

SimulationInterfaceMacro

5.0

Generated by Doxygen 1.8.5

Wed Jun 1 2022 12:04:46

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	15
6.3.2.3	ER7_UTILS_RESTRICT	15
6.3.2.4	ER7_UTILS_UNUSED	15
6.3.2.5	JEOD_ATTRIBUTES_POINTER_TYPE	15
6.3.2.6	JEOD_ATTRIBUTES_POINTER_TYPE	15
6.3.2.7	JEOD_ATTRIBUTES_SIM_ENGINE_HEADER	15
6.3.2.8	JEOD_ATTRIBUTES_TYPE	16
6.3.2.9	JEOD_ATTRIBUTES_TYPE	16
6.3.2.10	JEOD_CLASS_ESTABLISH_FRIENDS	16

6.3.2.11	JEOD_DECLARE_SIM_INTERFACES	16
6.3.2.12	JEOD_INTPTR_T	16
6.3.2.13	JEOD_MAKE_SIM_INTERFACES	16
6.3.2.14	JEOD_PTRDIFF_T	16
6.3.2.15	JEOD_SIM_INTEGRATOR_ENUM	16
6.3.2.16	JEOD_SIM_INTEGRATOR_FORWARD	17
6.3.2.17	JEOD_SIM_INTEGRATOR_POINTER_TYPE	17
6.3.2.18	JEOD_SIM_INTEGRATOR_POINTER_TYPE	17
6.3.2.19	JEOD_SIM_INTEGRATOR_SIM_ENGINE_HEADER	17
6.3.2.20	JEOD_SIZE_T	17
6.3.2.21	JEOD_UINTPTR_T	17
6.3.2.22	JEOD_UNUSED	17
6.3.2.23	MAKE_MESSAGE_CODE	17
6.3.2.24	MAX_MSG_SIZE	17
6.3.2.25	PATH	17
6.3.3	Variable Documentation	17
6.3.3.1	trick_curr_integ	17
6.3.3.2	trick_MM	18
6.3.3.3	trick_MM	18
6.3.3.4	trick_MM	18
6.3.3.5	trick_MM	18
7	Namespace Documentation	19
7.1	er7_utils Namespace Reference	19
7.1.1	Detailed Description	19
7.2	jeod Namespace Reference	19
7.2.1	Detailed Description	20
7.3	Trick Namespace Reference	20
7.3.1	Detailed Description	20
8	Data Structure Documentation	21
8.1	jeod::JeodTrickMemoryInterface::AllocationMapEntry Struct Reference	21
8.1.1	Detailed Description	21
8.1.2	Constructor & Destructor Documentation	21
8.1.2.1	AllocationMapEntry	21
8.1.3	Field Documentation	22
8.1.3.1	is_array	22
8.1.3.2	nelements	22
8.1.3.3	typeid_info	22
8.2	jeod::BasicJeodTrickSimInterface Class Reference	22
8.2.1	Detailed Description	24

8.2.2	Constructor & Destructor Documentation	24
8.2.2.1	BasicJeodTrickSimInterface	24
8.2.2.2	~BasicJeodTrickSimInterface	25
8.2.2.3	BasicJeodTrickSimInterface	25
8.2.3	Member Function Documentation	25
8.2.3.1	checkpoint_allocations	25
8.2.3.2	checkpoint_containers	25
8.2.3.3	close_checkpoint_file	25
8.2.3.4	close_restart_file	25
8.2.3.5	create_integrator_internal	25
8.2.3.6	get_checkpoint_file_name	26
8.2.3.7	get_checkpoint_reader_internal	26
8.2.3.8	get_checkpoint_writer_internal	26
8.2.3.9	get_job_cycle_internal	26
8.2.3.10	get_memory_interface_internal	27
8.2.3.11	open_checkpoint_file	27
8.2.3.12	open_restart_file	27
8.2.3.13	operator=	27
8.2.3.14	restore_allocations	27
8.2.3.15	restore_containers	28
8.2.3.16	set_checkpoint_file_name	28
8.2.3.17	set_mode	28
8.2.4	Friends And Related Function Documentation	28
8.2.4.1	init_attrjeod__BasicJeodTrickSimInterface	28
8.2.4.2	InputProcessor	28
8.2.5	Field Documentation	28
8.2.5.1	checkpoint_file_name	28
8.2.5.2	checkpoint_reader	29
8.2.5.3	checkpoint_writer	29
8.2.5.4	generic_message_handler	29
8.2.5.5	memory_manager	29
8.2.5.6	section_end	29
8.2.5.7	section_start	29
8.2.5.8	trick_memory_interface	30
8.3	jeod::CheckPointInputManager Class Reference	30
8.3.1	Detailed Description	31
8.3.2	Constructor & Destructor Documentation	31
8.3.2.1	CheckPointInputManager	31
8.3.2.2	CheckPointInputManager	31
8.3.3	Member Function Documentation	31

8.3.3.1	create_section_reader	31
8.3.3.2	create_section_reader	32
8.3.3.3	create_trick_section_reader	32
8.3.3.4	deregister_reader	33
8.3.3.5	have_active_reader	33
8.3.3.6	initialize	33
8.3.3.7	operator!	33
8.3.3.8	operator=	33
8.3.3.9	register_reader	34
8.3.4	Field Documentation	34
8.3.4.1	current_reader	34
8.3.4.2	filename	34
8.3.4.3	is_open	34
8.3.4.4	section_end	34
8.3.4.5	section_start	34
8.3.4.6	sections	35
8.3.4.7	stream	35
8.4	jeod::CheckPointOutputManager Class Reference	35
8.4.1	Detailed Description	36
8.4.2	Constructor & Destructor Documentation	36
8.4.2.1	CheckPointOutputManager	36
8.4.2.2	CheckPointOutputManager	37
8.4.3	Member Function Documentation	37
8.4.3.1	create_section_writer	37
8.4.3.2	create_section_writer	37
8.4.3.3	create_trick_section_writer	38
8.4.3.4	deregister_writer	38
8.4.3.5	have_active_writer	38
8.4.3.6	operator!	38
8.4.3.7	operator=	38
8.4.3.8	register_writer	39
8.4.4	Friends And Related Function Documentation	39
8.4.4.1	MemoryManagerWrapper	39
8.4.5	Field Documentation	39
8.4.5.1	current_writer	39
8.4.5.2	filename	39
8.4.5.3	is_open	39
8.4.5.4	section_end	39
8.4.5.5	section_start	40
8.4.5.6	stream	40

8.5	jeod::JeodTrickMemoryInterface::ContainerListEntry Struct Reference	40
8.5.1	Detailed Description	40
8.5.2	Constructor & Destructor Documentation	41
8.5.2.1	ContainerListEntry	41
8.5.3	Field Documentation	41
8.5.3.1	container	41
8.5.3.2	elem_name	41
8.5.3.3	owner	41
8.5.3.4	owner_type	41
8.6	jeod::JeodDynbodyIntegrationLoop Class Reference	42
8.6.1	Detailed Description	44
8.6.2	Constructor & Destructor Documentation	44
8.6.2.1	JeodDynbodyIntegrationLoop	44
8.6.2.2	JeodDynbodyIntegrationLoop	44
8.6.2.3	~JeodDynbodyIntegrationLoop	45
8.6.2.4	JeodDynbodyIntegrationLoop	45
8.6.3	Member Function Documentation	45
8.6.3.1	add_integrable_object	45
8.6.3.2	add_sim_object	45
8.6.3.3	add_sim_object_bodies	46
8.6.3.4	add_sim_object_bodies	46
8.6.3.5	collect_derivatives	46
8.6.3.6	find_containing_sim_object	46
8.6.3.7	gravitation	47
8.6.3.8	initialize_integ_loop	47
8.6.3.9	integrate_dt	47
8.6.3.10	operator=	47
8.6.3.11	remove_integrable_object	47
8.6.3.12	remove_sim_object	47
8.6.3.13	remove_sim_object_bodies	48
8.6.3.14	set_deriv_ephem_update	48
8.6.3.15	set_time_to_loop_start	48
8.6.3.16	update_integration_group	48
8.6.4	Friends And Related Function Documentation	49
8.6.4.1	init_attrjeod__JeodDynbodyIntegrationLoop	49
8.6.4.2	InputProcessor	49
8.6.5	Field Documentation	49
8.6.5.1	deriv_ephem_update	49
8.6.5.2	dyn_manager	49
8.6.5.3	gravity_manager	49

8.6.5.4	integ_constructor	49
8.6.5.5	integ_group	50
8.6.5.6	integ_group_factory	50
8.6.5.7	integ_interface	50
8.6.5.8	loop_sim_object	50
8.6.5.9	time_manager	50
8.7	jeod::JeodIntegratorInterface Class Reference	50
8.7.1	Detailed Description	51
8.7.2	Constructor & Destructor Documentation	51
8.7.2.1	~JeodIntegratorInterface	51
8.7.3	Member Function Documentation	51
8.7.3.1	get_integrator	51
8.7.3.2	interpret_integration_type	52
8.7.4	Friends And Related Function Documentation	52
8.7.4.1	init_attrjeod__JeodIntegratorInterface	52
8.7.4.2	InputProcessor	52
8.8	jeod::JeodMemoryInterface Class Reference	52
8.8.1	Detailed Description	53
8.8.2	Constructor & Destructor Documentation	53
8.8.2.1	JeodMemoryInterface	53
8.8.2.2	~JeodMemoryInterface	53
8.8.2.3	JeodMemoryInterface	53
8.8.3	Member Function Documentation	54
8.8.3.1	deregister_allocation	54
8.8.3.2	deregister_container	54
8.8.3.3	find_attributes	54
8.8.3.4	find_attributes	54
8.8.3.5	get_address_at_name	55
8.8.3.6	get_name_at_address	55
8.8.3.7	is_checkpoint_restart_supported	56
8.8.3.8	operator=	56
8.8.3.9	pointer_attributes	56
8.8.3.10	primitive_attributes	56
8.8.3.11	register_allocation	57
8.8.3.12	register_container	57
8.8.3.13	structure_attributes	57
8.8.3.14	void_pointer_attributes	57
8.8.4	Friends And Related Function Documentation	58
8.8.4.1	init_attrjeod__JeodMemoryInterface	58
8.8.4.2	InputProcessor	58

8.9	jeod::JeodSimulationInterface Class Reference	58
8.9.1	Detailed Description	60
8.9.2	Member Enumeration Documentation	60
8.9.2.1	Mode	60
8.9.3	Constructor & Destructor Documentation	60
8.9.3.1	JeodSimulationInterface	60
8.9.3.2	~JeodSimulationInterface	60
8.9.3.3	JeodSimulationInterface	61
8.9.4	Member Function Documentation	61
8.9.4.1	configure	61
8.9.4.2	create_integrator_interface	61
8.9.4.3	create_integrator_internal	61
8.9.4.4	get_address_at_name	61
8.9.4.5	get_checkpoint_reader	62
8.9.4.6	get_checkpoint_reader_internal	62
8.9.4.7	get_checkpoint_writer	62
8.9.4.8	get_checkpoint_writer_internal	62
8.9.4.9	get_job_cycle	63
8.9.4.10	get_job_cycle_internal	63
8.9.4.11	get_memory_interface	63
8.9.4.12	get_memory_interface_internal	63
8.9.4.13	get_mode	64
8.9.4.14	get_name_at_address	64
8.9.4.15	operator=	64
8.9.4.16	set_mode	64
8.9.5	Friends And Related Function Documentation	65
8.9.5.1	init_attrjeod__JeodSimulationInterface	65
8.9.5.2	InputProcessor	65
8.9.6	Field Documentation	65
8.9.6.1	mode	65
8.9.6.2	saved_mode	65
8.9.6.3	sim_interface	65
8.10	jeod::JeodSimulationInterfaceInit Class Reference	65
8.10.1	Detailed Description	66
8.10.2	Constructor & Destructor Documentation	66
8.10.2.1	JeodSimulationInterfaceInit	66
8.10.3	Field Documentation	66
8.10.3.1	memory_debug_level	66
8.10.3.2	message_suppress_id	66
8.10.3.3	message_suppress_location	67

8.10.3.4	<code>message_suppression_level</code>	67
8.11	<code>jeod::JeodTrick10MemoryInterface</code> Class Reference	67
8.11.1	Detailed Description	68
8.11.2	Constructor & Destructor Documentation	69
8.11.2.1	<code>JeodTrick10MemoryInterface</code>	69
8.11.2.2	<code>~JeodTrick10MemoryInterface</code>	69
8.11.2.3	<code>JeodTrick10MemoryInterface</code>	69
8.11.3	Member Function Documentation	69
8.11.3.1	<code>checkpoint_allocations</code>	69
8.11.3.2	<code>checkpoint_containers</code>	69
8.11.3.3	<code>deregister_container</code>	69
8.11.3.4	<code>get_address_at_name</code>	70
8.11.3.5	<code>get_container_id</code>	70
8.11.3.6	<code>get_name_at_address</code>	71
8.11.3.7	<code>get_trick_checkpoint_file</code>	71
8.11.3.8	<code>is_checkpoint_restart_supported</code>	71
8.11.3.9	<code>operator=</code>	71
8.11.3.10	<code>register_container</code>	72
8.11.3.11	<code>restore_allocations</code>	72
8.11.3.12	<code>restore_containers</code>	72
8.11.3.13	<code>translate_addr_to_name</code>	73
8.11.3.14	<code>translate_name_to_addr</code>	73
8.11.4	Friends And Related Function Documentation	73
8.11.4.1	<code>init_attrjeod__JeodTrick10MemoryInterface</code>	73
8.11.4.2	<code>InputProcessor</code>	73
8.11.5	Field Documentation	73
8.11.5.1	<code>trick_checkpoint_agent</code>	74
8.12	<code>jeod::JeodTrickIntegrator</code> Class Reference	74
8.12.1	Detailed Description	75
8.12.2	Constructor & Destructor Documentation	75
8.12.2.1	<code>JeodTrickIntegrator</code>	75
8.12.2.2	<code>~JeodTrickIntegrator</code>	75
8.12.2.3	<code>JeodTrickIntegrator</code>	75
8.12.3	Member Function Documentation	76
8.12.3.1	<code>get_dt</code>	76
8.12.3.2	<code>get_first_step_derivs_flag</code>	76
8.12.3.3	<code>get_integrator</code>	76
8.12.3.4	<code>interpret_integration_type</code>	76
8.12.3.5	<code>operator=</code>	76
8.12.3.6	<code>reset_first_step_derivs_flag</code>	76

8.12.3.7	restore_first_step_derivs_flag	77
8.12.3.8	set_first_step_derivs_flag	77
8.12.3.9	set_step_number	77
8.12.3.10	set_time	77
8.12.4	Friends And Related Function Documentation	77
8.12.4.1	init_attrjeod__JeodTrickIntegrator	77
8.12.4.2	InputProcessor	77
8.12.5	Field Documentation	77
8.12.5.1	default_first_step_deriv	78
8.12.5.2	trick_integrator	78
8.13	jeod::JeodTrickMemoryInterface Class Reference	78
8.13.1	Detailed Description	80
8.13.2	Member Typedef Documentation	80
8.13.2.1	AllocationMap	80
8.13.2.2	ContainerList	81
8.13.3	Constructor & Destructor Documentation	81
8.13.3.1	JeodTrickMemoryInterface	81
8.13.3.2	~JeodTrickMemoryInterface	81
8.13.3.3	JeodTrickMemoryInterface	81
8.13.4	Member Function Documentation	81
8.13.4.1	checkpoint_allocations	81
8.13.4.2	checkpoint_containers	81
8.13.4.3	construct_identifier	82
8.13.4.4	deregister_allocation	82
8.13.4.5	deregister_container	82
8.13.4.6	find_attributes	83
8.13.4.7	find_attributes	83
8.13.4.8	get_address_at_name	83
8.13.4.9	get_name_at_address	83
8.13.4.10	get_trick_checkpoint_file	84
8.13.4.11	is_checkpoint_restart_supported	84
8.13.4.12	operator=	84
8.13.4.13	pointer_attributes	84
8.13.4.14	primitive_attributes	85
8.13.4.15	register_allocation	85
8.13.4.16	register_container	85
8.13.4.17	restore_allocations	87
8.13.4.18	restore_containers	87
8.13.4.19	set_mode	87
8.13.4.20	structure_attributes	87

8.13.4.21	void_pointer_attributes	88
8.13.5	Friends And Related Function Documentation	88
8.13.5.1	init_attrjeod__JeodTrickMemoryInterface	88
8.13.5.2	InputProcessor	88
8.13.6	Field Documentation	88
8.13.6.1	allocation_map	88
8.13.6.2	container_list	88
8.13.6.3	dlhandle	89
8.13.6.4	id_length	89
8.13.6.5	id_prefix	89
8.13.6.6	mode	89
8.14	jeod::JeodTrickSimInterface Class Reference	89
8.14.1	Detailed Description	90
8.14.2	Constructor & Destructor Documentation	90
8.14.2.1	JeodTrickSimInterface	90
8.14.2.2	~JeodTrickSimInterface	90
8.14.2.3	JeodTrickSimInterface	91
8.14.3	Member Function Documentation	91
8.14.3.1	operator=	91
8.14.4	Friends And Related Function Documentation	91
8.14.4.1	init_attrjeod__JeodTrickSimInterface	91
8.14.4.2	InputProcessor	91
8.15	jeod::SectionedInputBuffer Class Reference	91
8.15.1	Detailed Description	92
8.15.2	Constructor & Destructor Documentation	92
8.15.2.1	~SectionedInputBuffer	92
8.15.2.2	SectionedInputBuffer	93
8.15.2.3	SectionedInputBuffer	93
8.15.3	Member Function Documentation	93
8.15.3.1	activate	93
8.15.3.2	deactivate	93
8.15.3.3	operator!	93
8.15.3.4	operator=	94
8.15.3.5	underflow	94
8.15.4	Friends And Related Function Documentation	94
8.15.4.1	SectionedInputStream	94
8.15.5	Field Documentation	94
8.15.5.1	at_eof	94
8.15.5.2	buf	94
8.15.5.3	curr_pos	94

8.15.5.4	<code>end_pos</code>	95
8.15.5.5	<code>file_buf</code>	95
8.15.5.6	<code>start_pos</code>	95
8.16	<code>jeod::SectionedInputStream</code> Class Reference	95
8.16.1	Detailed Description	97
8.16.2	Constructor & Destructor Documentation	98
8.16.2.1	<code>SectionedInputStream</code>	98
8.16.2.2	<code>SectionedInputStream</code>	98
8.16.2.3	<code>~SectionedInputStream</code>	99
8.16.2.4	<code>SectionedInputStream</code>	99
8.16.3	Member Function Documentation	99
8.16.3.1	<code>activate</code>	99
8.16.3.2	<code>deactivate</code>	99
8.16.3.3	<code>is_activatable</code>	100
8.16.3.4	<code>operator void *</code>	100
8.16.3.5	<code>operator!</code>	100
8.16.3.6	<code>operator=</code>	100
8.16.4	Friends And Related Function Documentation	100
8.16.4.1	<code>CheckpointInputManager</code>	100
8.16.5	Field Documentation	101
8.16.5.1	<code>end_pos</code>	101
8.16.5.2	<code>is_active</code>	101
8.16.5.3	<code>is_copy</code>	101
8.16.5.4	<code>manager</code>	101
8.16.5.5	<code>sectbuf</code>	101
8.16.5.6	<code>start_pos</code>	101
8.16.5.7	<code>stream</code>	102
8.17	<code>jeod::SectionedOutputBuffer</code> Class Reference	102
8.17.1	Detailed Description	103
8.17.2	Constructor & Destructor Documentation	103
8.17.2.1	<code>~SectionedOutputBuffer</code>	103
8.17.2.2	<code>SectionedOutputBuffer</code>	103
8.17.2.3	<code>SectionedOutputBuffer</code>	103
8.17.2.4	<code>SectionedOutputBuffer</code>	103
8.17.3	Member Function Documentation	103
8.17.3.1	<code>activate</code>	103
8.17.3.2	<code>deactivate</code>	104
8.17.3.3	<code>operator!</code>	104
8.17.3.4	<code>operator=</code>	104
8.17.3.5	<code>overflow</code>	104

