

# SimulationInterfaceMacro

5.1

Generated by Doxygen 1.8.5

Mon Jul 31 2023 11:39:12



# Contents

<b>1</b>	<b>Module Index</b>	<b>1</b>
1.1	Modules . . . . .	1
<b>2</b>	<b>Namespace Index</b>	<b>3</b>
2.1	Namespace List . . . . .	3
<b>3</b>	<b>Hierarchical Index</b>	<b>5</b>
3.1	Class Hierarchy . . . . .	5
<b>4</b>	<b>Data Structure Index</b>	<b>7</b>
4.1	Data Structures . . . . .	7
<b>5</b>	<b>File Index</b>	<b>9</b>
5.1	File List . . . . .	9
<b>6</b>	<b>Module Documentation</b>	<b>11</b>
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
<b>7</b>	<b>Namespace Documentation</b>	<b>19</b>
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
<b>8</b>	<b>Data Structure Documentation</b>	<b>21</b>
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	<a href="#">create_section_reader</a>	31
8.3.3.2	<a href="#">create_section_reader</a>	32
8.3.3.3	<a href="#">create_trick_section_reader</a>	32
8.3.3.4	<a href="#">deregister_reader</a>	33
8.3.3.5	<a href="#">have_active_reader</a>	33
8.3.3.6	<a href="#">initialize</a>	33
8.3.3.7	<a href="#">operator!</a>	33
8.3.3.8	<a href="#">operator=</a>	33
8.3.3.9	<a href="#">register_reader</a>	34
8.3.4	<a href="#">Field Documentation</a>	34
8.3.4.1	<a href="#">current_reader</a>	34
8.3.4.2	<a href="#">filename</a>	34
8.3.4.3	<a href="#">is_open</a>	34
8.3.4.4	<a href="#">section_end</a>	34
8.3.4.5	<a href="#">section_start</a>	34
8.3.4.6	<a href="#">sections</a>	35
8.3.4.7	<a href="#">stream</a>	35
8.4	<a href="#">jeod::CheckPointOutputManager Class Reference</a>	35
8.4.1	<a href="#">Detailed Description</a>	36
8.4.2	<a href="#">Constructor &amp; Destructor Documentation</a>	36
8.4.2.1	<a href="#">CheckPointOutputManager</a>	36
8.4.2.2	<a href="#">CheckPointOutputManager</a>	36
8.4.3	<a href="#">Member Function Documentation</a>	36
8.4.3.1	<a href="#">create_section_writer</a>	36
8.4.3.2	<a href="#">create_section_writer</a>	37
8.4.3.3	<a href="#">create_trick_section_writer</a>	37
8.4.3.4	<a href="#">deregister_writer</a>	37
8.4.3.5	<a href="#">have_active_writer</a>	38
8.4.3.6	<a href="#">operator!</a>	38
8.4.3.7	<a href="#">operator=</a>	38
8.4.3.8	<a href="#">register_writer</a>	38
8.4.4	<a href="#">Friends And Related Function Documentation</a>	39
8.4.4.1	<a href="#">MemoryManagerWrapper</a>	39
8.4.5	<a href="#">Field Documentation</a>	39
8.4.5.1	<a href="#">current_writer</a>	39
8.4.5.2	<a href="#">filename</a>	39
8.4.5.3	<a href="#">is_open</a>	39
8.4.5.4	<a href="#">section_end</a>	39
8.4.5.5	<a href="#">section_start</a>	39
8.4.5.6	<a href="#">stream</a>	39

8.5	<a href="#">jeod::JeodTrickMemoryInterface::ContainerListEntry Struct Reference</a>	40
8.5.1	<a href="#">Detailed Description</a>	40
8.5.2	<a href="#">Constructor &amp; Destructor Documentation</a>	40
8.5.2.1	<a href="#">ContainerListEntry</a>	40
8.5.3	<a href="#">Field Documentation</a>	41
8.5.3.1	<a href="#">container</a>	41
8.5.3.2	<a href="#">elem_name</a>	41
8.5.3.3	<a href="#">owner</a>	41
8.5.3.4	<a href="#">owner_type</a>	41
8.6	<a href="#">jeod::JeodDynbodyIntegrationLoop Class Reference</a>	41
8.6.1	<a href="#">Detailed Description</a>	43
8.6.2	<a href="#">Constructor &amp; Destructor Documentation</a>	44
8.6.2.1	<a href="#">JeodDynbodyIntegrationLoop</a>	44
8.6.2.2	<a href="#">JeodDynbodyIntegrationLoop</a>	44
8.6.2.3	<a href="#">~JeodDynbodyIntegrationLoop</a>	44
8.6.2.4	<a href="#">JeodDynbodyIntegrationLoop</a>	44
8.6.3	<a href="#">Member Function Documentation</a>	45
8.6.3.1	<a href="#">add_integrable_object</a>	45
8.6.3.2	<a href="#">add_sim_object</a>	46
8.6.3.3	<a href="#">add_sim_object_bodies</a>	46
8.6.3.4	<a href="#">add_sim_object_bodies</a>	46
8.6.3.5	<a href="#">collect_derivatives</a>	46
8.6.3.6	<a href="#">find_containing_sim_object</a>	47
8.6.3.7	<a href="#">gravitation</a>	48
8.6.3.8	<a href="#">initialize_integ_loop</a>	48
8.6.3.9	<a href="#">integrate_dt</a>	48
8.6.3.10	<a href="#">operator=</a>	48
8.6.3.11	<a href="#">remove_integrable_object</a>	48
8.6.3.12	<a href="#">remove_sim_object</a>	49
8.6.3.13	<a href="#">remove_sim_object_bodies</a>	49
8.6.3.14	<a href="#">set_deriv_ephem_update</a>	49
8.6.3.15	<a href="#">set_time_to_loop_start</a>	49
8.6.3.16	<a href="#">update_integration_group</a>	50
8.6.4	<a href="#">Friends And Related Function Documentation</a>	50
8.6.4.1	<a href="#">init_attrjeod__JeodDynbodyIntegrationLoop</a>	50
8.6.4.2	<a href="#">InputProcessor</a>	50
8.6.5	<a href="#">Field Documentation</a>	50
8.6.5.1	<a href="#">deriv_ephem_update</a>	50
8.6.5.2	<a href="#">dyn_manager</a>	50
8.6.5.3	<a href="#">gravity_manager</a>	50

