132, 133
- SectionedOutputBuffer
 - jeod::SectionedOutputBuffer, 139
- SectionedOutputStream
 - jeod::SectionedOutputBuffer, 141
 - jeod::SectionedOutputStream, 144, 145
- sections
 - jeod::CheckPointInputManager, 46
- set_checkpoint_file_name
 - jeod::BasicJeodTrickSimInterface, 36
- set_deriv_ephem_update
 - jeod::JeodDynbodyIntegrationLoop, 65
- set_first_step_derivs_flag
 - jeod::JeodTrickIntegrator, 104
- set_mode
 - jeod::BasicJeodTrickSimInterface, 36
 - jeod::JeodSimulationInterface, 87
 - jeod::JeodTrickMemoryInterface, 117
- set_step_number
 - jeod::JeodTrickIntegrator, 105
- set_time
 - jeod::JeodTrickIntegrator, 105
- set_time_to_loop_start
 - jeod::JeodDynbodyIntegrationLoop, 65
- sim_interface
 - jeod::JeodSimulationInterface, 89
- sim_interface_messages.cc, 171
- sim_interface_messages.hh, 172
- SimInterface, 13
 - CLASS, 15
 - ER7_UTILS_ALWAYS_INLINE, 15
 - ER7_UTILS_RESTRICT, 16
 - ER7_UTILS_UNUSED, 16
 - JEOD_ATTRIBUTES_POINTER_TYPE, 16
 - JEOD_ATTRIBUTES_SIM_ENGINE_HEADER, 16
 - JEOD_ATTRIBUTES_TYPE, 16, 17
 - JEOD_CLASS_ESTABLISH_FRIENDS1, 17
 - JEOD_CLASS_ESTABLISH_FRIENDS2, 17
 - JEOD_CLASS_ESTABLISH_FRIENDS3, 17
 - JEOD_CLASS_ESTABLISH_FRIENDS, 17
 - JEOD_DECLARE_SIM_INTERFACES, 18
 - JEOD_INTPTR_T, 18
 - JEOD_MAKE_SIM_INTERFACES, 18
 - JEOD_PTRDIFF_T, 19
 - JEOD_SIM_INTEGRATOR_ENUM, 19
 - JEOD_SIM_INTEGRATOR_FORWARD, 19
 - JEOD_SIM_INTEGRATOR_POINTER_TYPE, 19

- JEOD_SIM_INTEGRATOR_SIM_ENGINE_HEA↔
DER, 20
- JEOD_SIZE_T, 20
- JEOD_UINTPTR_T, 20
- JEOD_UNUSED, 20
- JEODVMACRO_SEQ_COUNT, 20
- JEODVMACRO_GET_COUNT, 22
- JEODVMACRO_REVSEQ_COUNT, 22
- JEODVMACRO_SELECT_FUNC_CAT, 23
- JEODVMACRO_SELECT_FUNC, 22
- JEODVMACRO_VA_SIZE, 23
- JEODVMACRO, 22
- MAKE_MESSAGE_CODE, 23
- MAX_MSG_SIZE, 23
- PATH, 23
- trick_MM, 24
- trick_curr_integ, 24
- SimInterfaceMessages
 - jeod::SimInterfaceMessages, 152
- simulation_interface.cc, 172
- simulation_interface.hh, 173
- singleton_error
 - jeod::SimInterfaceMessages, 154
- start_pos
 - jeod::CheckpointInputManager::SectionInfo, 151
 - jeod::SectionedInputBuffer, 128
 - jeod::SectionedInputStream, 137
- stream
 - jeod::CheckpointInputManager, 46
 - jeod::CheckpointOutputManager, 53
 - jeod::SectionedInputStream, 137
 - jeod::SectionedOutputStream, 149
- structure_attributes
 - jeod::JeodMemoryInterface, 78
 - jeod::JeodTrickMemoryInterface, 118
- tag
 - jeod::SectionedOutputStream, 149
- time_manager
 - jeod::JeodDynbodyIntegrationLoop, 69
- translate_addr_to_name
 - jeod::JeodTrick10MemoryInterface, 99
- translate_name_to_addr
 - jeod::JeodTrick10MemoryInterface, 100
- Trick, 26
- trick10_memory_interface.cc, 173
- trick10_memory_interface.hh, 174
- trick_MM
 - SimInterface, 24
- trick_checkpoint_agent
 - jeod::JeodTrick10MemoryInterface, 101
- trick_curr_integ
 - SimInterface, 24
- trick_dynbody_integ_loop.cc, 175
- trick_dynbody_integ_loop.hh, 175
- trick_integrator
 - jeod::JeodTrickIntegrator, 106
- trick_memory_interface
 - jeod::BasicJeodTrickSimInterface, 39
 - trick_memory_interface.cc, 176
 - trick_memory_interface.hh, 176
 - trick_memory_interface_alloc.cc, 177
 - trick_memory_interface_attrib.cc, 178
 - trick_memory_interface_chkpnt.cc, 178
 - trick_memory_interface_xlate.cc, 179
 - trick_message_handler.cc, 180
 - trick_message_handler.hh, 180
 - trick_sim_interface.cc, 181
 - trick_sim_interface.hh, 181
 - TrickMessageHandler
 - jeod::TrickMessageHandler, 157
 - TrickMessageHandlerMixin
 - jeod::TrickMessageHandlerMixin, 160
 - typeid_info
 - jeod::JeodTrickMemoryInterface::AllocationMap↔
Entry, 28
 - underflow
 - jeod::SectionedInputBuffer, 126
 - update_integration_group
 - jeod::JeodDynbodyIntegrationLoop, 66
 - Utils, 12
 - void_pointer_attributes
 - jeod::JeodMemoryInterface, 78
 - jeod::JeodTrickMemoryInterface, 118