8.17.4	Friends And Related Function Documentation	105
8.17.4.1	SectionedOutputStream	105
8.17.5	Field Documentation	105
8.17.5.1	file_buf	105
8.18	jeod::SectionedOutputStream Class Reference	105
8.18.1	Detailed Description	106
8.18.2	Constructor & Destructor Documentation	107
8.18.2.1	SectionedOutputStream	107
8.18.2.2	SectionedOutputStream	107
8.18.2.3	~SectionedOutputStream	107
8.18.2.4	SectionedOutputStream	107
8.18.3	Member Function Documentation	107
8.18.3.1	activate	107
8.18.3.2	deactivate	108
8.18.3.3	is_activatable	108
8.18.3.4	operator void *	108
8.18.3.5	operator!	108
8.18.3.6	operator=	109
8.18.4	Friends And Related Function Documentation	109
8.18.4.1	CheckpointOutputManager	109
8.18.5	Field Documentation	109
8.18.5.1	is_active	109
8.18.5.2	is_copy	109
8.18.5.3	manager	109
8.18.5.4	sectbuf	109
8.18.5.5	section_end	109
8.18.5.6	section_start	110
8.18.5.7	stream	110
8.18.5.8	tag	110
8.19	jeod::CheckpointInputManager::SectionInfo Struct Reference	110
8.19.1	Detailed Description	110
8.19.2	Constructor & Destructor Documentation	111
8.19.2.1	SectionInfo	111
8.19.3	Field Documentation	111
8.19.3.1	end_pos	111
8.19.3.2	start_pos	111
8.20	jeod::SimInterfaceMessages Class Reference	111
8.20.1	Detailed Description	112
8.20.2	Constructor & Destructor Documentation	112
8.20.2.1	SimInterfaceMessages	112

8.20.2.2	SimInterfaceMessages	112
8.20.3	Member Function Documentation	112
8.20.3.1	operator=	112
8.20.4	Field Documentation	112
8.20.4.1	implementation_error	112
8.20.4.2	integration_error	112
8.20.4.3	interface_error	112
8.20.4.4	phasing_error	113
8.20.4.5	singleton_error	113
8.21	jeod::TrickJeodIntegrator Class Reference	113
8.21.1	Detailed Description	114
8.21.2	Constructor & Destructor Documentation	114
8.21.2.1	~TrickJeodIntegrator	114
8.21.3	Member Function Documentation	114
8.21.3.1	initialize	114
8.21.3.2	integrate	114
8.22	jeod::TrickMessageHandler Class Reference	114
8.22.1	Detailed Description	115
8.22.2	Constructor & Destructor Documentation	115
8.22.2.1	TrickMessageHandler	115
8.22.2.2	~TrickMessageHandler	115
8.22.2.3	TrickMessageHandler	115
8.22.3	Member Function Documentation	116
8.22.3.1	operator=	116
8.22.3.2	process_message	116
8.22.3.3	register_contents	116
8.22.4	Friends And Related Function Documentation	116
8.22.4.1	init_attrjeod__TrickMessageHandler	116
8.22.4.2	InputProcessor	116
8.23	jeod::TrickMessageHandlerMixin Class Reference	116
8.23.1	Detailed Description	117
8.23.2	Constructor & Destructor Documentation	117
8.23.2.1	TrickMessageHandlerMixin	117
8.23.2.2	~TrickMessageHandlerMixin	117
8.23.2.3	TrickMessageHandlerMixin	118
8.23.3	Member Function Documentation	118
8.23.3.1	operator=	118
8.23.4	Friends And Related Function Documentation	118
8.23.4.1	init_attrjeod__TrickMessageHandlerMixin	118
8.23.4.2	InputProcessor	118

8.23.5	Field Documentation	118
8.23.5.1	message_handler	118
9	File Documentation	119
9.1	checkpoint_input_manager.cc File Reference	119
9.1.1	Detailed Description	119
9.2	checkpoint_input_manager.hh File Reference	119
9.2.1	Detailed Description	120
9.3	checkpoint_output_manager.cc File Reference	120
9.3.1	Detailed Description	120
9.4	checkpoint_output_manager.hh File Reference	120
9.4.1	Detailed Description	121
9.5	class_declarations.hh File Reference	121
9.5.1	Detailed Description	121
9.6	config.hh File Reference	121
9.6.1	Detailed Description	122
9.7	config_test_harness.hh File Reference	122
9.7.1	Detailed Description	122
9.8	config_trick10.hh File Reference	122
9.8.1	Detailed Description	122
9.9	jeod_class.hh File Reference	123
9.9.1	Detailed Description	123
9.10	jeod_integrator_interface.hh File Reference	123
9.10.1	Detailed Description	124
9.11	jeod_trick_integrator.hh File Reference	124
9.11.1	Detailed Description	124
9.12	memory_attributes.hh File Reference	124
9.12.1	Detailed Description	125
9.12.2	Macro Definition Documentation	125
9.12.2.1	JEOD_ATTRIBUTES	125
9.12.2.2	JEOD_DECLARE_ATTRIBUTES	125
9.13	memory_interface.cc File Reference	125
9.13.1	Detailed Description	126
9.14	memory_interface.hh File Reference	126
9.14.1	Detailed Description	126
9.15	sim_interface_messages.cc File Reference	126
9.15.1	Detailed Description	127
9.16	sim_interface_messages.hh File Reference	127
9.16.1	Detailed Description	127
9.17	simulation_interface.cc File Reference	127

9.17.1 Detailed Description	128
9.18 simulation_interface.hh File Reference	128
9.18.1 Detailed Description	128
9.19 trick10_memory_interface.cc File Reference	128
9.19.1 Detailed Description	129
9.20 trick10_memory_interface.hh File Reference	129
9.20.1 Detailed Description	129
9.21 trick_dynbody_integ_loop.cc File Reference	130
9.21.1 Detailed Description	130
9.22 trick_dynbody_integ_loop.hh File Reference	130
9.22.1 Detailed Description	131
9.23 trick_memory_interface.cc File Reference	131
9.23.1 Detailed Description	131
9.24 trick_memory_interface.hh File Reference	131
9.24.1 Detailed Description	132
9.25 trick_memory_interface_alloc.cc File Reference	132
9.25.1 Detailed Description	133
9.26 trick_memory_interface_attrib.cc File Reference	133
9.26.1 Detailed Description	133
9.27 trick_memory_interface_chkpnt.cc File Reference	133
9.27.1 Detailed Description	134
9.28 trick_memory_interface_xlate.cc File Reference	134
9.28.1 Detailed Description	134
9.29 trick_message_handler.cc File Reference	135
9.29.1 Detailed Description	135
9.30 trick_message_handler.hh File Reference	135
9.30.1 Detailed Description	136
9.31 trick_sim_interface.cc File Reference	136
9.31.1 Detailed Description	136
9.32 trick_sim_interface.hh File Reference	136
9.32.1 Detailed Description	137

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	19
jeod	Namespace jeod	19
Trick	Namespace Trick furnishes several standard functions for use in the Trick environment	20

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

jeod::JeodTrickMemoryInterface::AllocationMapEntry	21
jeod::CheckPointInputManager	30
jeod::CheckPointOutputManager	35
jeod::JeodTrickMemoryInterface::ContainerListEntry	40
IntegLoopScheduler	
jeod::JeodDynbodyIntegrationLoop	42
Integrator	
jeod::TrickJeodIntegrator	113
IntegratorInterface	
jeod::JeodIntegratorInterface	50
jeod::JeodTrickIntegrator	74
std::ios_base	
std::basic_ios	
std::basic_istream	
std::istream	
jeod::SectionedInputStream	95
std::basic_ostream	
std::ostream	
jeod::SectionedOutputStream	105
JeodIntegrationGroupOwner	
jeod::JeodDynbodyIntegrationLoop	42
jeod::JeodMemoryInterface	52
jeod::JeodTrickMemoryInterface	78
jeod::JeodTrick10MemoryInterface	67
jeod::JeodSimulationInterface	58
jeod::BasicJeodTrickSimInterface	22
jeod::JeodTrickSimInterface	89
jeod::JeodSimulationInterfaceInit	65
jeod::CheckPointInputManager::SectionInfo	110
jeod::SimInterfaceMessages	111
streambuf	
jeod::SectionedInputBuffer	91
jeod::SectionedOutputBuffer	102
SuppressedCodeMessageHandler	
jeod::TrickMessageHandler	114
jeod::TrickMessageHandlerMixin	116
jeod::JeodTrickSimInterface	89

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

jeod::JeodTrickMemoryInterface::AllocationMapEntry	21
Describes a chunk of JEOD-allocated memory	
jeod::BasicJeodTrickSimInterface	22
The BasicJeodTrickSimInterface implements the required capabilities of the generic Jeod-SimulationInterface in a Trick simulation environment	
jeod::CheckPointInputManager	30
A CheckPointInputManager provides tools for reading a checkpoint file	
jeod::CheckPointOutputManager	35
A CheckPointOutputManager provides the basic tools for writing a checkpoint file	
jeod::JeodTrickMemoryInterface::ContainerListEntry	40
Describes a Checkpointable object	
jeod::JeodDynbodyIntegrationLoop	42
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	
jeod::JeodIntegratorInterface	50
A JeodIntegratorInterface extends the ER7 IntegratorInterface with the concept of a pointer to the simulation engine's integration object	
jeod::JeodMemoryInterface	52
Abstract interface between the JEOD memory manager and the simulation engine	
jeod::JeodSimulationInterface	58
This abstract class defines the basis for the interface between JEOD and a simulation engine	
jeod::JeodSimulationInterfaceInit	65
Define configuration data needed to configure the dynamically-created message handler and memory manager	
jeod::JeodTrick10MemoryInterface	67
A TrickMemoryInterface implements the two required methods needed to register and deregister memory with the simulation engine, Trick in this case	
jeod::JeodTrickIntegrator	74
A JeodTrickIntegrator specializes the JeodIntegratorInterface for use with Trick as the simulation engine	
jeod::JeodTrickMemoryInterface	78
A TrickMemoryInterface implements the two required methods needed to register and deregister memory with the simulation engine, Trick in this case	
jeod::JeodTrickSimInterface	89
A JEODTrickSimInterface implements the required capabilities of the generic JeodSimulation-Interface in a Trick simulation environment	

jeod::SectionedInputBuffer	
A SectionedInputBuffer is a <code>std::streambuf</code> that reads from a section in a checkpoint file	91
jeod::SectionedInputStream	
A SectionedInputStream is a <code>std::istream</code> that reads from a section in a checkpoint file	95
jeod::SectionedOutputBuffer	
A SectionedOutputBuffer is a <code>std::streambuf</code> that writes a section of a checkpoint file	102
jeod::SectionedOutputStream	
A SectionedOutputStream is a <code>std::ostream</code> that writes a section of a checkpoint file	105
jeod::CheckpointInputManager::SectionInfo	
A SectionInfo contains the start and end positions of a checkpoint file section	110
jeod::SimInterfaceMessages	
Specifies the message IDs used in the <code>sim_interface</code> model	111
jeod::TrickJeodIntegrator	
A TrickJeodIntegrator specializes the <code>Trick::Integrator</code> to provide the Trick side of the integration interface between Trick and JEOD	113
jeod::TrickMessageHandler	
The <code>MessageHandler</code> class for designed for use in Trick -based simulations	114
jeod::TrickMessageHandlerMixin	
The TrickMessageHandlerMixin implements the required capabilities of the generic Jeod-SimulationInterface in a Trick simulation environment	116

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	119
checkpoint_input_manager.hh	Define class CheckPointInputManager and related classes	119
checkpoint_output_manager.cc	Define CheckPointOutputManager member functions and of related classes	120
checkpoint_output_manager.hh	Define class CheckPointOutputManager and related classes	120
class_declarations.hh	Forward declarations of classes defined in the utils/sim_interface model	121
config.hh	Configure JEOD for use by some simulation engine	121
config_test_harness.hh	Configure JEOD for use in standalone test mode	122
config_trick10.hh	Configure JEOD for use in a Trick10 environment	122
jeod_class.hh	Define the JEOD class declaration macros JEOD_MAKE_SIM_INTERFACES and and JEOD_DECLARE_SIM_INTERFACES	123
jeod_integrator_interface.hh	Define the interface for accessing / updating elements of a simulation engine's integrator object	123
jeod_trick_integrator.hh	Define the interface for accessing / updating elements of a Trick simulation integrator object	124
memory_attributes.hh	Define JEOD memory interface macros	124
memory_interface.cc	Implement the MemoryInterface class	125
memory_interface.hh	Define the MemoryInterface class, which abstractly defines the interface between the memory manager and the simulation engine	126
sim_interface_messages.cc	Implement the class SimInterfaceMessages	126
sim_interface_messages.hh	Define the class SimInterfaceMessages, the class that specifies the message IDs used in the sim interface model	127
simulation_interface.cc	Implement SimulationInterface methods	127

simulation_interface.hh	Define the abstract class JeodSimulationInterface	128
trick10_memory_interface.cc	Define JeodTrickMemoryInterface methods	128
trick10_memory_interface.hh	Define the interface for registering / deregistering memory with Trick	129
trick_dynbody_integ_loop.cc	Define JeodDynbodyIntegrationLoop methods	130
trick_dynbody_integ_loop.hh	Define the class IntegrationGroupIntegLoopScheduler, which replaces the base Trick integration loop for multi-rate JEOD-based simulations	130
trick_memory_interface.cc	Define JeodTrickMemoryInterface methods	131
trick_memory_interface.hh	Define the interface for registering / deregistering memory with Trick	131
trick_memory_interface_alloc.cc	Define JeodTrickMemoryInterface methods related to allocation/deallocation	132
trick_memory_interface_attrib.cc	Define JeodTrickMemoryInterface methods related to attributes	133
trick_memory_interface_chkpnt.cc	Define JeodTrick10MemoryInterface methods related to checkpoint/restart	133
trick_memory_interface_xlate.cc	Define JeodTrickMemoryInterface methods related to name translation	134
trick_message_handler.cc	Define member functions for the class TrickMessageHandler	135
trick_message_handler.hh	Define the class TrickMessageHandler, the message handler designed for use in Trick-based simulations	135
trick_sim_interface.cc	Implement TrickSimInterface methods	136
trick_sim_interface.hh	Define the class JeodTrickSimInterface	136

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 JEOD_DECLARE_SIM_INTERFACES.
- 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 [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 [memory_interface.cc](#)
Implement the MemoryInterface class.
- 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_chkpt.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_FRIENDS(class_name)`
- `#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 namespace Trick { class Integrator; }`
- `#define JEOD_SIM_INTEGRATOR_POINTER_TYPE Trick::Integrator *`
- `#define JEOD_SIM_INTEGRATOR_ENUM Integrator_type`

- `#define JEOD_MAKE_SIM_INTERFACES(class_name) JEOD_CLASS_ESTABLISH_FRIENDS(class_name)`
`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 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 `#define CLASS SimInterfaceMessages`

Definition at line 40 of file `sim_interface_messages.cc`.

6.3.2.2 `#define ER7_UTILS_ALWAYS_INLINE`

Definition at line 79 of file `config.hh`.

6.3.2.3 `#define ER7_UTILS_RESTRICT`

Definition at line 74 of file `config.hh`.

6.3.2.4 `#define ER7_UTILS_UNUSED`

Definition at line 69 of file `config.hh`.

6.3.2.5 `#define JEOD_ATTRIBUTES_POINTER_TYPE void *`

Definition at line 45 of file `config_test_harness.hh`.

6.3.2.6 `#define JEOD_ATTRIBUTES_POINTER_TYPE JEOD_ATTRIBUTES_TYPE *`

Definition at line 58 of file `config_trick10.hh`.

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

Definition at line 55 of file `config_trick10.hh`.

6.3.2.8 #define JEOD_ATTRIBUTES_TYPE int

Definition at line 44 of file config_test_harness.hh.

6.3.2.9 #define JEOD_ATTRIBUTES_TYPE struct ATTRIBUTES_tag

Definition at line 57 of file config_trick10.hh.

6.3.2.10 #define JEOD_CLASS_ESTABLISH_FRIENDS(class_name)

Value:

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

Definition at line 42 of file config_trick10.hh.

6.3.2.11 #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 107 of file jeod_class.hh.

6.3.2.12 #define JEOD_INTPTR_T long int

Definition at line 33 of file config_trick10.hh.

6.3.2.13 #define JEOD_MAKE_SIM_INTERFACES(class_name) JEOD_CLASS_ESTABLISH_FRIENDS(class_name)

[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 71 of file jeod_class.hh.

6.3.2.14 #define JEOD_PTRDIFF_T long int

Definition at line 32 of file config_trick10.hh.

6.3.2.15 #define JEOD_SIM_INTEGRATOR_ENUM Integrator_type

Definition at line 73 of file config_trick10.hh.

6.3.2.16 `#define JEOD_SIM_INTEGRATOR_FORWARD namespace Trick { class Integrator; }`

Definition at line 70 of file config_trick10.hh.

6.3.2.17 `#define JEOD_SIM_INTEGRATOR_POINTER_TYPE void *`

Definition at line 55 of file config_test_harness.hh.

6.3.2.18 `#define JEOD_SIM_INTEGRATOR_POINTER_TYPE Trick::Integrator *`

Definition at line 72 of file config_trick10.hh.

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

Definition at line 68 of file config_trick10.hh.

6.3.2.20 `#define JEOD_SIZE_T size_t`

Definition at line 31 of file config_trick10.hh.

6.3.2.21 `#define JEOD_UINTPTR_T unsigned long int`

Definition at line 34 of file config_trick10.hh.

6.3.2.22 `#define JEOD_UNUSED`

Definition at line 64 of file config.hh.

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

Definition at line 41 of file sim_interface_messages.cc.

6.3.2.24 `#define MAX_MSG_SIZE 4096`

Definition at line 50 of file trick_message_handler.cc.

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

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

Definition at line 39 of file sim_interface_messages.cc.

6.3.3 Variable Documentation

6.3.3.1 `Trick::Integrator*` `trick_curr_integ`

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

6.3.3.2 `Trick::MemoryManager*` `trick_MM`

6.3.3.3 `Trick::MemoryManager*` `trick_MM`

6.3.3.4 `Trick::MemoryManager*` `trick_MM`

6.3.3.5 `Trick::MemoryManager*` `trick_MM`

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

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 [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 [CheckPointInputManager](#)
A [CheckPointInputManager](#) provides tools for reading 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 [CheckPointOutputManager](#)
A [CheckPointOutputManager](#) provides the basic tools for writing a checkpoint file.
- class [JeodIntegratorInterface](#)
A [JeodIntegratorInterface](#) extends the `ER7 IntegratorInterface` with the concept of a pointer to the simulation engine's integration object.
- class [TrickJeodIntegrator](#)
A [TrickJeodIntegrator](#) specializes the `Trick::Integrator` to provide the [Trick](#) side of the integration interface between [Trick](#) and JEOD.
- class [JeodTrickIntegrator](#)
A [JeodTrickIntegrator](#) specializes the [JeodIntegratorInterface](#) for use with [Trick](#) as the simulation engine.
- class [JeodMemoryInterface](#)

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

 - class [SimInterfaceMessages](#)

Specifies the message IDs used in the `sim_interface` model.
 - class [JeodSimulationInterfaceInit](#)

Define configuration data needed to configure the dynamically-created message handler and memory manager.
 - class [JeodSimulationInterface](#)

This abstract class defines the basis for the interface between JEOD and a simulation engine.
 - class [JeodTrick10MemoryInterface](#)

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

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

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

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

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

A `JEODTrickSimInterface` 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 263 of file trick_memory_interface.hh.

8.1.2 Constructor & Destructor Documentation

8.1.2.1 `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 286 of file `trick_memory_interface.hh`.

8.1.3 Field Documentation

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

Is the item an array or a single object?

`trick_units(-)`

Definition at line 278 of file `trick_memory_interface.hh`.

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

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

The number of elements in the allocated chunk of memory.

`trick_units(-)`

Definition at line 273 of file `trick_memory_interface.hh`.

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

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

Descriptor of the data type.

`trick_units(-)`

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

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

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

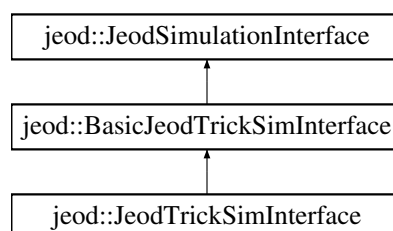
- [trick_memory_interface.hh](#)

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

Protected Member Functions

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

Private Member Functions

- [BasicJeodTrickSimInterface](#) (const [BasicJeodTrickSimInterface](#) &)
Not implemented.
- [BasicJeodTrickSimInterface](#) & `operator=` (const [BasicJeodTrickSimInterface](#) &)
Not implemented.

Friends

- class [InputProcessor](#)
- void `init_attrjeod__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 57 of file `trick_sim_interface.hh`.

8.2.2 Constructor & Destructor Documentation

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

Construct a [BasicJeodTrickSimInterface](#) object.

Parameters

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

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

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

8.2.2.2 `jeod::BasicJeodTrickSimInterface::~~BasicJeodTrickSimInterface (void) [virtual]`

Destroy a [BasicJeodTrickSimInterface](#) object.