8.6.5.4	<a href="#">integ_constructor</a>	51
8.6.5.5	<a href="#">integ_group</a>	51
8.6.5.6	<a href="#">integ_group_factory</a>	51
8.6.5.7	<a href="#">integ_interface</a>	51
8.6.5.8	<a href="#">loop_sim_object</a>	51
8.6.5.9	<a href="#">time_manager</a>	51
8.7	<a href="#">jeod::JeodIntegratorInterface Class Reference</a>	52
8.7.1	<a href="#">Detailed Description</a>	52
8.7.2	<a href="#">Constructor &amp; Destructor Documentation</a>	52
8.7.2.1	<a href="#">~JeodIntegratorInterface</a>	52
8.7.3	<a href="#">Member Function Documentation</a>	53
8.7.3.1	<a href="#">get_integrator</a>	53
8.7.3.2	<a href="#">interpret_integration_type</a>	53
8.7.4	<a href="#">Friends And Related Function Documentation</a>	53
8.7.4.1	<a href="#">init_attrjeod__JeodIntegratorInterface</a>	53
8.7.4.2	<a href="#">InputProcessor</a>	53
8.8	<a href="#">jeod::JeodMemoryInterface Class Reference</a>	53
8.8.1	<a href="#">Detailed Description</a>	54
8.8.2	<a href="#">Constructor &amp; Destructor Documentation</a>	54
8.8.2.1	<a href="#">JeodMemoryInterface</a>	54
8.8.2.2	<a href="#">~JeodMemoryInterface</a>	55
8.8.2.3	<a href="#">JeodMemoryInterface</a>	55
8.8.3	<a href="#">Member Function Documentation</a>	55
8.8.3.1	<a href="#">deregister_allocation</a>	55
8.8.3.2	<a href="#">deregister_container</a>	55
8.8.3.3	<a href="#">find_attributes</a>	55
8.8.3.4	<a href="#">find_attributes</a>	56
8.8.3.5	<a href="#">get_address_at_name</a>	56
8.8.3.6	<a href="#">get_name_at_address</a>	56
8.8.3.7	<a href="#">is_checkpoint_restart_supported</a>	57
8.8.3.8	<a href="#">operator=</a>	57
8.8.3.9	<a href="#">pointer_attributes</a>	57
8.8.3.10	<a href="#">primitive_attributes</a>	57
8.8.3.11	<a href="#">register_allocation</a>	58
8.8.3.12	<a href="#">register_container</a>	58
8.8.3.13	<a href="#">structure_attributes</a>	58
8.8.3.14	<a href="#">void_pointer_attributes</a>	58
8.8.4	<a href="#">Friends And Related Function Documentation</a>	59
8.8.4.1	<a href="#">init_attrjeod__JeodMemoryInterface</a>	59
8.8.4.2	<a href="#">InputProcessor</a>	59



8.9	<a href="#">jeod::JeodSimulationInterface Class Reference</a>	59
8.9.1	<a href="#">Detailed Description</a>	61
8.9.2	<a href="#">Member Enumeration Documentation</a>	61
8.9.2.1	<a href="#">Mode</a>	61
8.9.3	<a href="#">Constructor &amp; Destructor Documentation</a>	61
8.9.3.1	<a href="#">JeodSimulationInterface</a>	61
8.9.3.2	<a href="#">~JeodSimulationInterface</a>	61
8.9.3.3	<a href="#">JeodSimulationInterface</a>	62
8.9.4	<a href="#">Member Function Documentation</a>	62
8.9.4.1	<a href="#">configure</a>	62
8.9.4.2	<a href="#">create_integrator_interface</a>	62
8.9.4.3	<a href="#">create_integrator_internal</a>	62
8.9.4.4	<a href="#">get_address_at_name</a>	62
8.9.4.5	<a href="#">get_checkpoint_reader</a>	63
8.9.4.6	<a href="#">get_checkpoint_reader_internal</a>	63
8.9.4.7	<a href="#">get_checkpoint_writer</a>	63
8.9.4.8	<a href="#">get_checkpoint_writer_internal</a>	63
8.9.4.9	<a href="#">get_job_cycle</a>	64
8.9.4.10	<a href="#">get_job_cycle_internal</a>	64
8.9.4.11	<a href="#">get_memory_interface</a>	64
8.9.4.12	<a href="#">get_memory_interface_internal</a>	64
8.9.4.13	<a href="#">get_mode</a>	65
8.9.4.14	<a href="#">get_name_at_address</a>	65
8.9.4.15	<a href="#">operator=</a>	65
8.9.4.16	<a href="#">set_mode</a>	65
8.9.5	<a href="#">Friends And Related Function Documentation</a>	66
8.9.5.1	<a href="#">init_attrjeod__JeodSimulationInterface</a>	66
8.9.5.2	<a href="#">InputProcessor</a>	66
8.9.6	<a href="#">Field Documentation</a>	66
8.9.6.1	<a href="#">mode</a>	66
8.9.6.2	<a href="#">saved_mode</a>	66
8.9.6.3	<a href="#">sim_interface</a>	66
8.10	<a href="#">jeod::JeodSimulationInterfaceInit Class Reference</a>	66
8.10.1	<a href="#">Detailed Description</a>	67
8.10.2	<a href="#">Constructor &amp; Destructor Documentation</a>	67
8.10.2.1	<a href="#">JeodSimulationInterfaceInit</a>	67
8.10.3	<a href="#">Field Documentation</a>	67
8.10.3.1	<a href="#">memory_debug_level</a>	67
8.10.3.2	<a href="#">message_suppress_id</a>	67
8.10.3.3	<a href="#">message_suppress_location</a>	68

8.10.3.4	<a href="#">message_suppression_level</a>	68
8.11	<a href="#">jeod::JeodTrick10MemoryInterface Class Reference</a>	68
8.11.1	<a href="#">Detailed Description</a>	69
8.11.2	<a href="#">Constructor &amp; Destructor Documentation</a>	70
8.11.2.1	<a href="#">JeodTrick10MemoryInterface</a>	70
8.11.2.2	<a href="#">~JeodTrick10MemoryInterface</a>	70
8.11.2.3	<a href="#">JeodTrick10MemoryInterface</a>	70
8.11.3	<a href="#">Member Function Documentation</a>	70
8.11.3.1	<a href="#">checkpoint_allocations</a>	70
8.11.3.2	<a href="#">checkpoint_containers</a>	70
8.11.3.3	<a href="#">deregister_container</a>	70
8.11.3.4	<a href="#">get_address_at_name</a>	71
8.11.3.5	<a href="#">get_container_id</a>	71
8.11.3.6	<a href="#">get_name_at_address</a>	72
8.11.3.7	<a href="#">get_trick_checkpoint_file</a>	72
8.11.3.8	<a href="#">is_checkpoint_restart_supported</a>	72
8.11.3.9	<a href="#">operator=</a>	72
8.11.3.10	<a href="#">register_container</a>	73
8.11.3.11	<a href="#">restore_allocations</a>	73
8.11.3.12	<a href="#">restore_containers</a>	73
8.11.3.13	<a href="#">translate_addr_to_name</a>	74
8.11.3.14	<a href="#">translate_name_to_addr</a>	74
8.11.4	<a href="#">Friends And Related Function Documentation</a>	74
8.11.4.1	<a href="#">init_attrjeod__JeodTrick10MemoryInterface</a>	74
8.11.4.2	<a href="#">InputProcessor</a>	74
8.11.5	<a href="#">Field Documentation</a>	74
8.11.5.1	<a href="#">trick_checkpoint_agent</a>	75
8.12	<a href="#">jeod::JeodTrickIntegrator Class Reference</a>	75
8.12.1	<a href="#">Detailed Description</a>	76
8.12.2	<a href="#">Constructor &amp; Destructor Documentation</a>	76
8.12.2.1	<a href="#">JeodTrickIntegrator</a>	76
8.12.2.2	<a href="#">~JeodTrickIntegrator</a>	76
8.12.2.3	<a href="#">JeodTrickIntegrator</a>	76
8.12.3	<a href="#">Member Function Documentation</a>	77
8.12.3.1	<a href="#">get_dt</a>	77
8.12.3.2	<a href="#">get_first_step_derivs_flag</a>	77
8.12.3.3	<a href="#">get_integrator</a>	77
8.12.3.4	<a href="#">interpret_integration_type</a>	77
8.12.3.5	<a href="#">operator=</a>	77
8.12.3.6	<a href="#">reset_first_step_derivs_flag</a>	77

8.12.3.7	<a href="#">restore_first_step_derivs_flag</a>	78
8.12.3.8	<a href="#">set_first_step_derivs_flag</a>	78
8.12.3.9	<a href="#">set_step_number</a>	78
8.12.3.10	<a href="#">set_time</a>	78
8.12.4	<a href="#">Friends And Related Function Documentation</a>	78
8.12.4.1	<a href="#">init_attrjeod__JeodTrickIntegrator</a>	78
8.12.4.2	<a href="#">InputProcessor</a>	78
8.12.5	<a href="#">Field Documentation</a>	78
8.12.5.1	<a href="#">default_first_step_deriv</a>	79
8.12.5.2	<a href="#">trick_integrator</a>	79
8.13	<a href="#">jeod::JeodTrickMemoryInterface Class Reference</a>	79
8.13.1	<a href="#">Detailed Description</a>	81
8.13.2	<a href="#">Member Typedef Documentation</a>	81
8.13.2.1	<a href="#">AllocationMap</a>	81
8.13.2.2	<a href="#">ContainerList</a>	82
8.13.3	<a href="#">Constructor &amp; Destructor Documentation</a>	82
8.13.3.1	<a href="#">JeodTrickMemoryInterface</a>	82
8.13.3.2	<a href="#">~JeodTrickMemoryInterface</a>	82
8.13.3.3	<a href="#">JeodTrickMemoryInterface</a>	82
8.13.4	<a href="#">Member Function Documentation</a>	82
8.13.4.1	<a href="#">checkpoint_allocations</a>	82
8.13.4.2	<a href="#">checkpoint_containers</a>	82
8.13.4.3	<a href="#">construct_identifier</a>	83
8.13.4.4	<a href="#">deregister_allocation</a>	83
8.13.4.5	<a href="#">deregister_container</a>	83
8.13.4.6	<a href="#">find_attributes</a>	84
8.13.4.7	<a href="#">find_attributes</a>	84
8.13.4.8	<a href="#">get_address_at_name</a>	84
8.13.4.9	<a href="#">get_name_at_address</a>	84
8.13.4.10	<a href="#">get_trick_checkpoint_file</a>	85
8.13.4.11	<a href="#">is_checkpoint_restart_supported</a>	85
8.13.4.12	<a href="#">operator=</a>	85
8.13.4.13	<a href="#">pointer_attributes</a>	85
8.13.4.14	<a href="#">primitive_attributes</a>	86
8.13.4.15	<a href="#">register_allocation</a>	86
8.13.4.16	<a href="#">register_container</a>	86
8.13.4.17	<a href="#">restore_allocations</a>	88
8.13.4.18	<a href="#">restore_containers</a>	88
8.13.4.19	<a href="#">set_mode</a>	88
8.13.4.20	<a href="#">structure_attributes</a>	88

8.13.4.21	<code>void_pointer_attributes</code>	89
8.13.5	Friends And Related Function Documentation	89
8.13.5.1	<code>init_attrjeod__JeodTrickMemoryInterface</code>	89
8.13.5.2	<code>InputProcessor</code>	89
8.13.6	Field Documentation	89
8.13.6.1	<code>allocation_map</code>	89
8.13.6.2	<code>container_list</code>	89
8.13.6.3	<code>dlhandle</code>	90
8.13.6.4	<code>id_length</code>	90
8.13.6.5	<code>id_prefix</code>	90
8.13.6.6	<code>mode</code>	90
8.14	<code>jeod::JeodTrickSimInterface</code> Class Reference	90
8.14.1	Detailed Description	91
8.14.2	Constructor & Destructor Documentation	91
8.14.2.1	<code>JeodTrickSimInterface</code>	91
8.14.2.2	<code>~JeodTrickSimInterface</code>	91
8.14.2.3	<code>JeodTrickSimInterface</code>	92
8.14.3	Member Function Documentation	92
8.14.3.1	<code>operator=</code>	92
8.14.4	Friends And Related Function Documentation	92
8.14.4.1	<code>init_attrjeod__JeodTrickSimInterface</code>	92
8.14.4.2	<code>InputProcessor</code>	92
8.15	<code>jeod::SectionedInputBuffer</code> Class Reference	92
8.15.1	Detailed Description	93
8.15.2	Constructor & Destructor Documentation	93
8.15.2.1	<code>~SectionedInputBuffer</code>	93
8.15.2.2	<code>SectionedInputBuffer</code>	94
8.15.2.3	<code>SectionedInputBuffer</code>	94
8.15.3	Member Function Documentation	94
8.15.3.1	<code>activate</code>	94
8.15.3.2	<code>deactivate</code>	94
8.15.3.3	<code>operator!</code>	94
8.15.3.4	<code>operator=</code>	95
8.15.3.5	<code>underflow</code>	95
8.15.4	Friends And Related Function Documentation	95
8.15.4.1	<code>SectionedInputStream</code>	95
8.15.5	Field Documentation	95
8.15.5.1	<code>at_eof</code>	95
8.15.5.2	<code>buf</code>	95
8.15.5.3	<code>curr_pos</code>	95

8.15.5.4	<code>end_pos</code>	96
8.15.5.5	<code>file_buf</code>	96
8.15.5.6	<code>start_pos</code>	96
8.16	<code>jeod::SectionedInputStream</code> Class Reference	96
8.16.1	Detailed Description	98
8.16.2	Constructor & Destructor Documentation	99
8.16.2.1	<code>SectionedInputStream</code>	99
8.16.2.2	<code>SectionedInputStream</code>	99
8.16.2.3	<code>~SectionedInputStream</code>	100
8.16.2.4	<code>SectionedInputStream</code>	100
8.16.3	Member Function Documentation	100
8.16.3.1	<code>activate</code>	100
8.16.3.2	<code>deactivate</code>	100
8.16.3.3	<code>is_activatable</code>	101
8.16.3.4	<code>operator void *</code>	101
8.16.3.5	<code>operator!</code>	101
8.16.3.6	<code>operator=</code>	101
8.16.4	Friends And Related Function Documentation	101
8.16.4.1	<code>CheckpointInputManager</code>	101
8.16.5	Field Documentation	102
8.16.5.1	<code>end_pos</code>	102
8.16.5.2	<code>is_active</code>	102
8.16.5.3	<code>is_copy</code>	102
8.16.5.4	<code>manager</code>	102
8.16.5.5	<code>sectbuf</code>	102
8.16.5.6	<code>start_pos</code>	102
8.16.5.7	<code>stream</code>	103
8.17	<code>jeod::SectionedOutputBuffer</code> Class Reference	103
8.17.1	Detailed Description	104
8.17.2	Constructor & Destructor Documentation	104
8.17.2.1	<code>~SectionedOutputBuffer</code>	104
8.17.2.2	<code>SectionedOutputBuffer</code>	104
8.17.2.3	<code>SectionedOutputBuffer</code>	104
8.17.2.4	<code>SectionedOutputBuffer</code>	104
8.17.3	Member Function Documentation	104
8.17.3.1	<code>activate</code>	104
8.17.3.2	<code>deactivate</code>	105
8.17.3.3	<code>operator!</code>	105
8.17.3.4	<code>operator=</code>	105
8.17.3.5	<code>overflow</code>	105

8.17.4	Friends And Related Function Documentation	106
8.17.4.1	SectionedOutputStream	106
8.17.5	Field Documentation	106
8.17.5.1	file_buf	106
8.18	jeod::SectionedOutputStream Class Reference	106
8.18.1	Detailed Description	107
8.18.2	Constructor & Destructor Documentation	108
8.18.2.1	SectionedOutputStream	108
8.18.2.2	SectionedOutputStream	108
8.18.2.3	~SectionedOutputStream	108
8.18.2.4	SectionedOutputStream	108
8.18.3	Member Function Documentation	108
8.18.3.1	activate	108
8.18.3.2	deactivate	109
8.18.3.3	is_activatable	109
8.18.3.4	operator void *	109
8.18.3.5	operator!	109
8.18.3.6	operator=	110
8.18.4	Friends And Related Function Documentation	110
8.18.4.1	CheckPointOutputManager	110
8.18.5	Field Documentation	110
8.18.5.1	is_active	110
8.18.5.2	is_copy	110
8.18.5.3	manager	110
8.18.5.4	sectbuf	110
8.18.5.5	section_end	110
8.18.5.6	section_start	111
8.18.5.7	stream	111
8.18.5.8	tag	111
8.19	jeod::CheckPointInputManager::SectionInfo Struct Reference	111
8.19.1	Detailed Description	111
8.19.2	Constructor & Destructor Documentation	112
8.19.2.1	SectionInfo	112
8.19.3	Field Documentation	113
8.19.3.1	end_pos	113
8.19.3.2	start_pos	113
8.20	jeod::SimInterfaceMessages Class Reference	113
8.20.1	Detailed Description	114
8.20.2	Constructor & Destructor Documentation	114
8.20.2.1	SimInterfaceMessages	114