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

References `checkpoint_reader`, and `checkpoint_writer`.

8.2.2.3 `jeod::BasicJeodTrickSimInterface::BasicJeodTrickSimInterface (const BasicJeodTrickSimInterface &) [private]`

Not implemented.

8.2.3 Member Function Documentation**8.2.3.1** `void jeod::BasicJeodTrickSimInterface::checkpoint_allocations (void)`

Dump the allocation information to the checkpoint file.

Definition at line 325 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 `void jeod::BasicJeodTrickSimInterface::checkpoint_containers (void)`

Dump the container objects to the checkpoint file.

Definition at line 351 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 `void jeod::BasicJeodTrickSimInterface::close_checkpoint_file (void)`

Close the checkpoint output file.

Definition at line 238 of file `trick_sim_interface.cc`.

References `checkpoint_writer`.

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

Close the checkpoint input file.

Definition at line 311 of file `trick_sim_interface.cc`.

References `checkpoint_reader`.

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

Create an integration interface object.

Returns

Integrator interface that encapsulates an sim engine integrator.

Implements [jeod::JeodSimulationInterface](#).

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

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

Get the checkpoint file name.

Definition at line 78 of file `trick_sim_interface.hh`.

References `checkpoint_file_name`.

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

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

Returns

Checkpoint reader

Parameters

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

Implements [jeod::JeodSimulationInterface](#).

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

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

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

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

Returns

Checkpoint writer

Parameters

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

Implements [jeod::JeodSimulationInterface](#).

Definition at line 219 of file `trick_sim_interface.cc`.

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

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

Get the current job's cycle time.

Returns

Current job's cycle time
Units: s

Implements [jeod::JeodSimulationInterface](#).

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

8.2.3.10 JeodMemoryInterface & jeod::BasicJeodTrickSimInterface::get_memory_interface_internal (void) [protected], [virtual]

Get the memory interface.

Returns

Memory interface

Implements [jeod::JeodSimulationInterface](#).

Definition at line 168 of file `trick_sim_interface.cc`.

References `trick_memory_interface`.

8.2.3.11 void jeod::BasicJeodTrickSimInterface::open_checkpoint_file (void)

Open the checkpoint output file.

Definition at line 179 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 void jeod::BasicJeodTrickSimInterface::open_restart_file (void)

Open the checkpoint input file.

Definition at line 252 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 BasicJeodTrickSimInterface& jeod::BasicJeodTrickSimInterface::operator= (const BasicJeodTrickSimInterface &) [private]

Not implemented.

8.2.3.14 void jeod::BasicJeodTrickSimInterface::restore_allocations (void)

Restore the allocated data per the checkpoint file.

Definition at line 338 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 void jeod::BasicJeodTrickSimInterface::restore_containers (void)

Restore the container objects from the checkpoint file.

Definition at line 364 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 void jeod::BasicJeodTrickSimInterface::set_checkpoint_file_name (std::string name) [inline]

Set the checkpoint file name.

Definition at line 72 of file `trick_sim_interface.hh`.

References `checkpoint_file_name`.

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

Set the mode.

Assumptions and Limitations

- See `SimulationInterface::set_mode`.

Parameters

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

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

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

References `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 void init_attrjeod__BasicJeodTrickSimInterface () [friend]

8.2.4.2 friend class InputProcessor [friend]

Definition at line 58 of file `trick_sim_interface.hh`.

8.2.5 Field Documentation

8.2.5.1 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 168 of file `trick_sim_interface.hh`.

Referenced by `get_checkpoint_file_name()`, `open_checkpoint_file()`, `open_restart_file()`, and `set_checkpoint_file_name()`.

8.2.5.2 CheckPointInputManager* jeod::BasicJeodTrickSimInterface::checkpoint_reader [protected]

The object that manages reading from a checkpoint file.

trick_io(**)

Definition at line 183 of file trick_sim_interface.hh.

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

8.2.5.3 CheckPointOutputManager* jeod::BasicJeodTrickSimInterface::checkpoint_writer [protected]

The object that manages writing to a checkpoint file.

trick_io(**)

Definition at line 188 of file trick_sim_interface.hh.

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

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

The global MessageHandler.

trick_units(-)

Definition at line 149 of file trick_sim_interface.hh.

Referenced by BasicJeodTrickSimInterface().

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

The global JEOD memory manager.

trick_units(-)

Definition at line 159 of file trick_sim_interface.hh.

Referenced by restore_allocations(), and set_mode().

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

String indicating the end of a checkpoint file section.

trick_io(*o) trick_units(-)

Definition at line 178 of file trick_sim_interface.hh.

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

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

String indicating the start of a checkpoint file section.

trick_io(*o) trick_units(-)

Definition at line 173 of file trick_sim_interface.hh.

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

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

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

trick_units(-)

Definition at line 154 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.
- [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 ([SectionedInputStream](#) *reader)
Deregister the supplied section reader as the currently-active reader.

Private Member Functions

- void initialize (void)
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.
- [CheckPointInputManager](#) (const [CheckPointInputManager](#) &)
Not implemented.
- [CheckPointInputManager](#) & operator= (const [CheckPointInputManager](#) &)
Not implemented.

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`
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 383 of file `checkpoint_input_manager.hh`.

8.3.2 Constructor & Destructor Documentation

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

Construct a [CheckPointInputManager](#) object.

Parameters

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

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

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

8.3.2.2 `jeod::CheckPointInputManager::CheckPointInputManager (const CheckPointInputManager &) [private]`

Not implemented.

8.3.3 Member Function Documentation

8.3.3.1 `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 402 of file `checkpoint_input_manager.hh`.

Referenced by `create_trick_section_reader()`, and `jeod::BasicJeodTrickSimInterface::get_checkpoint_reader_internal()`.

8.3.3.2 [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

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

Definition at line 427 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 [SectionedInputStream](#) `jeod::CheckPointInputManager::create_trick_section_reader (void)` `[private]`

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

Returns

[Trick SectionedInputStream](#) object.

Definition at line 470 of file `checkpoint_input_manager.cc`.

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

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

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

Returns

True => success.

Parameters

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

Definition at line 507 of file checkpoint_input_manager.cc.

References `current_reader`.

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

8.3.3.5 `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 418 of file checkpoint_input_manager.hh.

References `current_reader`.

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

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

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

Definition at line 347 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 `bool jeod::CheckPointInputManager::operator! () const [inline]`

Conversion to boolean.

Returns

False if object is OK.

Definition at line 410 of file checkpoint_input_manager.hh.

References `is_open`, and `stream`.

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

Not implemented.

8.3.3.9 `bool jeod::CheckpointInputManager::register_reader (SectionedInputStream * reader)`

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

Returns

True => success.

Parameters

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

Definition at line 488 of file `checkpoint_input_manager.cc`.

References `current_reader`.

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

8.3.4 Field Documentation

8.3.4.1 `SectionedInputStream* jeod::CheckpointInputManager::current_reader` `[private]`

The reader that currently is active.

`trick_io(**)`

Definition at line 488 of file `checkpoint_input_manager.hh`.

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

8.3.4.2 `const std::string jeod::CheckpointInputManager::filename` `[private]`

The name of the checkpoint file.

`trick_io(**)`

Definition at line 493 of file `checkpoint_input_manager.hh`.

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

8.3.4.3 `bool jeod::CheckpointInputManager::is_open` `[private]`

Is the checkpoint file open?

`trick_io(**)`

Definition at line 508 of file `checkpoint_input_manager.hh`.

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

8.3.4.4 `const std::string& jeod::CheckpointInputManager::section_end` `[private]`

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

`trick_io(**)`

Definition at line 503 of file `checkpoint_input_manager.hh`.

Referenced by `initialize()`.

8.3.4.5 `const std::string& jeod::CheckpointInputManager::section_start` `[private]`

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

trick_io(**)

Definition at line 498 of file checkpoint_input_manager.hh.

Referenced by initialize().

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

Maps section names to section start/end positions.

trick_io(**)

Definition at line 478 of file checkpoint_input_manager.hh.

Referenced by create_section_reader(), and initialize().

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

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

trick_io(**)

Definition at line 483 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.
- [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 ([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.
- [CheckPointOutputManager](#) (const [CheckPointOutputManager](#) &)
Not implemented.
- [CheckPointOutputManager](#) & [operator=](#) (const [CheckPointOutputManager](#) &)
Not implemented.

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](#)
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 243 of file `checkpoint_output_manager.hh`.

8.4.2 Constructor & Destructor Documentation

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

Construct a [CheckPointOutputManager](#) object.

Parameters

<i>in</i>	<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 321 of file checkpoint_output_manager.cc.

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

8.4.2.2 jeod::CheckPointOutputManager::CheckPointOutputManager (const CheckPointOutputManager &) [private]

Not implemented.

8.4.3 Member Function Documentation

8.4.3.1 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 257 of file checkpoint_output_manager.hh.

Referenced by create_trick_section_writer(), and jeod::BasicJeodTrickSimInterface::get_checkpoint_writer_internal().

8.4.3.2 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 355 of file checkpoint_output_manager.cc.

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

8.4.3.3 SectionedOutputStream jeod::CheckPointOutputManager::create_trick_section_writer (void) [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 388 of file checkpoint_output_manager.cc.

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

8.4.3.4 bool jeod::CheckPointOutputManager::deregister_writer (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 425 of file checkpoint_output_manager.cc.

References [current_writer](#).

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

8.4.3.5 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 273 of file checkpoint_output_manager.hh.

References [current_writer](#).

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

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

Conversion to boolean.

Returns

False if object is OK.

Definition at line 265 of file checkpoint_output_manager.hh.

References [is_open](#), and [stream](#).

8.4.3.7 CheckPointOutputManager& jeod::CheckPointOutputManager::operator= (const CheckPointOutputManager &) [private]

Not implemented.

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

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

Returns

True => success.

Parameters

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

Definition at line 406 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** `friend class MemoryManagerWrapper [friend]`

Definition at line 244 of file checkpoint_output_manager.hh.

8.4.5 Field Documentation**8.4.5.1** `SectionedOutputStream* jeod::CheckPointOutputManager::current_writer [private]`

The writer that currently is active.

`trick_io(**)`

Definition at line 309 of file checkpoint_output_manager.hh.

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

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

The name of the checkpoint file.

`trick_io(**)`

Definition at line 314 of file checkpoint_output_manager.hh.

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

8.4.5.3 `bool jeod::CheckPointOutputManager::is_open [private]`

Is the checkpoint file open?

`trick_io(**)`

Definition at line 329 of file checkpoint_output_manager.hh.

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

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

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

`trick_io(**)`

Definition at line 324 of file `checkpoint_output_manager.hh`.

Referenced by `create_section_writer()`.

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

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

`trick_io(**)`

Definition at line 319 of file `checkpoint_output_manager.hh`.

Referenced by `create_section_writer()`.

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

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

`trick_io(**)`

Definition at line 304 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, const 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 216 of file `trick_memory_interface.hh`.

8.5.2 Constructor & Destructor Documentation

8.5.2.1 `jeod::JeodTrickMemoryInterface::ContainerListEntry::ContainerListEntry (const void * parent, const JeodMemoryTypeDescriptor & tdesc, const 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 246 of file `trick_memory_interface.hh`.

8.5.3 Field Documentation

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

The container itself.

`trick_units(-)`

Definition at line 236 of file `trick_memory_interface.hh`.

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

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

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

`trick_units(-)`

Definition at line 231 of file `trick_memory_interface.hh`.

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

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

The object that contains the container.

`trick_units(-)`

Definition at line 221 of file `trick_memory_interface.hh`.

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

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

Type description of the object that contains the container.

`trick_units(-)`

Definition at line 226 of file `trick_memory_interface.hh`.

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

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

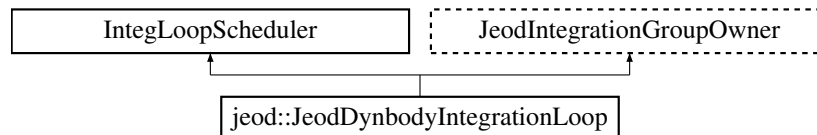
- [trick_memory_interface.hh](#)

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.
- virtual [~JeodDynbodyIntegrationLoop](#) ()
JeodDynbodyIntegrationLoop destructor.
- void [initialize_integ_loop](#) (void)
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.
- virtual void [update_integration_group](#) (JeodIntegrationGroup &group)
Update the provided integration group, which must be the integration group contained within this integration loop object.
- virtual int [add_sim_object](#) (Trick::SimObject &sim_obj)
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.
- virtual int [remove_sim_object](#) (Trick::SimObject &sim_obj)
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](#) (void)
Compute the gravitational accelerations of each dynamic body that is integrated by this integration loop.
- virtual void [collect_derivatives](#) (void)
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](#) (void)
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.
- virtual int [integrate_dt](#) (double beg_sim_time, double del_sim_time)
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.

Private Member Functions

- [JeodDynbodyIntegrationLoop](#) (const [JeodDynbodyIntegrationLoop](#) &)
< Deleted.
- [JeodDynbodyIntegrationLoop](#) & operator= (const [JeodDynbodyIntegrationLoop](#) &)

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 101 of file `trick_dynbody_integ_loop.hh`.

8.6.2 Constructor & Destructor Documentation

8.6.2.1 `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 60 of file `trick_dynbody_integ_loop.cc`.

8.6.2.2 `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 <code>integ_loop</code> 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 76 of file `trick_dynbody_integ_loop.cc`.

References `add_sim_object()`, `jeod::SimInterfaceMessages::integration_error`, and `loop_sim_object`.

8.6.2.3 jeod::JeodDynbodyIntegrationLoop::~JeodDynbodyIntegrationLoop (void) [virtual]

[JeodDynbodyIntegrationLoop](#) destructor.

Definition at line 111 of file `trick_dynbody_integ_loop.cc`.

References `integ_group`.

8.6.2.4 jeod::JeodDynbodyIntegrationLoop::JeodDynbodyIntegrationLoop (const JeodDynbodyIntegrationLoop &) [private]

< Deleted.

Deleted.

8.6.3 Member Function Documentation

8.6.3.1 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 189 of file `trick_dynbody_integ_loop.cc`.

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

8.6.3.2 int jeod::JeodDynbodyIntegrationLoop::add_sim_object (Trick::SimObject & sim_obj) [virtual]

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 290 of file `trick_dynbody_integ_loop.cc`.

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

Referenced by `JeodDynbodyIntegrationLoop()`.

8.6.3.3 `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 334 of file `trick_dynbody_integ_loop.cc`.

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

8.6.3.4 `void jeod::JeodDynbodyIntegrationLoop::add_sim_object_bodies (void)` `[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 353 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 `virtual void jeod::JeodDynbodyIntegrationLoop::collect_derivatives (void)` `[inline]`, `[virtual]`

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

Definition at line 252 of file `trick_dynbody_integ_loop.hh`.

References `integ_group`.

8.6.3.6 `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 151 of file `trick_dynbody_integ_loop.cc`.

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

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

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

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

Definition at line 242 of file `trick_dynbody_integ_loop.hh`.

References `dyn_manager`, `gravity_manager`, and `integ_group`.

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

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

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 227 of file `trick_dynbody_integ_loop.cc`.

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

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

8.6.3.11 `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 210 of file `trick_dynbody_integ_loop.cc`.

References `integ_group`.

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

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 312 of file trick_dynbody_integ_loop.cc.

References dyn_manager, and remove_sim_object_bodies().

8.6.3.13 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 374 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 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 deriv_ephem_update.
------------	-----------------------------------

Definition at line 262 of file trick_dynbody_integ_loop.hh.

References deriv_ephem_update, and integ_group.

8.6.3.15 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 219 of file trick_dynbody_integ_loop.cc.

References time_manager.

8.6.3.16 void jeod::JeodDynbodyIntegrationLoop::update_integration_group (JeodIntegrationGroup & *group*)
[virtual]

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

8.6.4.2 `friend class InputProcessor` `[friend]`

Definition at line 106 of file `trick_dynbody_integ_loop.hh`.

8.6.5 Field Documentation

8.6.5.1 `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 372 of file `trick_dynbody_integ_loop.hh`.

Referenced by `initialize_integ_loop()`, and `set_deriv_ephem_update()`.

8.6.5.2 `DynManager* jeod::JeodDynbodyIntegrationLoop::dyn_manager` `[protected]`

The JEOD dynamics manager.

`trick_units(-)`

Definition at line 332 of file `trick_dynbody_integ_loop.hh`.

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

8.6.5.3 `GravityManager* jeod::JeodDynbodyIntegrationLoop::gravity_manager` `[protected]`

The gravity model manager.

`trick_units(-)`

Definition at line 342 of file `trick_dynbody_integ_loop.hh`.

Referenced by `gravitation()`.

8.6.5.4 `er7_utils::IntegratorConstructor** jeod::JeodDynbodyIntegrationLoop::integ_constructor` `[protected]`

Handle to the integration constructor used to create integrators.

`trick_units(-)`

Definition at line 352 of file `trick_dynbody_integ_loop.hh`.

Referenced by `initialize_integ_loop()`.

8.6.5.5 `DynamicsIntegrationGroup* jeod::JeodDynbodyIntegrationLoop::integ_group` [protected]

The integration group that performs the integration.

`trick_units(-)`

Definition at line 363 of file `trick_dynbody_integ_loop.hh`.

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

8.6.5.6 `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 358 of file `trick_dynbody_integ_loop.hh`.

Referenced by `initialize_integ_loop()`.

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

Dummy integration interface; needed by the `integ_group`.

`trick_units(-)`

Definition at line 347 of file `trick_dynbody_integ_loop.hh`.

Referenced by `initialize_integ_loop()`.

8.6.5.8 `Trick::SimObject* jeod::JeodDynbodyIntegrationLoop::loop_sim_object` [protected]

The simulation object that contains this integration loop object.

`trick_units(-)`

Definition at line 327 of file `trick_dynbody_integ_loop.hh`.

Referenced by `JeodDynbodyIntegrationLoop()`.

8.6.5.9 `TimeManager* jeod::JeodDynbodyIntegrationLoop::time_manager` [protected]

The JEOD time manager.

`trick_units(-)`

Definition at line 337 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:

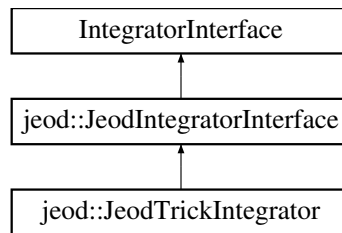
- [trick_dynbody_integ_loop.hh](#)
- [trick_dynbody_integ_loop.cc](#)

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

- virtual [~JeodIntegratorInterface](#) ()
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 55 of file jeod_integrator_interface.hh.

8.7.2 Constructor & Destructor Documentation

8.7.2.1 virtual jeod::JeodIntegratorInterface::~~JeodIntegratorInterface () [inline],[virtual]

Destructor.

Definition at line 67 of file jeod_integrator_interface.hh.

8.7.3 Member Function Documentation

8.7.3.1 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 `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 `void init_attrjeod__JeodIntegratorInterface () [friend]`

8.7.4.2 `friend class InputProcessor [friend]`

Definition at line 56 of file `jeod_integrator_interface.hh`.

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

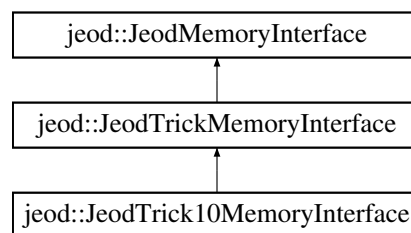
- [jeod_integrator_interface.hh](#)

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 constructor.
- `virtual ~JeodMemoryInterface` ()
Destructor.
- [JeodMemoryInterface](#) (const [JeodMemoryInterface](#) &)
Copy constructor.
- [JeodMemoryInterface](#) & operator= (const [JeodMemoryInterface](#) &)
Assignment operator.
- `virtual struct ATTRIBUTES_tag * find_attributes (const std::string &type_name) const =0`
Find the attributes for a given class name.
- `virtual 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](#) (void) 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 char *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 char *elem_name, JeodCheckpointable &checkpointable)=0
Deregister a JeodCheckpointable object with the simulation engine.
- virtual bool [is_checkpoint_restart_supported](#) (void) 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 58 of file memory_interface.hh.

8.8.2 Constructor & Destructor Documentation

8.8.2.1 jeod::JeodMemoryInterface::JeodMemoryInterface (void)

Default constructor.

Definition at line 45 of file memory_interface.cc.

8.8.2.2 jeod::JeodMemoryInterface::~~JeodMemoryInterface (void) [virtual]

Destructor.

Definition at line 55 of file memory_interface.cc.