8.20.2.2	<a href="#">SimInterfaceMessages</a>	114
8.20.3	<a href="#">Member Function Documentation</a>	114
8.20.3.1	<a href="#">operator=</a>	114
8.20.4	<a href="#">Field Documentation</a>	114
8.20.4.1	<a href="#">implementation_error</a>	114
8.20.4.2	<a href="#">integration_error</a>	114
8.20.4.3	<a href="#">interface_error</a>	114
8.20.4.4	<a href="#">phasing_error</a>	115
8.20.4.5	<a href="#">singleton_error</a>	115
8.21	<a href="#">jeod::TrickJeodIntegrator Class Reference</a>	115
8.21.1	<a href="#">Detailed Description</a>	116
8.21.2	<a href="#">Constructor &amp; Destructor Documentation</a>	116
8.21.2.1	<a href="#">~TrickJeodIntegrator</a>	116
8.21.3	<a href="#">Member Function Documentation</a>	116
8.21.3.1	<a href="#">initialize</a>	116
8.21.3.2	<a href="#">integrate</a>	116
8.22	<a href="#">jeod::TrickMessageHandler Class Reference</a>	116
8.22.1	<a href="#">Detailed Description</a>	117
8.22.2	<a href="#">Constructor &amp; Destructor Documentation</a>	117
8.22.2.1	<a href="#">TrickMessageHandler</a>	117
8.22.2.2	<a href="#">~TrickMessageHandler</a>	117
8.22.2.3	<a href="#">TrickMessageHandler</a>	117
8.22.3	<a href="#">Member Function Documentation</a>	117
8.22.3.1	<a href="#">operator=</a>	117
8.22.3.2	<a href="#">process_message</a>	118
8.22.3.3	<a href="#">register_contents</a>	118
8.22.4	<a href="#">Friends And Related Function Documentation</a>	118
8.22.4.1	<a href="#">init_attrjeod__TrickMessageHandler</a>	118
8.22.4.2	<a href="#">InputProcessor</a>	118
8.23	<a href="#">jeod::TrickMessageHandlerMixin Class Reference</a>	118
8.23.1	<a href="#">Detailed Description</a>	119
8.23.2	<a href="#">Constructor &amp; Destructor Documentation</a>	119
8.23.2.1	<a href="#">TrickMessageHandlerMixin</a>	119
8.23.2.2	<a href="#">~TrickMessageHandlerMixin</a>	119
8.23.2.3	<a href="#">TrickMessageHandlerMixin</a>	119
8.23.3	<a href="#">Member Function Documentation</a>	120
8.23.3.1	<a href="#">operator=</a>	120
8.23.4	<a href="#">Friends And Related Function Documentation</a>	120
8.23.4.1	<a href="#">init_attrjeod__TrickMessageHandlerMixin</a>	120
8.23.4.2	<a href="#">InputProcessor</a>	120

8.23.5	Field Documentation	120
8.23.5.1	message_handler	120
<b>9</b>	<b>File Documentation</b>	<b>121</b>
9.1	checkpoint_input_manager.cc File Reference	121
9.1.1	Detailed Description	121
9.2	checkpoint_input_manager.hh File Reference	121
9.2.1	Detailed Description	122
9.3	checkpoint_output_manager.cc File Reference	122
9.3.1	Detailed Description	122
9.4	checkpoint_output_manager.hh File Reference	122
9.4.1	Detailed Description	123
9.5	class_declarations.hh File Reference	123
9.5.1	Detailed Description	123
9.6	config.hh File Reference	123
9.6.1	Detailed Description	124
9.7	config_test_harness.hh File Reference	124
9.7.1	Detailed Description	124
9.8	config_trick10.hh File Reference	124
9.8.1	Detailed Description	124
9.9	jeod_class.hh File Reference	125
9.9.1	Detailed Description	125
9.10	jeod_integrator_interface.hh File Reference	125
9.10.1	Detailed Description	126
9.11	jeod_trick_integrator.hh File Reference	126
9.11.1	Detailed Description	126
9.12	memory_attributes.hh File Reference	126
9.12.1	Detailed Description	127
9.12.2	Macro Definition Documentation	127
9.12.2.1	JEOD_ATTRIBUTES	127
9.12.2.2	JEOD_DECLARE_ATTRIBUTES	127
9.13	memory_interface.cc File Reference	127
9.13.1	Detailed Description	128
9.14	memory_interface.hh File Reference	128
9.14.1	Detailed Description	128
9.15	sim_interface_messages.cc File Reference	128
9.15.1	Detailed Description	129
9.16	sim_interface_messages.hh File Reference	129
9.16.1	Detailed Description	129
9.17	simulation_interface.cc File Reference	129



9.17.1 Detailed Description . . . . .	130
9.18 simulation_interface.hh File Reference . . . . .	130
9.18.1 Detailed Description . . . . .	130
9.19 trick10_memory_interface.cc File Reference . . . . .	130
9.19.1 Detailed Description . . . . .	131
9.20 trick10_memory_interface.hh File Reference . . . . .	131
9.20.1 Detailed Description . . . . .	131
9.21 trick_dynbody_integ_loop.cc File Reference . . . . .	132
9.21.1 Detailed Description . . . . .	132
9.22 trick_dynbody_integ_loop.hh File Reference . . . . .	132
9.22.1 Detailed Description . . . . .	133
9.23 trick_memory_interface.cc File Reference . . . . .	133
9.23.1 Detailed Description . . . . .	133
9.24 trick_memory_interface.hh File Reference . . . . .	133
9.24.1 Detailed Description . . . . .	134
9.25 trick_memory_interface_alloc.cc File Reference . . . . .	134
9.25.1 Detailed Description . . . . .	135
9.26 trick_memory_interface_attrib.cc File Reference . . . . .	135
9.26.1 Detailed Description . . . . .	135
9.27 trick_memory_interface_chkpnt.cc File Reference . . . . .	135
9.27.1 Detailed Description . . . . .	136
9.28 trick_memory_interface_xlate.cc File Reference . . . . .	136
9.28.1 Detailed Description . . . . .	136
9.29 trick_message_handler.cc File Reference . . . . .	137
9.29.1 Detailed Description . . . . .	137
9.30 trick_message_handler.hh File Reference . . . . .	137
9.30.1 Detailed Description . . . . .	138
9.31 trick_sim_interface.cc File Reference . . . . .	138
9.31.1 Detailed Description . . . . .	138
9.32 trick_sim_interface.hh File Reference . . . . .	138
9.32.1 Detailed Description . . . . .	139



# 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:

<a href="#">er7_utils</a>	Namespace <a href="#">er7_utils</a> contains the state integration models used by JEOD . . . . .	<a href="#">19</a>
<a href="#">jeod</a>	Namespace jeod . . . . .	<a href="#">19</a>
<a href="#">Trick</a>	Namespace <a href="#">Trick</a> furnishes several standard functions for use in the <a href="#">Trick</a> environment . . . . .	<a href="#">20</a>



## 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 . . . . .	41
Integrator	
jeod::TrickJeodIntegrator . . . . .	115
IntegratorInterface	
jeod::JeodIntegratorInterface . . . . .	52
jeod::JeodTrickIntegrator . . . . .	75
std::ios_base	
std::basic_ios	
std::basic_istream	
std::istream	
jeod::SectionedInputStream . . . . .	96
std::basic_ostream	
std::ostream	
jeod::SectionedOutputStream . . . . .	106
JeodIntegrationGroupOwner	
jeod::JeodDynbodyIntegrationLoop . . . . .	41
jeod::JeodMemoryInterface . . . . .	53
jeod::JeodTrickMemoryInterface . . . . .	79
jeod::JeodTrick10MemoryInterface . . . . .	68
jeod::JeodSimulationInterface . . . . .	59
jeod::BasicJeodTrickSimInterface . . . . .	22
jeod::JeodTrickSimInterface . . . . .	90
jeod::JeodSimulationInterfaceInit . . . . .	66
jeod::CheckPointInputManager::SectionInfo . . . . .	111
jeod::SimInterfaceMessages . . . . .	113
streambuf	
jeod::SectionedInputBuffer . . . . .	92
jeod::SectionedOutputBuffer . . . . .	103
SuppressedCodeMessageHandler	
jeod::TrickMessageHandler . . . . .	116
jeod::TrickMessageHandlerMixin . . . . .	118
jeod::JeodTrickSimInterface . . . . .	90





## Chapter 4

# Data Structure Index

### 4.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">jeod::JeodTrickMemoryInterface::AllocationMapEntry</a>	21
Describes a chunk of JEOD-allocated memory . . . . .	
<a href="#">jeod::BasicJeodTrickSimInterface</a>	22
The <a href="#">BasicJeodTrickSimInterface</a> implements the required capabilities of the generic <a href="#">Jeod-SimulationInterface</a> in a <a href="#">Trick</a> simulation environment . . . . .	
<a href="#">jeod::CheckPointInputManager</a>	30
A <a href="#">CheckPointInputManager</a> provides tools for reading a checkpoint file . . . . .	
<a href="#">jeod::CheckPointOutputManager</a>	35
A <a href="#">CheckPointOutputManager</a> provides the basic tools for writing a checkpoint file . . . . .	
<a href="#">jeod::JeodTrickMemoryInterface::ContainerListEntry</a>	40
Describes a Checkpointable object . . . . .	
<a href="#">jeod::JeodDynbodyIntegrationLoop</a>	41
A <a href="#">Trick::IntegLoopScheduler</a> that provides the ability to integrate a collection of <a href="#">Trick::SimObject</a> instances over time, with the sim objects capable of being moved from one integration loop to another during run time . . . . .	
<a href="#">jeod::JeodIntegratorInterface</a>	52
A <a href="#">JeodIntegratorInterface</a> extends the ER7 <a href="#">IntegratorInterface</a> with the concept of a pointer to the simulation engine's integration object . . . . .	
<a href="#">jeod::JeodMemoryInterface</a>	53
Abstract interface between the JEOD memory manager and the simulation engine . . . . .	
<a href="#">jeod::JeodSimulationInterface</a>	59
This abstract class defines the basis for the interface between JEOD and a simulation engine . . . . .	
<a href="#">jeod::JeodSimulationInterfaceInit</a>	66
Define configuration data needed to configure the dynamically-created message handler and memory manager . . . . .	
<a href="#">jeod::JeodTrick10MemoryInterface</a>	68
A <a href="#">TrickMemoryInterface</a> implements the two required methods needed to register and deregister memory with the simulation engine, <a href="#">Trick</a> in this case . . . . .	
<a href="#">jeod::JeodTrickIntegrator</a>	75
A <a href="#">JeodTrickIntegrator</a> specializes the <a href="#">JeodIntegratorInterface</a> for use with <a href="#">Trick</a> as the simulation engine . . . . .	
<a href="#">jeod::JeodTrickMemoryInterface</a>	79
A <a href="#">TrickMemoryInterface</a> implements the two required methods needed to register and deregister memory with the simulation engine, <a href="#">Trick</a> in this case . . . . .	
<a href="#">jeod::JeodTrickSimInterface</a>	90
A <a href="#">JEODTrickSimInterface</a> implements the required capabilities of the generic <a href="#">JeodSimulation-Interface</a> in a <a href="#">Trick</a> simulation environment . . . . .	

<a href="#">jeod::SectionedInputBuffer</a>	
A <a href="#">SectionedInputBuffer</a> is a <code>std::streambuf</code> that reads from a section in a checkpoint file . . . . .	92
<a href="#">jeod::SectionedInputStream</a>	
A <a href="#">SectionedInputStream</a> is a <code>std::istream</code> that reads from a section in a checkpoint file . . . . .	96
<a href="#">jeod::SectionedOutputBuffer</a>	
A <a href="#">SectionedOutputBuffer</a> is a <code>std::streambuf</code> that writes a section of a checkpoint file . . . . .	103
<a href="#">jeod::SectionedOutputStream</a>	
A <a href="#">SectionedOutputStream</a> is a <code>std::ostream</code> that writes a section of a checkpoint file . . . . .	106
<a href="#">jeod::CheckpointInputManager::SectionInfo</a>	
A <a href="#">SectionInfo</a> contains the start and end positions of a checkpoint file section . . . . .	111
<a href="#">jeod::SimInterfaceMessages</a>	
Specifies the message IDs used in the <code>sim_interface</code> model . . . . .	113
<a href="#">jeod::TrickJeodIntegrator</a>	
A <a href="#">TrickJeodIntegrator</a> specializes the <code>Trick::Integrator</code> to provide the <a href="#">Trick</a> side of the integration interface between <a href="#">Trick</a> and JEOD . . . . .	115
<a href="#">jeod::TrickMessageHandler</a>	
The <code>MessageHandler</code> class for designed for use in <a href="#">Trick</a> -based simulations . . . . .	116
<a href="#">jeod::TrickMessageHandlerMixin</a>	
The <a href="#">TrickMessageHandlerMixin</a> implements the required capabilities of the generic <a href="#">Jeod-SimulationInterface</a> in a <a href="#">Trick</a> simulation environment . . . . .	118

## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

<a href="#">checkpoint_input_manager.cc</a>	Define CheckPointInputManager member functions and of related classes . . . . .	121
<a href="#">checkpoint_input_manager.hh</a>	Define class CheckPointInputManager and related classes . . . . .	121
<a href="#">checkpoint_output_manager.cc</a>	Define CheckPointOutputManager member functions and of related classes . . . . .	122
<a href="#">checkpoint_output_manager.hh</a>	Define class CheckPointOutputManager and related classes . . . . .	122
<a href="#">class_declarations.hh</a>	Forward declarations of classes defined in the utils/sim_interface model . . . . .	123
<a href="#">config.hh</a>	Configure JEOD for use by some simulation engine . . . . .	123
<a href="#">config_test_harness.hh</a>	Configure JEOD for use in standalone test mode . . . . .	124
<a href="#">config_trick10.hh</a>	Configure JEOD for use in a Trick10 environment . . . . .	124
<a href="#">jeod_class.hh</a>	Define the JEOD class declaration macros JEOD_MAKE_SIM_INTERFACES and and JEOD_DECLARE_SIM_INTERFACES . . . . .	125
<a href="#">jeod_integrator_interface.hh</a>	Define the interface for accessing / updating elements of a simulation engine's integrator object . . . . .	125
<a href="#">jeod_trick_integrator.hh</a>	Define the interface for accessing / updating elements of a <a href="#">Trick</a> simulation integrator object . . . . .	126
<a href="#">memory_attributes.hh</a>	Define JEOD memory interface macros . . . . .	126
<a href="#">memory_interface.cc</a>	Implement the MemoryInterface class . . . . .	127
<a href="#">memory_interface.hh</a>	Define the MemoryInterface class, which abstractly defines the interface between the memory manager and the simulation engine . . . . .	128
<a href="#">sim_interface_messages.cc</a>	Implement the class SimInterfaceMessages . . . . .	128
<a href="#">sim_interface_messages.hh</a>	Define the class SimInterfaceMessages, the class that specifies the message IDs used in the sim interface model . . . . .	129
<a href="#">simulation_interface.cc</a>	Implement SimulationInterface methods . . . . .	129