8.8.2.3 jeod::JeodMemoryInterface::JeodMemoryInterface (const JeodMemoryInterface & src) [explicit]

Copy constructor.

Parameters

in	<i>src</i>	Item to be copied
----	------------	-------------------

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

8.8.3 Member Function Documentation

8.8.3.1 `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 `virtual void jeod::JeodMemoryInterface::deregister_container (const void * container, const JeodMemoryTypeDescriptor & container_type, const char * 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 `virtual 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 `virtual struct ATTRIBUTES_tag* jeod::JeodMemoryInterface::find_attributes (const std::type_info & data_type) const [pure virtual]`

Find the attributes for a given class.

Parameters

<i>in</i>	<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 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

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

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

```
8.8.3.6 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](#).

8.8.3.7 `virtual bool jeod::JeodMemoryInterface::is_checkpoint_restart_supported (void) 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 `JeodMemoryInterface & jeod::JeodMemoryInterface::operator= (const JeodMemoryInterface & src)`

Assignment operator.

Returns

*this

Parameters

in	<i>src</i>	Item to be copied
----	------------	-------------------

Definition at line 79 of file `memory_interface.cc`.

8.8.3.9 `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 `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 `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 `virtual void jeod::JeodMemoryInterface::register_container (const void * container, const JeodMemoryTypeDescriptor & container_type, const char * 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 `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 `virtual struct ATTRIBUTES_tag jeod::JeodMemoryInterface::void_pointer_attributes (void) 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 `void init_attrjeod__JeodMemoryInterface () [friend]`

8.8.4.2 `friend class InputProcessor [friend]`

Definition at line 60 of file `memory_interface.hh`.

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

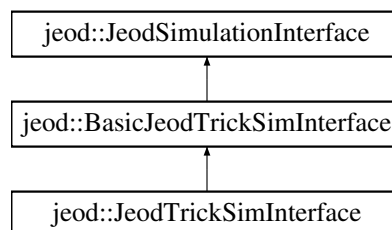
- [memory_interface.hh](#)
- [memory_interface.cc](#)

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.
- virtual void `configure` (const `JeodSimulationInterfaceInit` &config)
Configure a [JeodSimulationInterface](#) object.
- `Mode get_mode` (void) 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](#) (void)
Create a simulation integrator interface object.
- static double [get_job_cycle](#) (void)
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](#) (void)
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](#) (void)=0
Create an integration interface object.
- virtual double [get_job_cycle_internal](#) (void)=0
Get the simulation cycle time of the currently executing function.
- virtual [JeodMemoryInterface](#) & [get_memory_interface_internal](#) (void)=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](#)
The mode in which the simulation interface is operating.
- [Mode](#) [saved_mode](#)
The mode prior to a checkpoint or restart process.

Static Protected Attributes

- static [JeodSimulationInterface](#) * [sim_interface](#) = NULL
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.

Private Member Functions

- [JeodSimulationInterface](#) (const [JeodSimulationInterface](#) &)
Not implemented.
- [JeodSimulationInterface](#) & [operator=](#) (const [JeodSimulationInterface](#) &)
Not implemented.

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 [JeodMemoryManager](#) object; destruction must be performed in reverse order.

Definition at line 105 of file [simulation_interface.hh](#).

8.9.2 Member Enumeration Documentation

8.9.2.1 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 115 of file [simulation_interface.hh](#).

8.9.3 Constructor & Destructor Documentation

8.9.3.1 [jeod::JeodSimulationInterface::JeodSimulationInterface](#) (void)

Construct a [JeodSimulationInterface](#) object.

Definition at line 75 of file [simulation_interface.cc](#).

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

8.9.3.2 [jeod::JeodSimulationInterface::~~JeodSimulationInterface](#) (void) [virtual]

Destruct a [JeodSimulationInterface](#) object.

Definition at line 95 of file [simulation_interface.cc](#).

References [sim_interface](#).

8.9.3.3 `jeod::JeodSimulationInterface::JeodSimulationInterface (const JeodSimulationInterface &)` `[private]`

Not implemented.

8.9.4 Member Function Documentation

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

Configure a [JeodSimulationInterface](#) object.

Parameters

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

Definition at line 109 of file `simulation_interface.cc`.

References `jeod::JeodSimulationInterfaceInit::memory_debug_level`, `jeod::JeodSimulationInterfaceInit::message_suppress_id`, `jeod::JeodSimulationInterfaceInit::message_suppress_location`, and `jeod::JeodSimulationInterfaceInit::message_suppression_level`.

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

Create a simulation integrator interface object.

Returns

Constructed IntegratorInterface object.

Definition at line 131 of file `simulation_interface.cc`.

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

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

Translate the given symbolic name to an address.

Returns

Address

Parameters

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

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

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

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

8.9.4.5 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

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

Definition at line 250 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 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 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

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

Definition at line 271 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 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 `double jeod::JeodSimulationInterface::get_job_cycle (void) [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 154 of file `simulation_interface.cc`.

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

8.9.4.10 `virtual double jeod::JeodSimulationInterface::get_job_cycle_internal (void) [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 `JeodMemoryInterface & jeod::JeodSimulationInterface::get_memory_interface (void) [static]`

Get the interface with the simulation memory model.

Returns

Memory interface

Definition at line 177 of file `simulation_interface.cc`.

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

8.9.4.12 `virtual JeodMemoryInterface& jeod::JeodSimulationInterface::get_memory_interface_internal (void) [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 Mode jeod::JeodSimulationInterface::get_mode (void) const [inline]

Get the current mode.

Definition at line 217 of file simulation_interface.hh.

References mode.

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

8.9.4.14 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	addr	Address
in	tdesc	Descriptor

Definition at line 199 of file simulation_interface.cc.

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

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

8.9.4.15 JeodSimulationInterface& jeod::JeodSimulationInterface::operator= (const JeodSimulationInterface &) [private]

Not implemented.

8.9.4.16 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	new_mode	New mode
----	----------	----------

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

Definition at line 298 of file simulation_interface.cc.

References Checkpoint, Construction, Dead, jeod::SimInterfaceMessages::implementation_error, mode, NumModes, 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 `void init_attrjeod__JeodSimulationInterface () [friend]`

8.9.5.2 `friend class InputProcessor [friend]`

Definition at line 106 of file `simulation_interface.hh`.

8.9.6 Field Documentation

8.9.6.1 **Mode** `jeod::JeodSimulationInterface::mode [protected]`

The mode in which the simulation interface is operating.

`trick_units(-)`

Definition at line 280 of file `simulation_interface.hh`.

Referenced by `get_mode()`, and `set_mode()`.

8.9.6.2 **Mode** `jeod::JeodSimulationInterface::saved_mode [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 286 of file `simulation_interface.hh`.

Referenced by `set_mode()`.

8.9.6.3 **JeodSimulationInterface** * `jeod::JeodSimulationInterface::sim_interface = NULL [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 237 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 54 of file [simulation_interface.hh](#).

8.10.2 Constructor & Destructor Documentation

8.10.2.1 `jeod::JeodSimulationInterfaceInit::JeodSimulationInterfaceInit (void)`

Construct a [JeodSimulationInterfaceInit](#) object.

Definition at line 53 of file [simulation_interface.cc](#).

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

8.10.3 Field Documentation

8.10.3.1 `unsigned int jeod::JeodSimulationInterfaceInit::memory_debug_level`

Specifies the memory manager's debug level; see [JeodMemoryManager::debug_level](#) for details.

Default value: 0.trick_units(—)

Definition at line 94 of file [simulation_interface.hh](#).

Referenced by [jeod::JeodSimulationInterface::configure\(\)](#), and [JeodSimulationInterfaceInit\(\)](#).

8.10.3.2 `bool jeod::JeodSimulationInterfaceInit::message_suppress_id`

Specifies the message handler's [suppress_id](#) flag; see [MessageHandler::suppression_id](#) for details.

Default value: false.trick_units(—)

Definition at line 80 of file [simulation_interface.hh](#).

Referenced by [jeod::JeodSimulationInterface::configure\(\)](#).

8.10.3.3 bool jeod::JeodSimulationInterfaceInit::message_suppress_location

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

Default value: false.trick_units(-)

Definition at line 87 of file simulation_interface.hh.

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

8.10.3.4 unsigned int jeod::JeodSimulationInterfaceInit::message_suppression_level

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

Default value: MessageHandler::Warning (warnings and non-fatal errors).trick_units(-)

Definition at line 73 of file simulation_interface.hh.

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

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

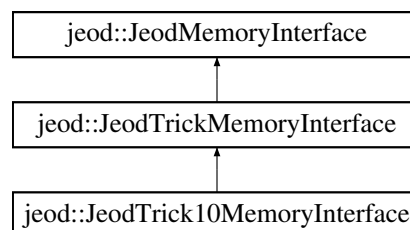
- [simulation_interface.hh](#)
- [simulation_interface.cc](#)

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.
- virtual [~JeodTrick10MemoryInterface](#) ()
Destruct a [JeodTrick10MemoryInterface](#) object.
- virtual void [register_container](#) (const void *owner, const JeodMemoryTypeDescriptor &owner_type, const char *elem_name, JeodCheckpointable &container)
Register the checkpointable object with [Trick](#).
- virtual void [deregister_container](#) (const void *owner, const JeodMemoryTypeDescriptor &owner_type, const char *elem_name, JeodCheckpointable &container)
Revoke the registrations performed by register_container.
- virtual const std::string [get_name_at_address](#) (const void *addr, const JeodMemoryTypeDescriptor *tdesc) const
Get the simulation name, if any, associated with the address.

- virtual void * [get_address_at_name](#) (const std::string &name) const
Get the address, if any, that corresponds to the given name.
- virtual bool [is_checkpoint_restart_supported](#) (void) const
The Trick10 memory interface supports 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](#) (void)
Dump the checkpointable objects to the checkpoint file.
- virtual void [restore_containers](#) (void)
Restore the checkpointable objects from the checkpoint file.
- virtual void [checkpoint_allocations](#) (void)
Dump the allocation information to the checkpoint file.
- virtual void [restore_allocations](#) (JeodMemoryManager &memory_manager)
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.

Private Member Functions

- [JeodTrick10MemoryInterface](#) (const [JeodTrick10MemoryInterface](#) &)
Not implemented.
- [JeodTrick10MemoryInterface](#) & operator= (const [JeodTrick10MemoryInterface](#) &)
Not implemented.

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 77 of file [trick10_memory_interface.hh](#).

8.11.2 Constructor & Destructor Documentation

8.11.2.1 jeod::JeodTrick10MemoryInterface::JeodTrick10MemoryInterface (void)

Construct a [JeodTrick10MemoryInterface](#) object.

Definition at line 69 of file `trick10_memory_interface.cc`.

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

8.11.2.2 jeod::JeodTrick10MemoryInterface::~JeodTrick10MemoryInterface (void) [virtual]

Destruct a [JeodTrick10MemoryInterface](#) object.

Definition at line 91 of file `trick10_memory_interface.cc`.

8.11.2.3 jeod::JeodTrick10MemoryInterface::JeodTrick10MemoryInterface (const JeodTrick10MemoryInterface &) [private]

Not implemented.

8.11.3 Member Function Documentation

8.11.3.1 void jeod::JeodTrick10MemoryInterface::checkpoint_allocations (void) [virtual]

Dump the allocation information to the checkpoint file.

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

Definition at line 438 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 void jeod::JeodTrick10MemoryInterface::checkpoint_containers (void) [virtual]

Dump the checkpointable objects to the checkpoint file.

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

Definition at line 250 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 void jeod::JeodTrick10MemoryInterface::deregister_container (const void * owner, const JeodMemoryTypeDescriptor & owner_type, const char * elem_name, JeodCheckpointable & container) [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

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

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

Definition at line 152 of file `trick_memory_interface_chkpnt.cc`.

References `jeod::JeodTrickMemoryInterface::container_list`, `jeod::JeodTrickMemoryInterface::ContainerListEntry::elem_name`, `jeod::SimInterfaceMessages::interface_error`, `jeod::JeodTrickMemoryInterface::ContainerListEntry::owner`, and `jeod::JeodTrickMemoryInterface::ContainerListEntry::owner_type`.

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

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

Returns

Name of the address, if any

Parameters

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

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

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

References `translate_name_to_addr()`.

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

Construct the identifier for a checkpointable object.

Returns

Container ID

Parameters

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

Definition at line 207 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 `const std::string jeod::JeodTrick10MemoryInterface::get_name_at_address (const void * addr, const JeodMemoryTypeDescriptor * tdesc) const` `[virtual]`

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

Returns

Name of the address, if any

Parameters

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

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

Definition at line 90 of file `trick_memory_interface_xlate.cc`.

References `translate_addr_to_name()`, and `translate_name_to_addr()`.

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

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

Returns

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

Units:

Parameters

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

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

Definition at line 69 of file `trick_memory_interface_xlate.cc`.

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

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

The Trick10 memory interface supports checkpoint/restart.

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

Definition at line 117 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 `JeodTrick10MemoryInterface& jeod::JeodTrick10MemoryInterface::operator= (const JeodTrick10MemoryInterface &)` `[private]`

Not implemented.

8.11.3.10 `void jeod::JeodTrick10MemoryInterface::register_container (const void * owner, const JeodMemoryTypeDescriptor & owner_type, const char * elem_name, JeodCheckpointable & container)` `[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

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

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

References `jeod::JeodTrickMemoryInterface::container_list`, `jeod::JeodTrickMemoryInterface::ContainerListEntry::elem_name`, `get_container_id()`, `jeod::SimInterfaceMessages::interface_error`, `jeod::JeodTrickMemoryInterface::ContainerListEntry::owner`, and `jeod::JeodTrickMemoryInterface::ContainerListEntry::owner_type`.

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

Restore the allocated data per the checkpoint file.

Parameters

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

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

Definition at line 508 of file `trick_memory_interface_chkpt.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 `void jeod::JeodTrick10MemoryInterface::restore_containers (void)` `[virtual]`

Restore the checkpointable objects from the checkpoint file.

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

Definition at line 319 of file `trick_memory_interface_chkpt.cc`.

References `jeod::SectionedInputStream::activate()`, `jeod::JeodTrickMemoryInterface::ContainerListEntry::container`, `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 `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 158 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 `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 198 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 `void init_attrjeod__JeodTrick10MemoryInterface ()` `[friend]`

8.11.4.2 `friend class InputProcessor` `[friend]`

Definition at line 79 of file `trick10_memory_interface.hh`.

8.11.5 Field Documentation

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

Trick checkpoint agent.

trick_io(**)

Definition at line 151 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:

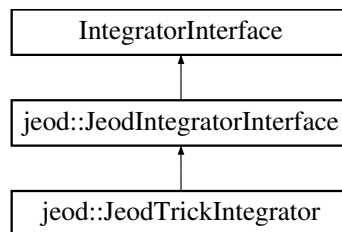
- [trick10_memory_interface.hh](#)
- [trick10_memory_interface.cc](#)
- [trick_memory_interface_chkpnt.cc](#)
- [trick_memory_interface_xlate.cc](#)

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.
- virtual [~JeodTrickIntegrator](#) ()
Destructor.
- virtual
er7_utils::Integration::Technique [interpret_integration_type](#) (int integ_technique) const
Interpret the integration technique.
- virtual Trick::Integrator* [get_integrator](#) ()
Get the simulation engine's integrator.
- virtual double [get_dt](#) () const
Get the integration cycle time step.
- virtual bool [get_first_step_derivs_flag](#) () const
Get the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle.
- virtual void [set_first_step_derivs_flag](#) (bool value)
Set the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle.
- virtual void [reset_first_step_derivs_flag](#) ()
Reset the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle.
- virtual void [restore_first_step_derivs_flag](#) ()
Restore the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle to its value prior to the most recent call to reset_first_step_derivs_flag.

- virtual void [set_step_number](#) (unsigned int stepno)
Set the step number within an integration cycle.
- virtual void [set_time](#) (double sim_time)
Update the time model given the simulation time.

Private Member Functions

- [JeodTrickIntegrator](#) (const [JeodTrickIntegrator](#) &)
Not implemented.
- [JeodTrickIntegrator](#) & [operator=](#) (const [JeodTrickIntegrator](#) &)
Not implemented.

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 87 of file [jeod_trick_integrator.hh](#).

8.12.2 Constructor & Destructor Documentation

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

Default constructor.

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

8.12.2.2 `virtual jeod::JeodTrickIntegrator::~~JeodTrickIntegrator () [inline],[virtual]`

Destructor.

Definition at line 107 of file [jeod_trick_integrator.hh](#).

8.12.2.3 `jeod::JeodTrickIntegrator::JeodTrickIntegrator (const JeodTrickIntegrator &) [private]`

Not implemented.

8.12.3 Member Function Documentation

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

Get the integration cycle time step.

Returns

Simulation time delta t, in seconds

Definition at line 133 of file `jeod_trick_integrator.hh`.

References `trick_integrator`.

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

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 143 of file `jeod_trick_integrator.hh`.

References `trick_integrator`.

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

Get the simulation engine's integrator.

Returns

Pointer to the simulation engine's integrator.

Implements [jeod::JeodIntegratorInterface](#).

Definition at line 124 of file `jeod_trick_integrator.hh`.

References `trick_integrator`.

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

Interpret the integration technique.

Implements [jeod::JeodIntegratorInterface](#).

Definition at line 112 of file `jeod_trick_integrator.hh`.

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

Not implemented.

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

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 164 of file jeod_trick_integrator.hh.

References `default_first_step_deriv`, and `trick_integrator`.

8.12.3.7 virtual void jeod::JeodTrickIntegrator::restore_first_step_derivs_flag () [inline],[virtual]

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 175 of file jeod_trick_integrator.hh.

References `default_first_step_deriv`, and `trick_integrator`.

8.12.3.8 virtual void jeod::JeodTrickIntegrator::set_first_step_derivs_flag (bool *value*) [inline],[virtual]

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 153 of file jeod_trick_integrator.hh.

References `trick_integrator`.

8.12.3.9 virtual void jeod::JeodTrickIntegrator::set_step_number (unsigned int *stepno*) [inline],[virtual]

Set the step number within an integration cycle.

Parameters

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

Definition at line 184 of file jeod_trick_integrator.hh.

References `trick_integrator`.

8.12.3.10 virtual void jeod::JeodTrickIntegrator::set_time (double *sim_time*) [inline],[virtual]

Update the time model given the simulation time.

Parameters

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

Definition at line 193 of file jeod_trick_integrator.hh.

References `trick_integrator`.

8.12.4 Friends And Related Function Documentation

8.12.4.1 void init_attrjeod__JeodTrickIntegrator () [friend]

8.12.4.2 friend class InputProcessor [friend]

Definition at line 88 of file jeod_trick_integrator.hh.

8.12.5 Field Documentation

8.12.5.1 `bool jeod::JeodTrickIntegrator::default_first_step_deriv` [private]

Default value of `trick_integrator.first_step_deriv`.

`trick_units(-)`

Definition at line 211 of file `jeod_trick_integrator.hh`.

Referenced by `reset_first_step_derivs_flag()`, and `restore_first_step_derivs_flag()`.

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

[Trick](#) integration structure.

`trick_units(-)`

Definition at line 206 of file `jeod_trick_integrator.hh`.

Referenced by `get_dt()`, `get_first_step_derivs_flag()`, `get_integrator()`, `reset_first_step_derivs_flag()`, `restore_first_step_derivs_flag()`, `set_first_step_derivs_flag()`, `set_step_number()`, and `set_time()`.

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

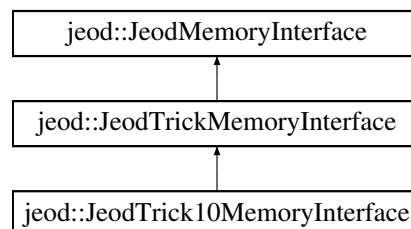
- [jeod_trick_integrator.hh](#)

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.
- virtual [~JeodTrickMemoryInterface](#) ()
[JeodTrickMemoryInterface](#) destructor.
- 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.
- virtual struct ATTRIBUTES_tag * [find_attributes](#) (const std::string &type_name) const
Find the attributes for a class in the symbol table.
- virtual struct ATTRIBUTES_tag * [find_attributes](#) (const std::type_info &data_type) const
Find the attributes for a class in the symbol table.
- virtual struct ATTRIBUTES_tag [primitive_attributes](#) (const std::type_info &data_type) const
Create an attributes structure that represents a primitive type.
- virtual struct ATTRIBUTES_tag [pointer_attributes](#) (const struct ATTRIBUTES_tag &target_attr) const
Create an attributes structure that represents a pointer type.
- virtual struct ATTRIBUTES_tag [void_pointer_attributes](#) () const
Create an attributes structure that represents a void pointer.*
- virtual struct ATTRIBUTES_tag [structure_attributes](#) (const struct ATTRIBUTES_tag *target_attr, std::size_t target_size) const
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)
Register newly allocated memory with Trick.
- virtual void [deregister_allocation](#) (const void *addr, const JeodMemoryItem &item, const JeodMemoryTypeDescriptor &tdesc, const char *file, unsigned int line)
Delete Trick information about some pointer – but not the pointer itself.
- virtual void [register_container](#) (const void *owner, const JeodMemoryTypeDescriptor &owner_type, const char *elem_name, JeodCheckpointable &container)
Register the checkpointable object with Trick.
- virtual void [deregister_container](#) (const void *owner, const JeodMemoryTypeDescriptor &owner_type, const char *elem_name, JeodCheckpointable &container)
Revoke the registrations performed by register_container.
- virtual const std::string [get_name_at_address](#) (const void *addr, const JeodMemoryTypeDescriptor *tdesc) const
Stubbed-out implementation of get_name_at_address for Trick implementations that do not fully support JEOD checkpoint/restart requirements.
- virtual void * [get_address_at_name](#) (const std::string &name) const
Stubbed-out implementation of get_address_at_name for Trick implementations that do not fully support JEOD checkpoint/restart requirements.
- virtual bool [is_checkpoint_restart_supported](#) () const
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

- `typedef std::map< uint32_t, AllocationMapEntry > AllocationMap`
Maps JEOD-allocated data names to (type, size) pairs.
- `typedef std::list< ContainerListEntry > ContainerList`
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`
Prefix used for constructing a unique name for JEOD-allocated memory.
- `const uint32_t id_length`
Number of digits in the numeric part of the unique identifier.
- `JeodSimulationInterface::Mode mode`
Simulation interface mode.

Private Member Functions

- `JeodTrickMemoryInterface (const JeodTrickMemoryInterface &)`
Not implemented.
- `JeodTrickMemoryInterface & operator= (const JeodTrickMemoryInterface &)`
Not implemented.

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 66 of file `trick_memory_interface.hh`.

8.13.2 Member Typedef Documentation

8.13.2.1 `typedef std::map<uint32_t, AllocationMapEntry> jeod::JeodTrickMemoryInterface::AllocationMap [protected]`

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

Definition at line 301 of file `trick_memory_interface.hh`.

8.13.2.2 `typedef std::list<ContainerListEntry> jeod::JeodTrickMemoryInterface::ContainerList`
`[protected]`

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

Definition at line 306 of file `trick_memory_interface.hh`.

8.13.3 Constructor & Destructor Documentation

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

[JeodTrickMemoryInterface](#) default constructor.

Definition at line 62 of file `trick_memory_interface.cc`.

References `dlhandle`, and `jeod::SimInterfaceMessages::implementation_error`.

8.13.3.2 `jeod::JeodTrickMemoryInterface::~~JeodTrickMemoryInterface () [virtual]`

[JeodTrickMemoryInterface](#) destructor.

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

References `dlhandle`.

8.13.3.3 `jeod::JeodTrickMemoryInterface::JeodTrickMemoryInterface (const JeodTrickMemoryInterface &)`
`[private]`

Not implemented.

8.13.4 Member Function Documentation

8.13.4.1 `virtual void jeod::JeodTrickMemoryInterface::checkpoint_allocations (void) [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 195 of file `trick_memory_interface.hh`.

8.13.4.2 `virtual void jeod::JeodTrickMemoryInterface::checkpoint_containers (void) [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 179 of file `trick_memory_interface.hh`.

8.13.4.3 `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 110 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 `void jeod::JeodTrickMemoryInterface::deregister_allocation (const void * addr, const JeodMemoryItem & item, const JeodMemoryTypeDescriptor & tdesc, const char * file, unsigned int line) [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 145 of file `trick_memory_interface_alloc.cc`.

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

8.13.4.5 `void jeod::JeodTrickMemoryInterface::deregister_container (const void * owner, const JeodMemoryTypeDescriptor & owner_type, const char * elem_name, JeodCheckpointable & container) [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](#).

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

Definition at line 155 of file `trick_memory_interface.cc`.

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

Find the attributes for a class in the symbol table.

Returns

Found attributes

Parameters

<i>in</i>	<i>type_name</i>	Demangled type name
-----------	------------------	---------------------

Implements [jeod::JeodMemoryInterface](#).

Definition at line 63 of file `trick_memory_interface_attrib.cc`.

References `dlhandle`, and `jeod::SimInterfaceMessages::interface_error`.

Referenced by `find_attributes()`.

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

Find the attributes for a class in the symbol table.

Returns

Found attributes

Parameters

<i>in</i>	<i>data_type</i>	Data type descriptor
-----------	------------------	----------------------

Implements [jeod::JeodMemoryInterface](#).

Definition at line 94 of file `trick_memory_interface_attrib.cc`.

References `find_attributes()`.

8.13.4.8 `void * jeod::JeodTrickMemoryInterface::get_address_at_name (const std::string & name) const` `[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](#).

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

Definition at line 189 of file `trick_memory_interface.cc`.

8.13.4.9 `const std::string jeod::JeodTrickMemoryInterface::get_name_at_address (const void * addr, const JeodMemoryTypeDescriptor * tdesc) const` `[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](#).

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

Definition at line 172 of file `trick_memory_interface.cc`.

8.13.4.10 `virtual const std::string jeod::JeodTrickMemoryInterface::get_trick_checkpoint_file (bool checkpoint)`
`[inline], [virtual]`

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

Parameters

<i>in</i>	<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 167 of file `trick_memory_interface.hh`.

8.13.4.11 `virtual bool jeod::JeodTrickMemoryInterface::is_checkpoint_restart_supported (void) const` `[inline],`
`[virtual]`

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

Implements [jeod::JeodMemoryInterface](#).

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

Definition at line 157 of file `trick_memory_interface.hh`.

8.13.4.12 `JeodTrickMemoryInterface& jeod::JeodTrickMemoryInterface::operator= (const`
`JeodTrickMemoryInterface &)` `[private]`

Not implemented.

8.13.4.13 `struct ATTRIBUTES_tag jeod::JeodTrickMemoryInterface::pointer_attributes (const struct ATTRIBUTES_tag &`
`target_attr) const` `[virtual]`

Create an attributes structure that represents a pointer type.

Returns

Constructed pointer attributes.

Parameters

<i>in</i>	<i>target_attr</i>	Pointed-to type attributes.
-----------	--------------------	-----------------------------

Implements [jeod::JeodMemoryInterface](#).

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

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

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

Create an attributes structure that represents a primitive type.

Returns

Constructed attributes.

Parameters

<i>in</i>	<i>data_type</i>	Data type descriptor
-----------	------------------	----------------------

Implements [jeod::JeodMemoryInterface](#).

Definition at line 108 of file `trick_memory_interface_attrib.cc`.

References `jeod::SimInterfaceMessages::interface_error`.

8.13.4.15 `bool jeod::JeodTrickMemoryInterface::register_allocation (const void * addr, const JeodMemoryItem & item, const JeodMemoryTypeDescriptor & tdesc, const char * file, unsigned int line) [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

<i>in</i>	<i>addr</i>	Allocated memory
<i>in</i>	<i>item</i>	Description of the memory
<i>in</i>	<i>tdesc</i>	Description of the type
<i>in</i>	<i>file</i>	Source file containing JEOD_ALLOC
<i>in</i>	<i>line</i>	Line number containing JEOD_ALLOC

Implements [jeod::JeodMemoryInterface](#).

Definition at line 82 of file `trick_memory_interface_alloc.cc`.

References `allocation_map`, `construct_identifier()`, `jeod::SimInterfaceMessages::interface_error`, and `trick_MM`.

8.13.4.16 `void jeod::JeodTrickMemoryInterface::register_container (const void * owner, const JeodMemoryTypeDescriptor & owner_type, const char * elem_name, JeodCheckpointable & container) [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](#).

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

Definition at line 134 of file `trick_memory_interface.cc`.

8.13.4.17 `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 204 of file `trick_memory_interface.hh`.

8.13.4.18 `virtual void jeod::JeodTrickMemoryInterface::restore_containers (void) [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 187 of file `trick_memory_interface.hh`.

8.13.4.19 `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 97 of file `trick_memory_interface.cc`.

References mode.

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

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

Create an attributes structure that represents a structured type.

Returns

Constructed structure attributes.

Parameters

in	<i>target_attr</i>	Return value from find_attributes.
in	<i>target_size</i>	Structure size.

Implements [jeod::JeodMemoryInterface](#).

Definition at line 287 of file trick_memory_interface_attrib.cc.

8.13.4.21 `struct ATTRIBUTES_tag jeod::JeodTrickMemoryInterface::void_pointer_attributes (void) const [virtual]`

Create an attributes structure that represents a void* pointer.

Returns

Constructed pointer attributes.

Implements [jeod::JeodMemoryInterface](#).

Definition at line 262 of file trick_memory_interface_attrib.cc.

8.13.5 Friends And Related Function Documentation

8.13.5.1 `void init_attrjeod__JeodTrickMemoryInterface () [friend]`

8.13.5.2 `friend class InputProcessor [friend]`

Definition at line 68 of file trick_memory_interface.hh.

8.13.6 Field Documentation

8.13.6.1 `AllocationMap jeod::JeodTrickMemoryInterface::allocation_map [protected]`

Map of allocated names to type info.

trick_io(**)

Definition at line 319 of file trick_memory_interface.hh.

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

8.13.6.2 `ContainerList jeod::JeodTrickMemoryInterface::container_list [protected]`

List of container checkpointables.

trick_io(**)

Definition at line 324 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 `void* jeod::JeodTrickMemoryInterface::dlhandle` [protected]

dlhandle, from dlopen.

trick_io(**)

Definition at line 314 of file trick_memory_interface.hh.

Referenced by find_attributes(), JeodTrickMemoryInterface(), and ~JeodTrickMemoryInterface().

8.13.6.4 `const uint32_t jeod::JeodTrickMemoryInterface::id_length` [protected]

Number of digits in the numeric part of the unique identifier.

trick_io(*o) trick_units(-)

Definition at line 334 of file trick_memory_interface.hh.

Referenced by construct_identifier().

8.13.6.5 `const std::string jeod::JeodTrickMemoryInterface::id_prefix` [protected]

Prefix used for constructing a unique name for JEOD-allocated memory.

trick_io(*o) trick_units(-)

Definition at line 329 of file trick_memory_interface.hh.

Referenced by construct_identifier().

8.13.6.6 `JeodSimulationInterface::Mode jeod::JeodTrickMemoryInterface::mode` [protected]

Simulation interface mode.

trick_units(-)

Definition at line 339 of file trick_memory_interface.hh.

Referenced by set_mode().

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

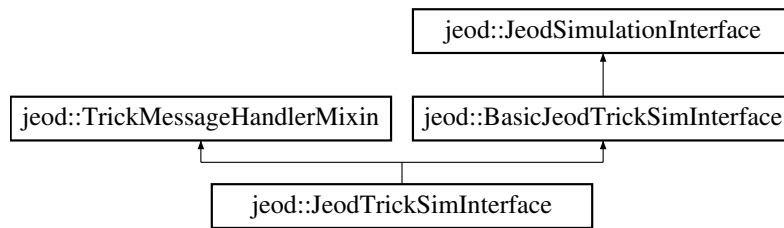
- [trick_memory_interface.hh](#)
- [trick_memory_interface.cc](#)
- [trick_memory_interface_alloc.cc](#)
- [trick_memory_interface_attrib.cc](#)

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.
- virtual [~JeodTrickSimInterface](#) ()
Destructor.

Private Member Functions

- [JeodTrickSimInterface](#) (const [JeodTrickSimInterface](#) &)
Not implemented.
- [JeodTrickSimInterface](#) & operator= (const [JeodTrickSimInterface](#) &)
Not implemented.

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 264 of file `trick_sim_interface.hh`.

8.14.2 Constructor & Destructor Documentation

8.14.2.1 `jeod::JeodTrickSimInterface::JeodTrickSimInterface () [inline], [explicit]`

Non-default constructor.

Definition at line 273 of file `trick_sim_interface.hh`.

8.14.2.2 `virtual jeod::JeodTrickSimInterface::~~JeodTrickSimInterface () [inline], [virtual]`

Destructor.

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

8.14.2.3 `jeod::JeodTrickSimInterface::JeodTrickSimInterface (const JeodTrickSimInterface &) [private]`

Not implemented.

8.14.3 Member Function Documentation

8.14.3.1 `JeodTrickSimInterface& jeod::JeodTrickSimInterface::operator= (const JeodTrickSimInterface &) [private]`

Not implemented.

8.14.4 Friends And Related Function Documentation

8.14.4.1 `void init_attrjeod__JeodTrickSimInterface () [friend]`

8.14.4.2 `friend class InputProcessor [friend]`

Definition at line 266 of file `trick_sim_interface.hh`.

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

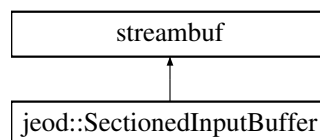
- [trick_sim_interface.hh](#)

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](#) ()
Destructor.
- `bool operator! () const`
Conversion to boolean.

Private Member Functions

- [SectionedInputBuffer](#) (void)
Default constructor.
- `void activate (std::ifstream &stream, std::size_t spos, std::size_t epos)`
Activate the object.
- `void deactivate` (void)
Deactivate the object.

- virtual `std::streambuf::int_type underflow ()`
Get a character in the case of depletion of the read buffer.
- `SectionedInputBuffer` (const `SectionedInputBuffer &`)
Not implemented.
- `SectionedInputBuffer & operator=` (const `SectionedInputBuffer &`)
Not implemented.

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`
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 53 of file `checkpoint_input_manager.hh`.

8.15.2 Constructor & Destructor Documentation

8.15.2.1 `jeod::SectionedInputBuffer::~~SectionedInputBuffer ()` `[inline]`

Destructor.

For now, this does nothing.

Definition at line 62 of file `checkpoint_input_manager.hh`.

8.15.2.2 `jeod::SectionedInputBuffer::SectionedInputBuffer (void) [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 45 of file `checkpoint_input_manager.cc`.

8.15.2.3 `jeod::SectionedInputBuffer::SectionedInputBuffer (const SectionedInputBuffer &) [private]`

Not implemented.

8.15.3 Member Function Documentation**8.15.3.1** `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

<i>in</i>	<i>stream</i>	Checkpoint file input file stream
<i>in</i>	<i>spos</i>	Section data start position
<i>in</i>	<i>epos</i>	Section data end position

Definition at line 68 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 `void jeod::SectionedInputBuffer::deactivate (void) [inline], [private]`

Deactivate the object.

Used to force a badly behaving stream to disconnect.

Definition at line 87 of file `checkpoint_input_manager.hh`.

References `at_eof`, and `file_buf`.

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

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

Conversion to boolean.

Returns

False if object is OK.

Definition at line 69 of file `checkpoint_input_manager.hh`.

References `file_buf`.

8.15.3.4 **SectionedInputBuffer& jeod::SectionedInputBuffer::operator= (const SectionedInputBuffer &)** [private]

Not implemented.

8.15.3.5 **std::streambuf::int_type jeod::SectionedInputBuffer::underflow (void)** [private], [virtual]

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 86 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 **friend class SectionedInputStream** [friend]

Definition at line 54 of file checkpoint_input_manager.hh.

8.15.5 Field Documentation

8.15.5.1 **bool jeod::SectionedInputBuffer::at_eof** [private]

At EOF in the file or in the section?

trick_io(**)

Definition at line 123 of file checkpoint_input_manager.hh.

Referenced by activate(), deactivate(), and underflow().

8.15.5.2 **char jeod::SectionedInputBuffer::buf** [private]

Input buffer.

trick_io(**)

Definition at line 128 of file checkpoint_input_manager.hh.

Referenced by underflow().

8.15.5.3 **size_t jeod::SectionedInputBuffer::curr_pos** [private]

The current position of the file_buf reader.

trick_io(**)

Definition at line 118 of file checkpoint_input_manager.hh.

Referenced by activate(), and underflow().

8.15.5.4 `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 113 of file `checkpoint_input_manager.hh`.

Referenced by `activate()`, and `underflow()`.

8.15.5.5 `std::filebuf* jeod::SectionedInputBuffer::file_buf` [private]

The file buffer that reads from the checkpoint file.

`trick_io(**)`

Definition at line 101 of file `checkpoint_input_manager.hh`.

Referenced by `activate()`, `deactivate()`, `operator!()`, and `underflow()`.

8.15.5.6 `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 107 of file `checkpoint_input_manager.hh`.

Referenced by `activate()`.

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

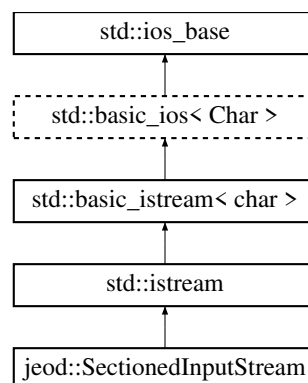
- [checkpoint_input_manager.hh](#)
- [checkpoint_input_manager.cc](#)

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](#) ()

Destruct a [SectionedInputStream](#) object.

- bool [is_activatable](#) () const

Determine if the stream is able to be activated.

- bool [activate](#) ()

Activate the object.

- void [deactivate](#) (void)

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](#).

- [SectionedInputStream](#) & [operator=](#) (const [SectionedInputStream](#) &)

Not implemented.

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 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 273 of file checkpoint_input_manager.hh.

8.16.2 Constructor & Destructor Documentation

8.16.2.1 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 130 of file checkpoint_input_manager.cc.

8.16.2.2 jeod::SectionedInputStream::SectionedInputStream (const SectionedInputStream & source)

Construct a [SectionedInputStream](#) object by copying from another.

Parameters

<i>in</i>	<i>source</i>	Source object
-----------	---------------	---------------

Definition at line 176 of file checkpoint_input_manager.cc.

References [jeod::SimInterfaceMessages::implementation_error](#), [is_active](#), [is_copy](#), [manager](#), and [stream](#).

8.16.2.3 jeod::SectionedInputStream::~~SectionedInputStream (void)

Destruct a [SectionedInputStream](#) object.

Definition at line 204 of file checkpoint_input_manager.cc.

References [jeod::CheckPointInputManager::deregister_reader\(\)](#), [is_active](#), and [manager](#).

8.16.2.4 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

<i>in</i>	<i>mngr</i>	The stream manager
<i>in</i>	<i>ifstream</i>	The input file stream
<i>in</i>	<i>spos</i>	Start position of section data
<i>in</i>	<i>epos</i>	End position of section data

Definition at line 153 of file checkpoint_input_manager.cc.

8.16.3 Member Function Documentation

8.16.3.1 bool jeod::SectionedInputStream::activate (void)

Activate the object.

Note

Using the object for reading prior to activation will result in EOF.

Returns

True if activated.

Definition at line 246 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 void jeod::SectionedInputStream::deactivate (void)

Deactivate the object.

Note

Deactivation is undoable.

Definition at line 292 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::JeodTrick10MemoryInterface::restore_containers().

8.16.3.3 `bool jeod::SectionedInputStream::is_activatable (void) const`

Determine if the stream is able to be activated.

Returns

True if object can be activated.

Definition at line 219 of file checkpoint_input_manager.cc.

References jeod::CheckPointInputManager::have_active_reader(), is_active, manager, and stream.

8.16.3.4 `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 310 of file checkpoint_input_manager.hh.

8.16.3.5 `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 301 of file checkpoint_input_manager.hh.

References is_active, sectbuf, and stream.

8.16.3.6 `SectionedInputStream& jeod::SectionedInputStream::operator= (const SectionedInputStream &) [private]`

Not implemented.

8.16.4 Friends And Related Function Documentation

8.16.4.1 `friend class CheckPointInputManager [friend]`

Definition at line 274 of file checkpoint_input_manager.hh.

8.16.5 Field Documentation

8.16.5.1 `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 351 of file checkpoint_input_manager.hh.

Referenced by activate().

8.16.5.2 `bool jeod::SectionedInputStream::is_active` `[private]`

Is this an active object? In the end, there can be only one.

trick_io(**)

Definition at line 363 of file checkpoint_input_manager.hh.

Referenced by activate(), deactivate(), is_activatable(), operator!(), SectionedInputStream(), and ~SectionedInputStream().

8.16.5.3 `bool jeod::SectionedInputStream::is_copy` `[private]`

Is this a copy of some other [SectionedInputStream](#)? Copies of copies are verboten.

trick_io(**)

Definition at line 357 of file checkpoint_input_manager.hh.

Referenced by SectionedInputStream().

8.16.5.4 `CheckpointInputManager* jeod::SectionedInputStream::manager` `[private]`

The input manager that created this object.

trick_io(**)

Definition at line 334 of file checkpoint_input_manager.hh.

Referenced by activate(), deactivate(), is_activatable(), SectionedInputStream(), and ~SectionedInputStream().

8.16.5.5 `SectionedInputBuffer jeod::SectionedInputStream::sectbuf` `[private]`

The std::streambuf that does the reading from the file.

trick_io(**)

Definition at line 329 of file checkpoint_input_manager.hh.

Referenced by activate(), deactivate(), and operator!().

8.16.5.6 `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 345 of file checkpoint_input_manager.hh.

Referenced by activate().

8.16.5.7 `std::ifstream*` `jeod::SectionedInputStream::stream` [private]

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

`trick_io(**)`

Definition at line 339 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:

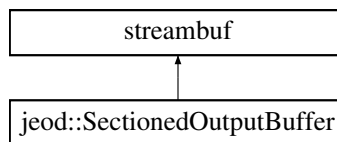
- [checkpoint_input_manager.hh](#)
- [checkpoint_input_manager.cc](#)

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](#) ()
Destructor.
- `bool operator! () const`
Conversion to boolean.

Private Member Functions

- [SectionedOutputBuffer](#) (void)
Default constructor.
- [SectionedOutputBuffer](#) (std::ofstream *stream)
- void [activate](#) (std::ofstream &stream)
Activate the object.
- void [deactivate](#) (void)
Deactivate the object.
- virtual std::streambuf::int_type [overflow](#) (std::streambuf::int_type c)
Write a character in the case of overflow of the write buffer.
- [SectionedOutputBuffer](#) (const [SectionedOutputBuffer](#) &)
Not implemented.
- [SectionedOutputBuffer](#) & [operator=](#) (const [SectionedOutputBuffer](#) &)
Not implemented.

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 50 of file `checkpoint_output_manager.hh`.

8.17.2 Constructor & Destructor Documentation

8.17.2.1 `jeod::SectionedOutputBuffer::~~SectionedOutputBuffer () [inline]`

Destructor.

For now, this does nothing.

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

8.17.2.2 `jeod::SectionedOutputBuffer::SectionedOutputBuffer (void) [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 45 of file `checkpoint_output_manager.cc`.

8.17.2.3 `jeod::SectionedOutputBuffer::SectionedOutputBuffer (std::ofstream * stream) [explicit], [private]`

8.17.2.4 `jeod::SectionedOutputBuffer::SectionedOutputBuffer (const SectionedOutputBuffer &) [private]`

Not implemented.

8.17.3 Member Function Documentation

8.17.3.1 `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

<i>in</i>	<i>stream</i>	Output file stream
-----------	---------------	--------------------

Definition at line 61 of file checkpoint_output_manager.cc.

References file_buf.

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

8.17.3.2 void jeod::SectionedOutputBuffer::deactivate (void) [inline], [private]

Deactivate the object.

Used to disconnect the buffer when the stream is done, sometimes by force.

Definition at line 85 of file checkpoint_output_manager.hh.

References file_buf.

Referenced by jeod::SectionedOutputStream::deactivate().

8.17.3.3 bool jeod::SectionedOutputBuffer::operator! () const [inline]

Conversion to boolean.

Returns

False if object is OK.

Definition at line 66 of file checkpoint_output_manager.hh.

References file_buf.

8.17.3.4 SectionedOutputBuffer& jeod::SectionedOutputBuffer::operator= (const SectionedOutputBuffer &) [private]

Not implemented.

8.17.3.5 std::streambuf::int_type jeod::SectionedOutputBuffer::overflow (std::streambuf::int_type *ch*) [private], [virtual]

Write a character in the case of overflow of the write buffer.

For now, the buffer always overflows.

Returns

Status: EOF => failed

Parameters

<i>in</i>	<i>ch</i>	Character to be writter
-----------	-----------	-------------------------

Definition at line 76 of file checkpoint_output_manager.cc.

References file_buf.

8.17.4 Friends And Related Function Documentation

8.17.4.1 friend class SectionedOutputStream [friend]

Definition at line 51 of file checkpoint_output_manager.hh.

8.17.5 Field Documentation

8.17.5.1 std::filebuf* jeod::SectionedOutputBuffer::file_buf [private]

The file buffer that writes to the checkpoint file.

trick_io(**)

Definition at line 101 of file checkpoint_output_manager.hh.

Referenced by activate(), deactivate(), operator!(), and overflow().

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

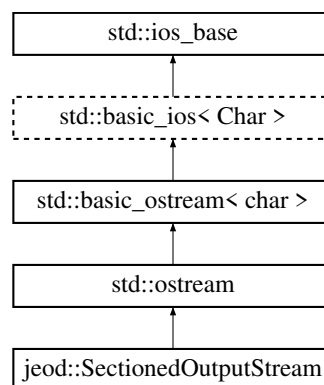
- [checkpoint_output_manager.hh](#)
- [checkpoint_output_manager.cc](#)

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](#) ()
Destruct a [SectionedOutputStream](#) object.
- 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](#).*
- [SectionedOutputStream](#) & [operator=](#) (const [SectionedOutputStream](#) &)
- *Not implemented.*

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 132 of file checkpoint_output_manager.hh.

8.18.2 Constructor & Destructor Documentation

8.18.2.1 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 118 of file `checkpoint_output_manager.cc`.

8.18.2.2 jeod::SectionedOutputStream::SectionedOutputStream (const SectionedOutputStream & source)

Construct a [SectionedOutputStream](#) object by copying from another.

Parameters

in	<i>source</i>	Source object
----	---------------	---------------

Definition at line 168 of file `checkpoint_output_manager.cc`.

References `jeod::SimInterfaceMessages::implementation_error`, `is_active`, `is_copy`, `manager`, and `stream`.

8.18.2.3 jeod::SectionedOutputStream::~~SectionedOutputStream (void)

Destruct a [SectionedOutputStream](#) object.

Definition at line 197 of file `checkpoint_output_manager.cc`.

References `deactivate()`.

8.18.2.4 jeod::SectionedOutputStream::SectionedOutputStream (CheckPointOutputManager * mngr, std::ofstream & ofstream, 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>ofstream</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 143 of file `checkpoint_output_manager.cc`.

8.18.3 Member Function Documentation

8.18.3.1 bool jeod::SectionedOutputStream::activate (void)

Activate the object.

Note

Using the object for writing prior to activation will write nothing.

Returns

True if activated.

Definition at line 237 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 void jeod::SectionedOutputStream::deactivate (void)

Deactivate the object.

Note

Deactivation is undoable.

Definition at line 291 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 bool jeod::SectionedOutputStream::is_activatable (void) const

Determine if the stream is able to be activated.

Returns

True if object can be activated.

Definition at line 210 of file checkpoint_output_manager.cc.

References jeod::CheckPointOutputManager::have_active_writer(), is_active, manager, and stream.

8.18.3.4 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 168 of file checkpoint_output_manager.hh.

8.18.3.5 bool jeod::SectionedOutputStream::operator! () const [inline]

Conversion to boolean.

Returns

False if object is OK.

Definition at line 159 of file checkpoint_output_manager.hh.

References is_active, sectbuf, and stream.

8.18.3.6 SectionedOutputStream& jeod::SectionedOutputStream::operator= (const SectionedOutputStream &)
[private]

Not implemented.

8.18.4 Friends And Related Function Documentation

8.18.4.1 friend class CheckPointOutputManager [friend]

Definition at line 133 of file checkpoint_output_manager.hh.

8.18.5 Field Documentation

8.18.5.1 bool jeod::SectionedOutputStream::is_active [private]

Is this an active object? In the end, there can be only one.

trick_io(**)

Definition at line 225 of file checkpoint_output_manager.hh.

Referenced by activate(), deactivate(), is_activatable(), operator!(), and SectionedOutputStream().

8.18.5.2 bool jeod::SectionedOutputStream::is_copy [private]

Is this a copy of some other [SectionedOutputStream](#)? Copies of copies are verboten.

trick_io(**)

Definition at line 219 of file checkpoint_output_manager.hh.

Referenced by SectionedOutputStream().

8.18.5.3 CheckPointOutputManager* jeod::SectionedOutputStream::manager [private]

The input manager that created this object.

trick_io(**)

Definition at line 193 of file checkpoint_output_manager.hh.

Referenced by activate(), deactivate(), is_activatable(), and SectionedOutputStream().

8.18.5.4 SectionedOutputBuffer jeod::SectionedOutputStream::sectbuf [private]

The std::streambuf that does the writing to the file.

trick_io(**)

Definition at line 188 of file checkpoint_output_manager.hh.

Referenced by activate(), deactivate(), and operator!().

8.18.5.5 const std::string* jeod::SectionedOutputStream::section_end [private]

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

trick_io(**)

Definition at line 208 of file checkpoint_output_manager.hh.

Referenced by deactivate().

8.18.5.6 `const std::string* jeod::SectionedOutputStream::section_start` `[private]`

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

trick_io(**)

Definition at line 203 of file checkpoint_output_manager.hh.

Referenced by activate().

8.18.5.7 `std::ofstream* jeod::SectionedOutputStream::stream` `[private]`

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

trick_io(**)

Definition at line 198 of file checkpoint_output_manager.hh.

Referenced by activate(), deactivate(), is_activatable(), operator!(), and SectionedOutputStream().

8.18.5.8 `const std::string jeod::SectionedOutputStream::tag` `[private]`

The name of the checkpoint file section.

trick_io(**)

Definition at line 213 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 436 of file checkpoint_input_manager.hh.

8.19.2 Constructor & Destructor Documentation

8.19.2.1 jeod::CheckPointInputManager::SectionInfo::SectionInfo (std::size_t start, std::size_t end) [inline]

Non-default constructor.

Parameters

in	start	Start position
in	end	End position

Definition at line 452 of file checkpoint_input_manager.hh.

8.19.3 Field Documentation

8.19.3.1 size_t jeod::CheckPointInputManager::SectionInfo::end_pos

Position of the first unreadable character after a section.

trick_io(**)

Definition at line 445 of file checkpoint_input_manager.hh.

Referenced by jeod::CheckPointInputManager::create_section_reader().

8.19.3.2 size_t jeod::CheckPointInputManager::SectionInfo::start_pos

Position of the first readable character of a section.

trick_io(**)

Definition at line 440 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>
```