<a href="#">simulation_interface.hh</a>	Define the abstract class JeodSimulationInterface . . . . .	130
<a href="#">trick10_memory_interface.cc</a>	Define JeodTrickMemoryInterface methods . . . . .	130
<a href="#">trick10_memory_interface.hh</a>	Define the interface for registering / deregistering memory with <a href="#">Trick</a> . . . . .	131
<a href="#">trick_dynbody_integ_loop.cc</a>	Define JeodDynbodyIntegrationLoop methods . . . . .	132
<a href="#">trick_dynbody_integ_loop.hh</a>	Define the class IntegrationGroupIntegLoopScheduler, which replaces the base <a href="#">Trick</a> integration loop for multi-rate JEOD-based simulations . . . . .	132
<a href="#">trick_memory_interface.cc</a>	Define JeodTrickMemoryInterface methods . . . . .	133
<a href="#">trick_memory_interface.hh</a>	Define the interface for registering / deregistering memory with <a href="#">Trick</a> . . . . .	133
<a href="#">trick_memory_interface_alloc.cc</a>	Define JeodTrickMemoryInterface methods related to allocation/deallocation . . . . .	134
<a href="#">trick_memory_interface_attrib.cc</a>	Define JeodTrickMemoryInterface methods related to attributes . . . . .	135
<a href="#">trick_memory_interface_chkpnt.cc</a>	Define JeodTrick10MemoryInterface methods related to checkpoint/restart . . . . .	135
<a href="#">trick_memory_interface_xlate.cc</a>	Define JeodTrickMemoryInterface methods related to name translation . . . . .	136
<a href="#">trick_message_handler.cc</a>	Define member functions for the class TrickMessageHandler . . . . .	137
<a href="#">trick_message_handler.hh</a>	Define the class TrickMessageHandler, the message handler designed for use in Trick-based simulations . . . . .	137
<a href="#">trick_sim_interface.cc</a>	Implement TrickSimInterface methods . . . . .	138
<a href="#">trick_sim_interface.hh</a>	Define the class JeodTrickSimInterface . . . . .	138

## 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 38 of file `sim_interface_messages.cc`.

#### 6.3.2.2 `#define ER7_UTILS_ALWAYS_INLINE`

Definition at line 113 of file `config.hh`.

#### 6.3.2.3 `#define ER7_UTILS_RESTRICT`

Definition at line 108 of file `config.hh`.

#### 6.3.2.4 `#define ER7_UTILS_UNUSED`

Definition at line 103 of file `config.hh`.

#### 6.3.2.5 `#define JEOD_ATTRIBUTES_POINTER_TYPE void *`

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

#### 6.3.2.6 `#define JEOD_ATTRIBUTES_POINTER_TYPE JEOD_ATTRIBUTES_TYPE *`

Definition at line 93 of file `config_trick10.hh`.

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

Definition at line 90 of file `config_trick10.hh`.

### 6.3.2.8 #define JEOD\_ATTRIBUTES\_TYPE int

Definition at line 79 of file config\_test\_harness.hh.

### 6.3.2.9 #define JEOD\_ATTRIBUTES\_TYPE struct ATTRIBUTES\_tag

Definition at line 92 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 77 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 138 of file jeod\_class.hh.

### 6.3.2.12 #define JEOD\_INTPTR\_T long int

Definition at line 68 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 102 of file jeod\_class.hh.

### 6.3.2.14 #define JEOD\_PTRDIFF\_T long int

Definition at line 67 of file config\_trick10.hh.

### 6.3.2.15 #define JEOD\_SIM\_INTEGRATOR\_ENUM Integrator\_type

Definition at line 108 of file config\_trick10.hh.

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

Definition at line 105 of file config\_trick10.hh.

6.3.2.17 `#define JEOD_SIM_INTEGRATOR_POINTER_TYPE void *`

Definition at line 90 of file config\_test\_harness.hh.

6.3.2.18 `#define JEOD_SIM_INTEGRATOR_POINTER_TYPE Trick::Integrator *`

Definition at line 107 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 103 of file config\_trick10.hh.

6.3.2.20 `#define JEOD_SIZE_T size_t`

Definition at line 66 of file config\_trick10.hh.

6.3.2.21 `#define JEOD_UINTPTR_T unsigned long int`

Definition at line 69 of file config\_trick10.hh.

6.3.2.22 `#define JEOD_UNUSED`

Definition at line 98 of file config.hh.

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

Definition at line 39 of file sim\_interface\_messages.cc.

6.3.2.24 `#define MAX_MSG_SIZE 4096`

Definition at line 45 of file trick\_message\_handler.cc.

Referenced by jeod::TrickMessageHandler::process\_message().

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

Definition at line 37 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 295 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 318 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 310 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 305 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 300 of file `trick_memory_interface.hh`.

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

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

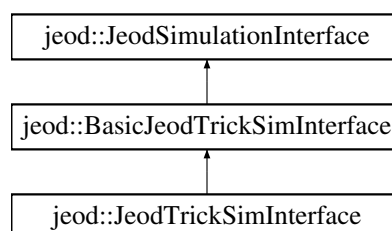
- [trick\\_memory\\_interface.hh](#)

## 8.2 `jeod::BasicJeodTrickSimInterface` Class Reference

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

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

Inheritance diagram for `jeod::BasicJeodTrickSimInterface`:





## Public Member Functions

- [BasicJeodTrickSimInterface](#) (MessageHandler &message\_handler)  
Construct a [BasicJeodTrickSimInterface](#) object.
- [~BasicJeodTrickSimInterface](#) () override  
Destroy a [BasicJeodTrickSimInterface](#) object.
- void [set\\_checkpoint\\_file\\_name](#) (const std::string &name)  
Set the checkpoint file name.
- std::string [get\\_checkpoint\\_file\\_name](#) () const  
Get the checkpoint file name.
- void [set\\_mode](#) (JeodSimulationInterface::Mode new\_mode) override  
Set the mode.
- void [checkpoint\\_allocations](#) (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

- [JeodIntegratorInterface](#) \* [create\\_integrator\\_internal](#) (void) override  
Create an integration interface object.
- double [get\\_job\\_cycle\\_internal](#) (void) override  
Get the current job's cycle time.
- [JeodMemoryInterface](#) & [get\\_memory\\_interface\\_internal](#) (void) override  
Get the memory interface.
- [SectionedInputStream](#) [get\\_checkpoint\\_reader\\_internal](#) (const std::string &section\_id) override  
Get a reader to a section of the currently open checkpoint file.
- [SectionedOutputStream](#) [get\\_checkpoint\\_writer\\_internal](#) (const std::string &section\_id) override  
Get a writer to a section of the currently open checkpoint file.

## Protected Attributes

- MessageHandler & [generic\\_message\\_handler](#)  
The global MessageHandler.
- [JeodTrick10MemoryInterface](#) [trick\\_memory\\_interface](#)  
The interface between JEOD and *Trick*'s memory management schemes.
- JeodMemoryManager [memory\\_manager](#)  
The global JEOD memory manager.