Static Public Attributes

- static char const * [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 char const * [interface_error](#) = "utils/sim_interface/" "interface_error"
Message issued when issues arise from interacting with the sim engine.
- static char const * [phasing_error](#) = "utils/sim_interface/" "phasing_error"
Message issued when things happen out of order.
- static char const * [integration_error](#) = "utils/sim_interface/" "integration_error"
Message issued when something goes awry with integration.
- static char const * [implementation_error](#) = "utils/sim_interface/" "implementation_error"
Message issued when something went wrong with the implementation.

Private Member Functions

- [SimInterfaceMessages](#) (void)
- [SimInterfaceMessages](#) (const [SimInterfaceMessages](#) &)
- [SimInterfaceMessages](#) & operator= (const [SimInterfaceMessages](#) &)

8.20.1 Detailed Description

Specifies the message IDs used in the sim_interface model.

Definition at line 45 of file sim_interface_messages.hh.

8.20.2 Constructor & Destructor Documentation

8.20.2.1 `jeod::SimInterfaceMessages::SimInterfaceMessages (void) [private]`

8.20.2.2 `jeod::SimInterfaceMessages::SimInterfaceMessages (const SimInterfaceMessages &) [private]`

8.20.3 Member Function Documentation

8.20.3.1 `SimInterfaceMessages& jeod::SimInterfaceMessages::operator= (const SimInterfaceMessages &) [private]`

8.20.4 Field Documentation

8.20.4.1 `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 75 of file sim_interface_messages.hh.

Referenced by `jeod::SectionedOutputStream::activate()`, `jeod::SectionedInputStream::activate()`, `jeod::CheckPointInputManager::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 `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 70 of file sim_interface_messages.hh.

Referenced by `jeod::JeodDynbodyIntegrationLoop::add_integrable_object()`, `jeod::JeodDynbodyIntegrationLoop::initialize_integ_loop()`, `jeod::JeodDynbodyIntegrationLoop::integrate_dt()`, `jeod::JeodDynbodyIntegrationLoop::JeodDynbodyIntegrationLoop()`, and `jeod::JeodDynbodyIntegrationLoop::update_integration_group()`.

8.20.4.3 `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 60 of file `sim_interface_messages.hh`.

Referenced by `jeod::JeodTrick10MemoryInterface::checkpoint_allocations()`, `jeod::JeodTrick10MemoryInterface::checkpoint_containers()`, `jeod::JeodTrickMemoryInterface::deregister_allocation()`, `jeod::JeodTrick10MemoryInterface::deregister_container()`, `jeod::JeodTrickMemoryInterface::find_attributes()`, `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 `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 65 of file `sim_interface_messages.hh`.

Referenced by `jeod::BasicJeodTrickSimInterface::get_checkpoint_reader_internal()`, `jeod::BasicJeodTrickSimInterface::get_checkpoint_writer_internal()`, and `jeod::JeodSimulationInterface::set_mode()`.

8.20.4.5 `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 55 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:

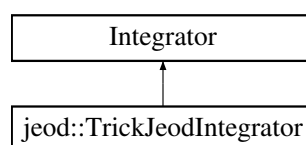
- [sim_interface_messages.hh](#)
- [sim_interface_messages.cc](#)

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

- virtual [~TrickJeodIntegrator](#) ()
Destructor.
- int [integrate](#) ()
Does nothing.
- void [initialize](#) (int, double)
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 51 of file `jeod_trick_integrator.hh`.

8.21.2 Constructor & Destructor Documentation

8.21.2.1 virtual `jeod::TrickJeodIntegrator::~~TrickJeodIntegrator ()` `[inline]`, `[virtual]`

Destructor.

Definition at line 64 of file `jeod_trick_integrator.hh`.

8.21.3 Member Function Documentation

8.21.3.1 void `jeod::TrickJeodIntegrator::initialize (int , double)` `[inline]`

Does nothing.

Definition at line 79 of file `jeod_trick_integrator.hh`.

8.21.3.2 int `jeod::TrickJeodIntegrator::integrate ()` `[inline]`

Does nothing.

Definition at line 74 of file `jeod_trick_integrator.hh`.

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

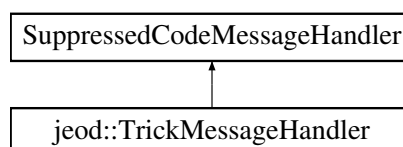
- [jeod_trick_integrator.hh](#)

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](#) (void)
Default constructor.
- virtual [~TrickMessageHandler](#) (void)
Destructor.
- virtual void [register_contents](#) (void)
Register the [TrickMessageHandler](#)'s checkpointable contents.

Protected Member Functions

- virtual 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
Handle a message.

Private Member Functions

- [TrickMessageHandler](#) (const [TrickMessageHandler](#) &)
Not implemented.
- [TrickMessageHandler](#) & [operator=](#) (const [TrickMessageHandler](#) &)
Not implemented.

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 58 of file `trick_message_handler.hh`.

8.22.2 Constructor & Destructor Documentation

8.22.2.1 `jeod::TrickMessageHandler::TrickMessageHandler (void) [inline]`

Default constructor.

Definition at line 75 of file `trick_message_handler.hh`.

8.22.2.2 `virtual jeod::TrickMessageHandler::~~TrickMessageHandler (void) [inline],[virtual]`

Destructor.

Definition at line 80 of file `trick_message_handler.hh`.

8.22.2.3 `jeod::TrickMessageHandler::TrickMessageHandler (const TrickMessageHandler &) [private]`

Not implemented.

8.22.3 Member Function Documentation

8.22.3.1 `TrickMessageHandler& jeod::TrickMessageHandler::operator= (const TrickMessageHandler &)`
[private]

Not implemented.

8.22.3.2 `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` [protected],[virtual]

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 91 of file `trick_message_handler.cc`.

References `MAX_MSG_SIZE`.

8.22.3.3 `void jeod::TrickMessageHandler::register_contents (void)` [virtual]

Register the [TrickMessageHandler](#)'s checkpointable contents.

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

8.22.4 Friends And Related Function Documentation

8.22.4.1 `void init_attrjeod__TrickMessageHandler ()` [friend]

8.22.4.2 `friend class InputProcessor` [friend]

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

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

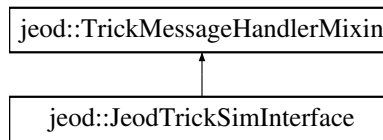
- [trick_message_handler.hh](#)
- [trick_message_handler.cc](#)

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 constructor.
- virtual [~TrickMessageHandlerMixin](#) ()
Destructor.

Protected Attributes

- [TrickMessageHandler](#) message_handler
The global MessageHandler.

Private Member Functions

- [TrickMessageHandlerMixin](#) (const [TrickMessageHandlerMixin](#) &)
Not implemented.
- [TrickMessageHandlerMixin](#) & operator= (const [TrickMessageHandlerMixin](#) &)
Not implemented.

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 215 of file `trick_sim_interface.hh`.

8.23.2 Constructor & Destructor Documentation

8.23.2.1 `jeod::TrickMessageHandlerMixin::TrickMessageHandlerMixin ()` `[inline]`

Default constructor.

Definition at line 223 of file `trick_sim_interface.hh`.

8.23.2.2 `virtual jeod::TrickMessageHandlerMixin::~~TrickMessageHandlerMixin ()` `[inline]`, `[virtual]`

Destructor.

Definition at line 228 of file `trick_sim_interface.hh`.

8.23.2.3 `jeod::TrickMessageHandlerMixin::TrickMessageHandlerMixin (const TrickMessageHandlerMixin &)`
[private]

Not implemented.

8.23.3 Member Function Documentation

8.23.3.1 `TrickMessageHandlerMixin& jeod::TrickMessageHandlerMixin::operator= (const TrickMessageHandlerMixin &)` [private]

Not implemented.

8.23.4 Friends And Related Function Documentation

8.23.4.1 `void init_attrjeod__TrickMessageHandlerMixin ()` [friend]

8.23.4.2 `friend class InputProcessor` [friend]

Definition at line 216 of file `trick_sim_interface.hh`.

8.23.5 Field Documentation

8.23.5.1 `TrickMessageHandler jeod::TrickMessageHandlerMixin::message_handler` [protected]

The global MessageHandler.

`trick_units(-)`

Definition at line 237 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.

Definition in file [checkpoint_input_manager.cc](#).

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 <istream>
#include <fstream>
#include <string>
#include <map>
```

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.

Definition in file [checkpoint_input_manager.hh](#).

9.3 checkpoint_output_manager.cc File Reference

Define [CheckpointOutputManager](#) member functions and of related classes.

```
#include <cstdint>
#include <cstring>
#include <iostream>
#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.

Definition in file [checkpoint_output_manager.cc](#).

9.4 checkpoint_output_manager.hh File Reference

Define class [CheckpointOutputManager](#) and related classes.

```
#include "utils/sim_interface/include/jeod_class.hh"
#include <ostream>
#include <fstream>
#include <string>
#include <map>
```

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.

Definition in file [checkpoint_output_manager.hh](#).

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.

Definition in file [class_declarations.hh](#).

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.

Definition in file [config.hh](#).

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.

Definition in file [config_test_harness.hh](#).

9.8 config_trick10.hh File Reference

Configure JEOD for use in a Trick10 environment.

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_FRIENDS(class_name)`
- `#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 namespace Trick { class Integrator; }`
- `#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.

Definition in file [config_trick10.hh](#).

9.9 jeod_class.hh File Reference

Define the JEOD class declaration macros JEOD_MAKE_SIM_INTERFACES and and JEOD_DECLARE_SIM_INTERFACES.

```
#include "config.hh"
```

Macros

- `#define JEOD_MAKE_SIM_INTERFACES(class_name) JEOD_CLASS_ESTABLISH_FRIENDS(class_name)`
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 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](#).

Definition in file [jeod_class.hh](#).

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.

Definition in file [jeod_integrator_interface.hh](#).

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.

Definition in file [jeod_trick_integrator.hh](#).

9.12 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.

9.12.1 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.

Definition in file [memory_attributes.hh](#).

9.12.2 Macro Definition Documentation

9.12.2.1 `#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 78 of file [memory_attributes.hh](#).

9.12.2.2 `#define JEOD_DECLARE_ATTRIBUTES(class_name)`

[JEOD_DECLARE_ATTRIBUTES\(class_name\)](#) This macro is obsolete.

Definition at line 68 of file [memory_attributes.hh](#).