- `std::string` [checkpoint\\_file\\_name](#)  
*The name of the segmented checkpoint file used for the next checkpoint / restart action.*
- `std::string` [section\\_start](#)  
*String indicating the start of a checkpoint file section.*
- `std::string` [section\\_end](#)  
*String indicating the end of a checkpoint file section.*
- [CheckPointInputManager](#) \* [checkpoint\\_reader](#)  
*The object that manages reading from a checkpoint file.*
- [CheckPointOutputManager](#) \* [checkpoint\\_writer](#)  
*The object that manages writing to a checkpoint file.*

## 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 89 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 62 of file `trick_sim_interface.cc`.

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

**8.2.2.2** `jeod::BasicJeodTrickSimInterface::~~BasicJeodTrickSimInterface ( void )` `[override]`

Destroy a [BasicJeodTrickSimInterface](#) object.

Definition at line 98 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 312 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 338 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 229 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 300 of file `trick_sim_interface.cc`.

References `checkpoint_reader`.

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

Create an integration interface object.

**Returns**

Integrator interface that encapsulates an sim engine integrator.

Implements [jeod::JeodSimulationInterface](#).

Definition at line 135 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 110 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 ) [override], [protected], [virtual]`

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

**Returns**

Checkpoint reader

**Parameters**

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

Implements [jeod::JeodSimulationInterface](#).

Definition at line 281 of file `trick_sim_interface.cc`.

References `checkpoint_reader`, `jeod::CheckPointInputManager::create_section_reader()`, and `jeod::SimInterfaceMessages::phasing_error`.

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

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

**Returns**

Checkpoint writer

**Parameters**

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

Implements [jeod::JeodSimulationInterface](#).

Definition at line 210 of file `trick_sim_interface.cc`.

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

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

Get the current job's cycle time.

**Returns**

Current job's cycle time  
Units: s

Implements [jeod::JeodSimulationInterface](#).

Definition at line 147 of file `trick_sim_interface.cc`.

### 8.2.3.10 JeodMemoryInterface & jeod::BasicJeodTrickSimInterface::get\_memory\_interface\_internal ( void ) [override], [protected], [virtual]

Get the memory interface.

**Returns**

Memory interface

Implements [jeod::JeodSimulationInterface](#).

Definition at line 159 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 170 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 241 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 325 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 351 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 ( const std::string & name ) [inline]

Set the checkpoint file name.

Definition at line 104 of file trick\_sim\_interface.hh.

References [checkpoint\\_file\\_name](#).

#### 8.2.3.17 void jeod::BasicJeodTrickSimInterface::set\_mode ( JeodSimulationInterface::Mode new\_mode ) [override], [virtual]

Set the mode.

#### Assumptions and Limitations

- See [SimulationInterface::set\\_mode](#).

#### Parameters

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

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

Definition at line 116 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 90 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 200 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 215 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 220 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 181 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 191 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 210 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 205 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 186 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 (const [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 419 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 314 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 438 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 <a href="#">Trick</a> section reader?
<i>in</i>	<i>tag</i>	Tag identifying the section to be read.

Definition at line 424 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 467 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 ( const 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 504 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 454 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 344 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 446 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 485 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 524 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.

Definition at line 529 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 544 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.

Definition at line 539 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.

Definition at line 534 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 514 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 519 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](#) (const [SectionedOutputStream](#) \*writer)  
*Deregister the supplied section writer as the currently-active writer.*

### Private Member Functions

- [SectionedOutputStream create\\_section\\_writer](#) (bool trick, const std::string &tag)  
*Create a C++ output stream that writes to a checkpoint file section.*
- [SectionedOutputStream create\\_trick\\_section\\_writer](#) ()  
*Create a C++ output stream that writes a [Trick](#) checkpoint file section.*
- [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 279 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

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

Definition at line 318 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 293 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 <a href="#">Trick</a> section writer?
in	<i>tag</i>	Tag identifying the section to be written.

Definition at line 352 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 385 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 ( const [SectionedOutputStream](#) \* *writer* )

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

**Returns**

True => success.

**Parameters**

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

Definition at line 422 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 309 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 301 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 403 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 280 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 345 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.

Definition at line 350 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 365 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.

Definition at line 360 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.

Definition at line 355 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 340 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 248 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 278 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 268 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 263 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 253 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 258 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:

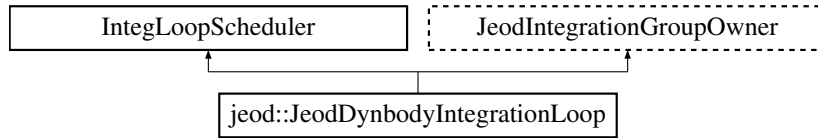
- [trick\\_memory\\_interface.hh](#)

## 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.
- [~JeodDynbodyIntegrationLoop](#) () override  
*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.*
- void [update\\_integration\\_group](#) (JeodIntegrationGroup &group) override  
*Update the provided integration group, which must be the integration group contained within this integration loop object.*
- int [add\\_sim\\_object](#) (Trick::SimObject &sim\_obj) override  
*Add a sim object to the set of objects to be integrated by this integration loop object.*
- virtual void [add\\_integrable\\_object](#) (er7\_utils::IntegrableObject &integrable\_object)  
*Add the specified integrable object, which should not be a DynBody, to the integration group's set of integrable objects.*
- int [remove\\_sim\\_object](#) (Trick::SimObject &sim\_obj) override  
*Remove a sim object from the set of objects to be integrated by this integration loop object.*
- virtual void [remove\\_integrable\\_object](#) (er7\_utils::IntegrableObject &integrable\_object)  
*Remove the specified integrable object from the integration group's set of integrable objects.*
- virtual void [gravitation](#) (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.*
- int [integrate\\_dt](#) (double beg\_sim\_time, double del\_sim\_time) override  
*Integrate sim objects over the specified time span.*

## Protected Attributes

- Trick::SimObject \* [loop\\_sim\\_object](#)  
The simulation object that contains this integration loop object.
- DynManager \* [dyn\\_manager](#)  
The JEOD dynamics manager.
- TimeManager \* [time\\_manager](#)  
The JEOD time manager.
- GravityManager \* [gravity\\_manager](#)  
The gravity model manager.
- [JeodTrickIntegrator integ\\_interface](#)  
Dummy integration interface; needed by the integ\_group.
- er7\_utils::IntegratorConstructor \*\* [integ\\_constructor](#)  
Handle to the integration constructor used to create integrators.
- const DynamicsIntegrationGroup \* [integ\\_group\\_factory](#)  
The externally-supplied integration group used as a template for creating this integration loop's integration group.
- DynamicsIntegrationGroup \* [integ\\_group](#)  
The integration group that performs the integration.
- bool [deriv\\_ephem\\_update](#)  
If set, ephemerides will be updated at the derivative rate.

## 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 133 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 55 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 <code>SimObject</code> that contains this <a href="#">JeodDynbodyIntegrationLoop</a> 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 71 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 )` `[override]`

[JeodDynbodyIntegrationLoop](#) destructor.

Definition at line 106 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 184 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 ) [override]`

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

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

## Parameters

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

## Returns

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

Definition at line 285 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 329 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 348 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 284 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 146 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 274 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 117 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 ) [override], [protected]`

Integrate sim objects over the specified time span.

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

## Returns

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

## Parameters

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

Definition at line 222 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 205 of file `trick_dynbody_integ_loop.cc`.

References `integ_group`.

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

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

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

## Parameters

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

## Returns

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

Definition at line 307 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 369 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 <code>deriv_ephem_update</code> .
------------	---

Definition at line 294 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 214 of file `trick_dynbody_integ_loop.cc`.

References `time_manager`.

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

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

#### Note

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

#### Parameters

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

Definition at line 388 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 138 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 404 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 364 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 374 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 384 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 395 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 390 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 379 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 359 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 369 of file `trick_dynbody_integ_loop.hh`.