9.13 memory_interface.cc File Reference

Implement the MemoryInterface class.

```
#include "../include/memory_interface.hh"
```

Namespaces

- [jeod](#)

Namespace jeod.

9.13.1 Detailed Description

Implement the MemoryInterface class.

Definition in file [memory_interface.cc](#).

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 <cstdlib>
#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.

Definition in file [memory_interface.hh](#).

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`.

Definition in file `sim_interface_messages.cc`.

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.

Definition in file `sim_interface_messages.hh`.

9.17 `simulation_interface.cc` File Reference

Implement `SimulationInterface` methods.

```
#include <cstdint>
#include "utils/message/include/message_handler.hh"
#include "utils/memory/include/memory_manager.hh"
#include "../include/simulation_interface.hh"
#include "../include/sim_interface_messages.hh"
```

Namespaces

- `jeod`
Namespace `jeod`.

9.17.1 Detailed Description

Implement SimulationInterface methods.

Definition in file [simulation_interface.cc](#).

9.18 simulation_interface.hh File Reference

Define the abstract class JeodSimulationInterface.

```
#include <string>
#include "class_declarations.hh"
#include "jeod_class.hh"
#include "checkpoint_input_manager.hh"
#include "checkpoint_output_manager.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.

Definition in file [simulation_interface.hh](#).

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/simulation_interface.hh"
#include "../include/sim_interface_messages.hh"
#include "../include/trick10_memory_interface.hh"
```

Namespaces

- [jeod](#)

Namespace jeod.

Variables

- Trick::MemoryManager * [trick_MM](#)

9.19.1 Detailed Description

Define JeodTrickMemoryInterface methods.

Definition in file [trick10_memory_interface.cc](#).

9.20 trick10_memory_interface.hh File Reference

Define the interface for registering / deregistering memory with [Trick](#).

```
#include <cstdlib>
#include <cstring>
#include <list>
#include <map>
#include <stdint.h>
#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](#).

Definition in file [trick10_memory_interface.hh](#).

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_manager/include/dyn_manager.hh"
#include "dynamics/dyn_body/include/dyn_body.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.

Definition in file [trick_dynbody_integ_loop.cc](#).

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.

Definition in file [trick_dynbody_integ_loop.hh](#).

9.23 trick_memory_interface.cc File Reference

Define JeodTrickMemoryInterface methods.

```
#include <cstdlib>
#include <dlfcn.h>
#include <iomanip>
#include <sstream>
#include "utils/message/include/message_handler.hh"
#include "../include/simulation_interface.hh"
#include "../include/sim_interface_messages.hh"
#include "../include/trick_memory_interface.hh"
```

Namespaces

- [jeod](#)

Namespace [jeod](#).

9.23.1 Detailed Description

Define JeodTrickMemoryInterface methods.

Definition in file [trick_memory_interface.cc](#).

9.24 trick_memory_interface.hh File Reference

Define the interface for registering / deregistering memory with [Trick](#).

```
#include <cstdlib>
#include <cstring>
#include <list>
#include <map>
#include <stdint.h>
#include <string>
#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](#).

Definition in file [trick_memory_interface.hh](#).

9.25 [trick_memory_interface_alloc.cc](#) File Reference

Define *JeodTrickMemoryInterface* methods related to allocation/deallocation.

```
#include <cstddef>
#include <cstdlib>
#include <cstdio>
#include <cstring>
#include <dlfcn.h>
#include <iomanip>
#include <sstream>
#include <typeinfo>
#include "sim_services/MemoryManager/include/attributes.h"
#include "sim_services/MemoryManager/include/ADefParseContext.hh"
#include "sim_services/MemoryManager/include/MemoryManager.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/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.

Definition in file [trick_memory_interface_alloc.cc](#).

9.26 trick_memory_interface_attrib.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.

Definition in file [trick_memory_interface_attrib.cc](#).

9.27 trick_memory_interface_chkpnt.cc File Reference

Define JeodTrick10MemoryInterface methods related to checkpoint/restart.

```
#include <cstdint>
#include <cstdlib>
#include <cstdio>
#include <cstring>
#include <dlfcn.h>
#include <iomanip>
#include <sstream>
#include <typeinfo>
#include "sim_services/MemoryManager/include/attributes.h"
#include "sim_services/MemoryManager/include/ADefParseContext.hh"
#include "sim_services/MemoryManager/include/MemoryManager.hh"
#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.

Definition in file [trick_memory_interface_chkpt.cc](#).

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/attributes.h"
#include "sim_services/MemoryManager/include/MemoryManager.hh"
#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/simulation_interface.hh"
#include "../include/sim_interface_messages.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.

Definition in file [trick_memory_interface_xlate.cc](#).

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.

Definition in file [trick_message_handler.cc](#).

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.

Definition in file [trick_message_handler.hh](#).

9.31 `trick_sim_interface.cc` File Reference

Implement `TrickSimInterface` methods.

```
#include "sim_services/Executive/include/exec_proto.h"
#include "sim_services/Message/include/message_proto.h"
#include "sim_services/CommandLineArguments/include/command_line_protos.h"
#include "utils/memory/include/jeod_alloc.hh"
#include "utils/message/include/message_handler.hh"
#include "../include/sim_interface_messages.hh"
#include "../include/jeod_trick_integrator.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.

Definition in file [trick_sim_interface.cc](#).

9.32 `trick_sim_interface.hh` File Reference

Define the class `JeodTrickSimInterface`.

```
#include "utils/memory/include/memory_manager.hh"
#include "simulation_interface.hh"
#include "trick_memory_interface.hh"
#include "trickl0_memory_interface.hh"
#include "trick_message_handler.hh"
#include "jeod_class.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.

Definition in file [trick_sim_interface.hh](#).

Index

- ~BasicJeodTrickSimInterface
 - jeod::BasicJeodTrickSimInterface, [24](#)
- ~JeodDynbodyIntegrationLoop
 - jeod::JeodDynbodyIntegrationLoop, [45](#)
- ~JeodIntegratorInterface
 - jeod::JeodIntegratorInterface, [51](#)
- ~JeodMemoryInterface
 - jeod::JeodMemoryInterface, [53](#)
- ~JeodSimulationInterface
 - jeod::JeodSimulationInterface, [60](#)
- ~JeodTrick10MemoryInterface
 - jeod::JeodTrick10MemoryInterface, [69](#)
- ~JeodTrickIntegrator
 - jeod::JeodTrickIntegrator, [75](#)
- ~JeodTrickMemoryInterface
 - jeod::JeodTrickMemoryInterface, [81](#)
- ~JeodTrickSimInterface
 - jeod::JeodTrickSimInterface, [90](#)
- ~SectionedInputBuffer
 - jeod::SectionedInputBuffer, [92](#)
- ~SectionedInputStream
 - jeod::SectionedInputStream, [99](#)
- ~SectionedOutputBuffer
 - jeod::SectionedOutputBuffer, [103](#)
- ~SectionedOutputStream
 - jeod::SectionedOutputStream, [107](#)
- ~TrickJeodIntegrator
 - jeod::TrickJeodIntegrator, [114](#)
- ~TrickMessageHandler
 - jeod::TrickMessageHandler, [115](#)
- ~TrickMessageHandlerMixin
 - jeod::TrickMessageHandlerMixin, [117](#)
- activate
 - jeod::SectionedInputBuffer, [93](#)
 - jeod::SectionedInputStream, [99](#)
 - jeod::SectionedOutputBuffer, [103](#)
 - jeod::SectionedOutputStream, [107](#)
- add_integrable_object
 - jeod::JeodDynbodyIntegrationLoop, [45](#)
- add_sim_object
 - jeod::JeodDynbodyIntegrationLoop, [45](#)
- add_sim_object_bodies
 - jeod::JeodDynbodyIntegrationLoop, [46](#)
- allocation_map
 - jeod::JeodTrickMemoryInterface, [88](#)
- AllocationMap
 - jeod::JeodTrickMemoryInterface, [80](#)
- AllocationMapEntry
 - jeod::JeodTrickMemoryInterface::AllocationMapEntry, [21](#)
- at_eof
 - jeod::SectionedInputBuffer, [94](#)
- BasicJeodTrickSimInterface
 - jeod::BasicJeodTrickSimInterface, [24](#), [25](#)
- buf
 - jeod::SectionedInputBuffer, [94](#)
- CLASS
 - SimInterface, [15](#)
- CheckPointInputManager
 - jeod::CheckPointInputManager, [31](#)
 - jeod::SectionedInputStream, [100](#)
- CheckPointOutputManager
 - jeod::CheckPointOutputManager, [36](#), [37](#)
 - jeod::SectionedOutputStream, [109](#)
- Checkpoint
 - jeod::JeodSimulationInterface, [60](#)
- checkpoint_allocations
 - jeod::BasicJeodTrickSimInterface, [25](#)
 - jeod::JeodTrick10MemoryInterface, [69](#)
 - jeod::JeodTrickMemoryInterface, [81](#)
- checkpoint_containers
 - jeod::BasicJeodTrickSimInterface, [25](#)
 - jeod::JeodTrick10MemoryInterface, [69](#)
 - jeod::JeodTrickMemoryInterface, [81](#)
- checkpoint_file_name
 - jeod::BasicJeodTrickSimInterface, [28](#)
- checkpoint_input_manager.cc, [119](#)
- checkpoint_input_manager.hh, [119](#)
- checkpoint_output_manager.cc, [120](#)
- checkpoint_output_manager.hh, [120](#)
- checkpoint_reader
 - jeod::BasicJeodTrickSimInterface, [28](#)
- checkpoint_writer
 - jeod::BasicJeodTrickSimInterface, [29](#)
- class_declarations.hh, [121](#)
- close_checkpoint_file
 - jeod::BasicJeodTrickSimInterface, [25](#)
- close_restart_file
 - jeod::BasicJeodTrickSimInterface, [25](#)
- collect_derivatives
 - jeod::JeodDynbodyIntegrationLoop, [46](#)
- config.hh, [121](#)
- config_test_harness.hh, [122](#)
- config_trick10.hh, [122](#)
- configure
 - jeod::JeodSimulationInterface, [61](#)

- construct_identifier
 - jeod::JeodTrickMemoryInterface, 81
- Construction
 - jeod::JeodSimulationInterface, 60
- container
 - jeod::JeodTrickMemoryInterface::ContainerListEntry, 41
- container_list
 - jeod::JeodTrickMemoryInterface, 88
- ContainerList
 - jeod::JeodTrickMemoryInterface, 80
- ContainerListEntry
 - jeod::JeodTrickMemoryInterface::ContainerListEntry, 41
- create_integrator_interface
 - jeod::JeodSimulationInterface, 61
- create_integrator_internal
 - jeod::BasicJeodTrickSimInterface, 25
 - jeod::JeodSimulationInterface, 61
- create_section_reader
 - jeod::CheckPointInputManager, 31, 32
- create_section_writer
 - jeod::CheckPointOutputManager, 37
- create_trick_section_reader
 - jeod::CheckPointInputManager, 32
- create_trick_section_writer
 - jeod::CheckPointOutputManager, 37
- curr_pos
 - jeod::SectionedInputBuffer, 94
- current_reader
 - jeod::CheckPointInputManager, 34
- current_writer
 - jeod::CheckPointOutputManager, 39
- deactivate
 - jeod::SectionedInputBuffer, 93
 - jeod::SectionedInputStream, 99
 - jeod::SectionedOutputBuffer, 104
 - jeod::SectionedOutputStream, 108
- Dead
 - jeod::JeodSimulationInterface, 60
- default_first_step_deriv
 - jeod::JeodTrickIntegrator, 77
- deregister_allocation
 - jeod::JeodMemoryInterface, 54
 - jeod::JeodTrickMemoryInterface, 82
- deregister_container
 - jeod::JeodMemoryInterface, 54
 - jeod::JeodTrick10MemoryInterface, 69
 - jeod::JeodTrickMemoryInterface, 82
- deregister_reader
 - jeod::CheckPointInputManager, 32
- deregister_writer
 - jeod::CheckPointOutputManager, 38
- deriv_ephem_update
 - jeod::JeodDynbodyIntegrationLoop, 49
- dlhandle
 - jeod::JeodTrickMemoryInterface, 88
- dyn_manager
 - jeod::JeodDynbodyIntegrationLoop, 49
- ER7_UTILS_RESTRICT
 - SimInterface, 15
- ER7_UTILS_UNUSED
 - SimInterface, 15
- elem_name
 - jeod::JeodTrickMemoryInterface::ContainerListEntry, 41
- end_pos
 - jeod::CheckPointInputManager::SectionInfo, 111
 - jeod::SectionedInputBuffer, 94
 - jeod::SectionedInputStream, 101
- er7_utils, 19
- file_buf
 - jeod::SectionedInputBuffer, 95
 - jeod::SectionedOutputBuffer, 105
- filename
 - jeod::CheckPointInputManager, 34
 - jeod::CheckPointOutputManager, 39
- find_attributes
 - jeod::JeodMemoryInterface, 54
 - jeod::JeodTrickMemoryInterface, 82, 83
- find_containing_sim_object
 - jeod::JeodDynbodyIntegrationLoop, 46
- generic_message_handler
 - jeod::BasicJeodTrickSimInterface, 29
- get_address_at_name
 - jeod::JeodMemoryInterface, 55
 - jeod::JeodSimulationInterface, 61
 - jeod::JeodTrick10MemoryInterface, 70
 - jeod::JeodTrickMemoryInterface, 83
- get_checkpoint_file_name
 - jeod::BasicJeodTrickSimInterface, 26
- get_checkpoint_reader
 - jeod::JeodSimulationInterface, 62
- get_checkpoint_reader_internal
 - jeod::BasicJeodTrickSimInterface, 26
 - jeod::JeodSimulationInterface, 62
- get_checkpoint_writer
 - jeod::JeodSimulationInterface, 62
- get_checkpoint_writer_internal
 - jeod::BasicJeodTrickSimInterface, 26
 - jeod::JeodSimulationInterface, 62
- get_container_id
 - jeod::JeodTrick10MemoryInterface, 70
- get_dt
 - jeod::JeodTrickIntegrator, 76
- get_first_step_derivs_flag
 - jeod::JeodTrickIntegrator, 76
- get_integrator
 - jeod::JeodIntegratorInterface, 51
 - jeod::JeodTrickIntegrator, 76
- get_job_cycle
 - jeod::JeodSimulationInterface, 63
- get_job_cycle_internal
 - jeod::BasicJeodTrickSimInterface, 26

- jeod::JeodSimulationInterface, 63
- get_memory_interface
 - jeod::JeodSimulationInterface, 63
- get_memory_interface_internal
 - jeod::BasicJeodTrickSimInterface, 27
 - jeod::JeodSimulationInterface, 63
- get_mode
 - jeod::JeodSimulationInterface, 63
- get_name_at_address
 - jeod::JeodMemoryInterface, 55
 - jeod::JeodSimulationInterface, 64
 - jeod::JeodTrick10MemoryInterface, 70
 - jeod::JeodTrickMemoryInterface, 83
- get_trick_checkpoint_file
 - jeod::JeodTrick10MemoryInterface, 71
 - jeod::JeodTrickMemoryInterface, 84
- gravitation
 - jeod::JeodDynbodyIntegrationLoop, 46
- gravity_manager
 - jeod::JeodDynbodyIntegrationLoop, 49
- have_active_reader
 - jeod::CheckPointInputManager, 33
- have_active_writer
 - jeod::CheckPointOutputManager, 38
- id_length
 - jeod::JeodTrickMemoryInterface, 89
- id_prefix
 - jeod::JeodTrickMemoryInterface, 89
- implementation_error
 - jeod::SimInterfaceMessages, 112
- init_attrjeod__BasicJeodTrickSimInterface
 - jeod::BasicJeodTrickSimInterface, 28
- init_attrjeod__JeodDynbodyIntegrationLoop
 - jeod::JeodDynbodyIntegrationLoop, 49
- init_attrjeod__JeodIntegratorInterface
 - jeod::JeodIntegratorInterface, 52
- init_attrjeod__JeodMemoryInterface
 - jeod::JeodMemoryInterface, 58
- init_attrjeod__JeodSimulationInterface
 - jeod::JeodSimulationInterface, 65
- init_attrjeod__JeodTrick10MemoryInterface
 - jeod::JeodTrick10MemoryInterface, 73
- init_attrjeod__JeodTrickIntegrator
 - jeod::JeodTrickIntegrator, 77
- init_attrjeod__JeodTrickMemoryInterface
 - jeod::JeodTrickMemoryInterface, 88
- init_attrjeod__JeodTrickSimInterface
 - jeod::JeodTrickSimInterface, 91
- init_attrjeod__TrickMessageHandler
 - jeod::TrickMessageHandler, 116
- init_attrjeod__TrickMessageHandlerMixin
 - jeod::TrickMessageHandlerMixin, 118
- Initialization
 - jeod::JeodSimulationInterface, 60
- initialize
 - jeod::CheckPointInputManager, 33
 - jeod::TrickJeodIntegrator, 114
- initialize_integ_loop
 - jeod::JeodDynbodyIntegrationLoop, 47
- InputProcessor
 - jeod::BasicJeodTrickSimInterface, 28
 - jeod::JeodDynbodyIntegrationLoop, 49
 - jeod::JeodIntegratorInterface, 52
 - jeod::JeodMemoryInterface, 58
 - jeod::JeodSimulationInterface, 65
 - jeod::JeodTrick10MemoryInterface, 73
 - jeod::JeodTrickIntegrator, 77
 - jeod::JeodTrickMemoryInterface, 88
 - jeod::JeodTrickSimInterface, 91
 - jeod::TrickMessageHandler, 116
 - jeod::TrickMessageHandlerMixin, 118
- integ_constructor
 - jeod::JeodDynbodyIntegrationLoop, 49
- integ_group
 - jeod::JeodDynbodyIntegrationLoop, 49
- integ_group_factory
 - jeod::JeodDynbodyIntegrationLoop, 50
- integ_interface
 - jeod::JeodDynbodyIntegrationLoop, 50
- integrate
 - jeod::TrickJeodIntegrator, 114
- integrate_dt
 - jeod::JeodDynbodyIntegrationLoop, 47
- integration_error
 - jeod::SimInterfaceMessages, 112
- interface_error
 - jeod::SimInterfaceMessages, 112
- interpret_integration_type
 - jeod::JeodIntegratorInterface, 51
 - jeod::JeodTrickIntegrator, 76
- is_activatable
 - jeod::SectionedInputStream, 100
 - jeod::SectionedOutputStream, 108
- is_active
 - jeod::SectionedInputStream, 101
 - jeod::SectionedOutputStream, 109
- is_array
 - jeod::JeodTrickMemoryInterface::AllocationMap-Entry, 22
- is_checkpoint_restart_supported
 - jeod::JeodMemoryInterface, 56
 - jeod::JeodTrick10MemoryInterface, 71
 - jeod::JeodTrickMemoryInterface, 84
- is_copy
 - jeod::SectionedInputStream, 101
 - jeod::SectionedOutputStream, 109
- is_open
 - jeod::CheckPointInputManager, 34
 - jeod::CheckPointOutputManager, 39
- JEOD_ATTRIBUTES
 - memory_attributes.hh, 125
- JEOD_INTPTR_T
 - SimInterface, 16
- JEOD_PTRDIFF_T
 - SimInterface, 16

- JEOD_SIZE_T
 - SimInterface, 17
- JEOD_UINTPTR_T
 - SimInterface, 17
- JEOD_UNUSED
 - SimInterface, 17
- jeod, 19
- jeod::JeodSimulationInterface
 - Checkpoint, 60
 - Construction, 60
 - Dead, 60
 - Initialization, 60
 - NumModes, 60
 - Operational, 60
 - PostCheckpoint, 60
 - PreCheckpoint, 60
 - Restart, 60
 - Restore, 60
 - Shutdown, 60
- jeod::BasicJeodTrickSimInterface, 22
 - ~BasicJeodTrickSimInterface, 24
 - BasicJeodTrickSimInterface, 24, 25
 - checkpoint_allocations, 25
 - checkpoint_containers, 25
 - checkpoint_file_name, 28
 - checkpoint_reader, 28
 - checkpoint_writer, 29
 - close_checkpoint_file, 25
 - close_restart_file, 25
 - create_integrator_internal, 25
 - generic_message_handler, 29
 - get_checkpoint_file_name, 26
 - get_checkpoint_reader_internal, 26
 - get_checkpoint_writer_internal, 26
 - get_job_cycle_internal, 26
 - get_memory_interface_internal, 27
 - init_attrjeod__BasicJeodTrickSimInterface, 28
 - InputProcessor, 28
 - memory_manager, 29
 - open_checkpoint_file, 27
 - open_restart_file, 27
 - operator=, 27
 - restore_allocations, 27
 - restore_containers, 27
 - section_end, 29
 - section_start, 29
 - set_checkpoint_file_name, 28
 - set_mode, 28
 - trick_memory_interface, 29
- jeod::CheckPointInputManager, 30
 - CheckPointInputManager, 31
 - create_section_reader, 31, 32
 - create_trick_section_reader, 32
 - current_reader, 34
 - deregister_reader, 32
 - filename, 34
 - have_active_reader, 33
 - initialize, 33
 - is_open, 34
 - operator=, 33
 - register_reader, 33
 - section_end, 34
 - section_start, 34
 - sections, 35
 - stream, 35
- jeod::CheckPointInputManager::SectionInfo, 110
 - end_pos, 111
 - SectionInfo, 111
 - start_pos, 111
- jeod::CheckPointOutputManager, 35
 - CheckPointOutputManager, 36, 37
 - create_section_writer, 37
 - create_trick_section_writer, 37
 - current_writer, 39
 - deregister_writer, 38
 - filename, 39
 - have_active_writer, 38
 - is_open, 39
 - MemoryManagerWrapper, 39
 - operator=, 38
 - register_writer, 38
 - section_end, 39
 - section_start, 40
 - stream, 40
- jeod::JeodDynbodyIntegrationLoop, 42
 - ~JeodDynbodyIntegrationLoop, 45
 - add_integrable_object, 45
 - add_sim_object, 45
 - add_sim_object_bodies, 46
 - collect_derivatives, 46
 - deriv_ephem_update, 49
 - dyn_manager, 49
 - find_containing_sim_object, 46
 - gravitation, 46
 - gravity_manager, 49
 - init_attrjeod__JeodDynbodyIntegrationLoop, 49
 - initialize_integ_loop, 47
 - InputProcessor, 49
 - integ_constructor, 49
 - integ_group, 49
 - integ_group_factory, 50
 - integ_interface, 50
 - integrate_dt, 47
 - JeodDynbodyIntegrationLoop, 44, 45
 - loop_sim_object, 50
 - operator=, 47
 - remove_integrable_object, 47
 - remove_sim_object, 47
 - remove_sim_object_bodies, 48
 - set_deriv_ephem_update, 48
 - set_time_to_loop_start, 48
 - time_manager, 50
 - update_integration_group, 48
- jeod::JeodIntegratorInterface, 50
 - ~JeodIntegratorInterface, 51
 - get_integrator, 51

- init_attrjeod__JeodIntegratorInterface, 52
- InputProcessor, 52
- interpret_integration_type, 51
- jeod::JeodMemoryInterface, 52
 - ~JeodMemoryInterface, 53
 - deregister_allocation, 54
 - deregister_container, 54
 - find_attributes, 54
 - get_address_at_name, 55
 - get_name_at_address, 55
 - init_attrjeod__JeodMemoryInterface, 58
 - InputProcessor, 58
 - is_checkpoint_restart_supported, 56
 - JeodMemoryInterface, 53
 - operator=, 56
 - pointer_attributes, 56
 - primitive_attributes, 56
 - register_allocation, 57
 - register_container, 57
 - structure_attributes, 57
 - void_pointer_attributes, 57
- jeod::JeodSimulationInterface, 58
 - ~JeodSimulationInterface, 60
 - configure, 61
 - create_integrator_interface, 61
 - create_integrator_internal, 61
 - get_address_at_name, 61
 - get_checkpoint_reader, 62
 - get_checkpoint_reader_internal, 62
 - get_checkpoint_writer, 62
 - get_checkpoint_writer_internal, 62
 - get_job_cycle, 63
 - get_job_cycle_internal, 63
 - get_memory_interface, 63
 - get_memory_interface_internal, 63
 - get_mode, 63
 - get_name_at_address, 64
 - init_attrjeod__JeodSimulationInterface, 65
 - InputProcessor, 65
 - JeodSimulationInterface, 60
 - Mode, 60
 - mode, 65
 - operator=, 64
 - saved_mode, 65
 - set_mode, 64
 - sim_interface, 65
- jeod::JeodSimulationInterfaceInit, 65
 - JeodSimulationInterfaceInit, 66
 - memory_debug_level, 66
 - message_suppress_id, 66
 - message_suppress_location, 66
 - message_suppression_level, 67
- jeod::JeodTrick10MemoryInterface, 67
 - ~JeodTrick10MemoryInterface, 69
 - checkpoint_allocations, 69
 - checkpoint_containers, 69
 - deregister_container, 69
 - get_address_at_name, 70
 - get_container_id, 70
 - get_name_at_address, 70
 - get_trick_checkpoint_file, 71
 - init_attrjeod__JeodTrick10MemoryInterface, 73
 - InputProcessor, 73
 - is_checkpoint_restart_supported, 71
 - JeodTrick10MemoryInterface, 69
 - operator=, 71
 - register_container, 71
 - restore_allocations, 72
 - restore_containers, 72
 - translate_addr_to_name, 72
 - translate_name_to_addr, 73
 - trick_checkpoint_agent, 73
- jeod::JeodTrickIntegrator, 74
 - ~JeodTrickIntegrator, 75
 - default_first_step_deriv, 77
 - get_dt, 76
 - get_first_step_derivs_flag, 76
 - get_integrator, 76
 - init_attrjeod__JeodTrickIntegrator, 77
 - InputProcessor, 77
 - interpret_integration_type, 76
 - JeodTrickIntegrator, 75
 - operator=, 76
 - reset_first_step_derivs_flag, 76
 - restore_first_step_derivs_flag, 77
 - set_first_step_derivs_flag, 77
 - set_step_number, 77
 - set_time, 77
 - trick_integrator, 78
- jeod::JeodTrickMemoryInterface, 78
 - ~JeodTrickMemoryInterface, 81
 - allocation_map, 88
 - AllocationMap, 80
 - checkpoint_allocations, 81
 - checkpoint_containers, 81
 - construct_identifier, 81
 - container_list, 88
 - ContainerList, 80
 - deregister_allocation, 82
 - deregister_container, 82
 - dlhandle, 88
 - find_attributes, 82, 83
 - get_address_at_name, 83
 - get_name_at_address, 83
 - get_trick_checkpoint_file, 84
 - id_length, 89
 - id_prefix, 89
 - init_attrjeod__JeodTrickMemoryInterface, 88
 - InputProcessor, 88
 - is_checkpoint_restart_supported, 84
 - JeodTrickMemoryInterface, 81
 - mode, 89
 - operator=, 84
 - pointer_attributes, 84
 - primitive_attributes, 85
 - register_allocation, 85

- register_container, 85
- restore_allocations, 87
- restore_containers, 87
- set_mode, 87
- structure_attributes, 87
- void_pointer_attributes, 88
- jeod::JeodTrickMemoryInterface::AllocationMapEntry, 21
 - AllocationMapEntry, 21
 - is_array, 22
 - nelements, 22
 - typeid_info, 22
- jeod::JeodTrickMemoryInterface::ContainerListEntry, 40
 - container, 41
 - ContainerListEntry, 41
 - elem_name, 41
 - owner, 41
 - owner_type, 41
- jeod::JeodTrickSimInterface, 89
 - ~JeodTrickSimInterface, 90
 - init_attrjeod__JeodTrickSimInterface, 91
 - InputProcessor, 91
 - JeodTrickSimInterface, 90
 - operator=, 91
- jeod::SectionedInputBuffer, 91
 - ~SectionedInputBuffer, 92
 - activate, 93
 - at_eof, 94
 - buf, 94
 - curr_pos, 94
 - deactivate, 93
 - end_pos, 94
 - file_buf, 95
 - operator=, 93
 - SectionedInputBuffer, 92, 93
 - SectionedInputStream, 94
 - start_pos, 95
 - underflow, 94
- jeod::SectionedInputStream, 95
 - ~SectionedInputStream, 99
 - activate, 99
 - CheckPointInputManager, 100
 - deactivate, 99
 - end_pos, 101
 - is_activatable, 100
 - is_active, 101
 - is_copy, 101
 - manager, 101
 - operator void *, 100
 - operator=, 100
 - sectbuf, 101
 - SectionedInputStream, 98, 99
 - start_pos, 101
 - stream, 101
- jeod::SectionedOutputBuffer, 102
 - ~SectionedOutputBuffer, 103
 - activate, 103
 - deactivate, 104
 - file_buf, 105
 - operator=, 104
 - overflow, 104
 - SectionedOutputBuffer, 103
 - SectionedOutputStream, 105
- jeod::SectionedOutputStream, 105
 - ~SectionedOutputStream, 107
 - activate, 107
 - CheckPointOutputManager, 109
 - deactivate, 108
 - is_activatable, 108
 - is_active, 109
 - is_copy, 109
 - manager, 109
 - operator void *, 108
 - operator=, 108
 - sectbuf, 109
 - section_end, 109
 - section_start, 110
 - SectionedOutputStream, 107
 - stream, 110
 - tag, 110
- jeod::SimInterfaceMessages, 111
 - implementation_error, 112
 - integration_error, 112
 - interface_error, 112
 - operator=, 112
 - phasing_error, 113
 - SimInterfaceMessages, 112
 - singleton_error, 113
- jeod::TrickJeodIntegrator, 113
 - ~TrickJeodIntegrator, 114
 - initialize, 114
 - integrate, 114
- jeod::TrickMessageHandler, 114
 - ~TrickMessageHandler, 115
 - init_attrjeod__TrickMessageHandler, 116
 - InputProcessor, 116
 - operator=, 116
 - process_message, 116
 - register_contents, 116
 - TrickMessageHandler, 115
- jeod::TrickMessageHandlerMixin, 116
 - ~TrickMessageHandlerMixin, 117
 - init_attrjeod__TrickMessageHandlerMixin, 118
 - InputProcessor, 118
 - message_handler, 118
 - operator=, 118
 - TrickMessageHandlerMixin, 117
- jeod_class.hh, 123
- jeod_integrator_interface.hh, 123
- jeod_trick_integrator.hh, 124
- JeodDynbodyIntegrationLoop
 - jeod::JeodDynbodyIntegrationLoop, 44, 45
- JeodMemoryInterface
 - jeod::JeodMemoryInterface, 53
- JeodSimulationInterface
 - jeod::JeodSimulationInterface, 60

- JeodSimulationInterfaceInit
 - jeod::JeodSimulationInterfaceInit, 66
- JeodTrick10MemoryInterface
 - jeod::JeodTrick10MemoryInterface, 69
- JeodTrickIntegrator
 - jeod::JeodTrickIntegrator, 75
- JeodTrickMemoryInterface
 - jeod::JeodTrickMemoryInterface, 81
- JeodTrickSimInterface
 - jeod::JeodTrickSimInterface, 90
- loop_sim_object
 - jeod::JeodDynbodyIntegrationLoop, 50
- MAKE_MESSAGE_CODE
 - SimInterface, 17
- MAX_MSG_SIZE
 - SimInterface, 17
- manager
 - jeod::SectionedInputStream, 101
 - jeod::SectionedOutputStream, 109
- memory_attributes.hh, 124
 - JEOD_ATTRIBUTES, 125
- memory_debug_level
 - jeod::JeodSimulationInterfaceInit, 66
- memory_interface.cc, 125
- memory_interface.hh, 126
- memory_manager
 - jeod::BasicJeodTrickSimInterface, 29
- MemoryManagerWrapper
 - jeod::CheckPointOutputManager, 39
- message_handler
 - jeod::TrickMessageHandlerMixin, 118
- message_suppress_id
 - jeod::JeodSimulationInterfaceInit, 66
- message_suppress_location
 - jeod::JeodSimulationInterfaceInit, 66
- message_suppression_level
 - jeod::JeodSimulationInterfaceInit, 67
- Mode
 - jeod::JeodSimulationInterface, 60
- mode
 - jeod::JeodSimulationInterface, 65
 - jeod::JeodTrickMemoryInterface, 89
- Models, 11
- nelements
 - jeod::JeodTrickMemoryInterface::AllocationMap-Entry, 22
- NumModes
 - jeod::JeodSimulationInterface, 60
- open_checkpoint_file
 - jeod::BasicJeodTrickSimInterface, 27
- open_restart_file
 - jeod::BasicJeodTrickSimInterface, 27
- Operational
 - jeod::JeodSimulationInterface, 60
- operator void *
- jeod::SectionedInputStream, 100
- jeod::SectionedOutputStream, 108
- operator=
 - jeod::BasicJeodTrickSimInterface, 27
 - jeod::CheckPointInputManager, 33
 - jeod::CheckPointOutputManager, 38
 - jeod::JeodDynbodyIntegrationLoop, 47
 - jeod::JeodMemoryInterface, 56
 - jeod::JeodSimulationInterface, 64
 - jeod::JeodTrick10MemoryInterface, 71
 - jeod::JeodTrickIntegrator, 76
 - jeod::JeodTrickMemoryInterface, 84
 - jeod::JeodTrickSimInterface, 91
 - jeod::SectionedInputBuffer, 93
 - jeod::SectionedInputStream, 100
 - jeod::SectionedOutputBuffer, 104
 - jeod::SectionedOutputStream, 108
 - jeod::SimInterfaceMessages, 112
 - jeod::TrickMessageHandler, 116
 - jeod::TrickMessageHandlerMixin, 118
- overflow
 - jeod::SectionedOutputBuffer, 104
- owner
 - jeod::JeodTrickMemoryInterface::ContainerList-Entry, 41
- owner_type
 - jeod::JeodTrickMemoryInterface::ContainerList-Entry, 41
- PATH
 - SimInterface, 17
- phasing_error
 - jeod::SimInterfaceMessages, 113
- pointer_attributes
 - jeod::JeodMemoryInterface, 56
 - jeod::JeodTrickMemoryInterface, 84
- PostCheckpoint
 - jeod::JeodSimulationInterface, 60
- PreCheckpoint
 - jeod::JeodSimulationInterface, 60
- primitive_attributes
 - jeod::JeodMemoryInterface, 56
 - jeod::JeodTrickMemoryInterface, 85
- process_message
 - jeod::TrickMessageHandler, 116
- register_allocation
 - jeod::JeodMemoryInterface, 57
 - jeod::JeodTrickMemoryInterface, 85
- register_container
 - jeod::JeodMemoryInterface, 57
 - jeod::JeodTrick10MemoryInterface, 71
 - jeod::JeodTrickMemoryInterface, 85
- register_contents
 - jeod::TrickMessageHandler, 116
- register_reader
 - jeod::CheckPointInputManager, 33
- register_writer
 - jeod::CheckPointOutputManager, 38

- remove_integrable_object
 - jeod::JeodDynbodyIntegrationLoop, 47
- remove_sim_object
 - jeod::JeodDynbodyIntegrationLoop, 47
- remove_sim_object_bodies
 - jeod::JeodDynbodyIntegrationLoop, 48
- reset_first_step_derivs_flag
 - jeod::JeodTrickIntegrator, 76
- Restart
 - jeod::JeodSimulationInterface, 60
- Restore
 - jeod::JeodSimulationInterface, 60
- restore_allocations
 - jeod::BasicJeodTrickSimInterface, 27
 - jeod::JeodTrick10MemoryInterface, 72
 - jeod::JeodTrickMemoryInterface, 87
- restore_containers
 - jeod::BasicJeodTrickSimInterface, 27
 - jeod::JeodTrick10MemoryInterface, 72
 - jeod::JeodTrickMemoryInterface, 87
- restore_first_step_derivs_flag
 - jeod::JeodTrickIntegrator, 77
- saved_mode
 - jeod::JeodSimulationInterface, 65
- sectbuf
 - jeod::SectionedInputStream, 101
 - jeod::SectionedOutputStream, 109
- section_end
 - jeod::BasicJeodTrickSimInterface, 29
 - jeod::CheckPointInputManager, 34
 - jeod::CheckPointOutputManager, 39
 - jeod::SectionedOutputStream, 109
- section_start
 - jeod::BasicJeodTrickSimInterface, 29
 - jeod::CheckPointInputManager, 34
 - jeod::CheckPointOutputManager, 40
 - jeod::SectionedOutputStream, 110
- SectionInfo
 - jeod::CheckPointInputManager::SectionInfo, 111
- SectionedInputBuffer
 - jeod::SectionedInputBuffer, 92, 93
- SectionedInputStream
 - jeod::SectionedInputBuffer, 94
 - jeod::SectionedInputStream, 98, 99
- SectionedOutputBuffer
 - jeod::SectionedOutputBuffer, 103
- SectionedOutputStream
 - jeod::SectionedOutputBuffer, 105
 - jeod::SectionedOutputStream, 107
- sections
 - jeod::CheckPointInputManager, 35
- set_checkpoint_file_name
 - jeod::BasicJeodTrickSimInterface, 28
- set_deriv_ephem_update
 - jeod::JeodDynbodyIntegrationLoop, 48
- set_first_step_derivs_flag
 - jeod::JeodTrickIntegrator, 77
- set_mode
 - jeod::BasicJeodTrickSimInterface, 28
 - jeod::JeodSimulationInterface, 64
 - jeod::JeodTrickMemoryInterface, 87
- set_step_number
 - jeod::JeodTrickIntegrator, 77
- set_time
 - jeod::JeodTrickIntegrator, 77
- set_time_to_loop_start
 - jeod::JeodDynbodyIntegrationLoop, 48
- Shutdown
 - jeod::JeodSimulationInterface, 60
- sim_interface
 - jeod::JeodSimulationInterface, 65
- sim_interface_messages.cc, 126
- sim_interface_messages.hh, 127
- SimInterface, 13
 - CLASS, 15
 - ER7_UTILS_RESTRICT, 15
 - ER7_UTILS_UNUSED, 15
 - JEOD_INTPTR_T, 16
 - JEOD_PTRDIFF_T, 16
 - JEOD_SIZE_T, 17
 - JEOD_UINTPTR_T, 17
 - JEOD_UNUSED, 17
 - MAKE_MESSAGE_CODE, 17
 - MAX_MSG_SIZE, 17
 - PATH, 17
 - trick_MM, 17, 18
 - trick_curr_integ, 17
- SimInterfaceMessages
 - jeod::SimInterfaceMessages, 112
- simulation_interface.cc, 127
- simulation_interface.hh, 128
- singleton_error
 - jeod::SimInterfaceMessages, 113
- start_pos
 - jeod::CheckPointInputManager::SectionInfo, 111
 - jeod::SectionedInputBuffer, 95
 - jeod::SectionedInputStream, 101
- stream
 - jeod::CheckPointInputManager, 35
 - jeod::CheckPointOutputManager, 40
 - jeod::SectionedInputStream, 101
 - jeod::SectionedOutputStream, 110
- structure_attributes
 - jeod::JeodMemoryInterface, 57
 - jeod::JeodTrickMemoryInterface, 87
- tag
 - jeod::SectionedOutputStream, 110
- time_manager
 - jeod::JeodDynbodyIntegrationLoop, 50
- translate_addr_to_name
 - jeod::JeodTrick10MemoryInterface, 72
- translate_name_to_addr
 - jeod::JeodTrick10MemoryInterface, 73
- Trick, 20
- trick10_memory_interface.cc, 128
- trick10_memory_interface.hh, 129

- trick_MM
 - SimInterface, [17](#), [18](#)
- trick_checkpoint_agent
 - jeod::JeodTrick10MemoryInterface, [73](#)
- trick_curr_integ
 - SimInterface, [17](#)
- trick_dynbody_integ_loop.cc, [130](#)
- trick_dynbody_integ_loop.hh, [130](#)
- trick_integrator
 - jeod::JeodTrickIntegrator, [78](#)
- trick_memory_interface
 - jeod::BasicJeodTrickSimInterface, [29](#)
- trick_memory_interface.cc, [131](#)
- trick_memory_interface.hh, [131](#)
- trick_memory_interface_alloc.cc, [132](#)
- trick_memory_interface_attrib.cc, [133](#)
- trick_memory_interface_chkpnt.cc, [133](#)
- trick_memory_interface_xlate.cc, [134](#)
- trick_message_handler.cc, [135](#)
- trick_message_handler.hh, [135](#)
- trick_sim_interface.cc, [136](#)
- trick_sim_interface.hh, [136](#)
- TrickMessageHandler
 - jeod::TrickMessageHandler, [115](#)
- TrickMessageHandlerMixin
 - jeod::TrickMessageHandlerMixin, [117](#)
- typeid_info
 - jeod::JeodTrickMemoryInterface::AllocationMapEntry, [22](#)
- underflow
 - jeod::SectionedInputBuffer, [94](#)
- update_integration_group
 - jeod::JeodDynbodyIntegrationLoop, [48](#)
- Utils, [12](#)
- void_pointer_attributes
 - jeod::JeodMemoryInterface, [57](#)
 - jeod::JeodTrickMemoryInterface, [88](#)