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

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

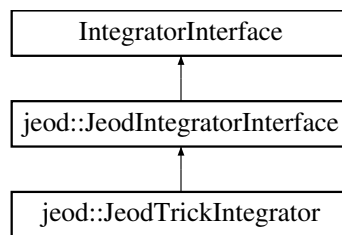
- [trick\\_dynbody\\_integ\\_loop.hh](#)
- [trick\\_dynbody\\_integ\\_loop.cc](#)

## 8.7 jeod::JeodIntegratorInterface Class Reference

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

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

Inheritance diagram for jeod::JeodIntegratorInterface:



### Public Member Functions

- [~JeodIntegratorInterface](#) () override  
*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 86 of file [jeod\\_integrator\\_interface.hh](#).

### 8.7.2 Constructor & Destructor Documentation

#### 8.7.2.1 jeod::JeodIntegratorInterface::~JeodIntegratorInterface ( ) [inline], [override]

Destructor.

Definition at line 98 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 87 of file `jeod_integrator_interface.hh`.

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

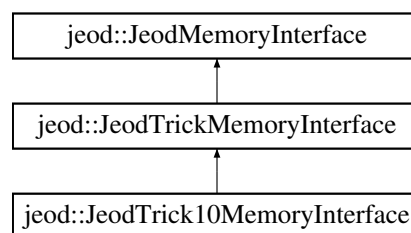
- [jeod\\_integrator\\_interface.hh](#)

## 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 90 of file `memory_interface.hh`.

### 8.8.2 Constructor & Destructor Documentation

#### 8.8.2.1 `jeod::JeodMemoryInterface::JeodMemoryInterface ( void )`

Default constructor.

Definition at line 41 of file `memory_interface.cc`.



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

Destructor.

Definition at line 51 of file `memory_interface.cc`.

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

Copy constructor.

Parameters

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

Definition at line 62 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

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

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

<i>in</i>	<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 <a href="#">get_name_at_address()</a>
-----------	-------------	---

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**

<i>in</i>	<i>addr</i>	Address of memory to identified by name
<i>in</i>	<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**

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

Definition at line 75 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**

<i>in</i>	<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 92 of file memory\_interface.hh.

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

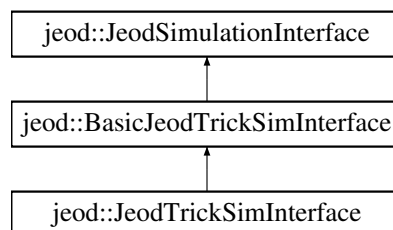
- [memory\\_interface.hh](#)
- [memory\\_interface.cc](#)

## 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 &section\_id)  
*Get a reader of a section of the currently open checkpoint file.*
- static [SectionedOutputStream](#) [get\\_checkpoint\\_writer](#) (const std::string &section\_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 &section\_id)=0  
*Get a checkpoint section reader.*
- virtual [SectionedOutputStream](#) [get\\_checkpoint\\_writer\\_internal](#) (const std::string &section\_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](#) = nullptr  
*The singleton instance of a SimulationInterface object that must be created by a conforming JEOD simulation before any call can be made to one of the three static methods declared above.*

## 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 138 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 148 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 73 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 93 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 107 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 129 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 223 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 248 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 269 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 152 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 175 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 250 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 197 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 296 of file simulation\_interface.cc.

References Checkpoint, Construction, Dead, jeod::SimInterfaceMessages::implementation\_error, Initialization, mode, NumModes, Operational, jeod::SimInterfaceMessages::phasing\_error, PostCheckpoint, PreCheckpoint, Restart, Restore, saved\_mode, and Shutdown.

Referenced by jeod::BasicJeodTrickSimInterface::set\_mode().

### 8.9.5 Friends And Related Function Documentation

8.9.5.1 `void init_attrjeod__JeodSimulationInterface ( ) [friend]`

8.9.5.2 `friend class InputProcessor [friend]`

Definition at line 139 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 313 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 319 of file `simulation_interface.hh`.

Referenced by `set_mode()`.

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

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

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

Definition at line 270 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 87 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 51 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 127 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 113 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 120 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 106 of file simulation\_interface.hh.

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

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

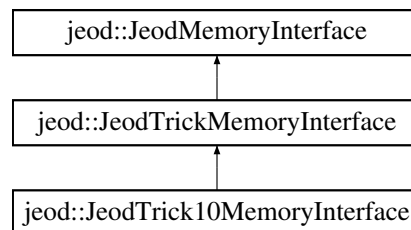
- [simulation\\_interface.hh](#)
- [simulation\\_interface.cc](#)

## 8.11 jeod::JeodTrick10MemoryInterface Class Reference

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

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

Inheritance diagram for jeod::JeodTrick10MemoryInterface:



### Public Member Functions

- [JeodTrick10MemoryInterface](#) ()  
*Construct a [JeodTrick10MemoryInterface](#) object.*
- [~JeodTrick10MemoryInterface](#) () override  
*Destruct a [JeodTrick10MemoryInterface](#) object.*
- void [register\\_container](#) (const void \*owner, const JeodMemoryTypeDescriptor &owner\_type, const char \*elem\_name, JeodCheckpointable &container) override  
*Register the checkpointable object with [Trick](#).*
- void [deregister\\_container](#) (const void \*owner, const JeodMemoryTypeDescriptor &owner\_type, const char \*elem\_name, JeodCheckpointable &container) override  
*Revoke the registrations performed by register\_container.*
- const std::string [get\\_name\\_at\\_address](#) (const void \*addr, const JeodMemoryTypeDescriptor \*tdesc) const override  
*Get the simulation name, if any, associated with the address.*

- void \* [get\\_address\\_at\\_name](#) (const std::string &name) const override  
*Get the address, if any, that corresponds to the given name.*
- bool [is\\_checkpoint\\_restart\\_supported](#) (void) const override  
*The Trick10 memory interface supports checkpoint/restart.*
- const std::string [get\\_trick\\_checkpoint\\_file](#) (bool checkpoint) override  
*Get the name of the current [Trick](#) checkpoint file.*
- void [checkpoint\\_containers](#) (void) override  
*Dump the checkpointable objects to the checkpoint file.*
- void [restore\\_containers](#) (void) override  
*Restore the checkpointable objects from the checkpoint file.*
- void [checkpoint\\_allocations](#) (void) override  
*Dump the allocation information to the checkpoint file.*
- void [restore\\_allocations](#) (JeodMemoryManager &memory\_manager) override  
*Restore the allocated data per the checkpoint file.*

### Protected Member Functions

- std::string [get\\_container\\_id](#) (const [ContainerListEntry](#) &entry) const  
*Construct the identifier for a checkpointable object.*
- std::string [translate\\_addr\\_to\\_name](#) (const void \*addr, const ATTRIBUTES \*attr) const  
*Translate the given address to an address specification string, with the address interpreted in the context of the supplied attributes.*
- void \* [translate\\_name\\_to\\_addr](#) (const std::string &spec) const  
*Translate the given address specification string to an address.*

### Protected Attributes

- Trick::ClassicCheckPointAgent \* [trick\\_checkpoint\\_agent](#)  
*[Trick](#) checkpoint agent.*

### 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 109 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 62 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 ) [override]`

Destruct a [JeodTrick10MemoryInterface](#) object.

Definition at line 84 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 ) [override],[virtual]`

Dump the allocation information to the checkpoint file.

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

Definition at line 431 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 ) [override],[virtual]`

Dump the checkpointable objects to the checkpoint file.

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

Definition at line 243 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 ) [override],[virtual]`

Revoke the registrations performed by `register_container`.

This function is typically called at destruction time via `JEOD_DEREGISTER_CHECKPOINTABLE`.



## Assumptions and Limitations

- The following unenforced assumptions are made:
  - A corresponding register\_container was previously made.
  - Trick has been pre-initialized.

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

## Parameters

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

Implements [jeod::JeodMemoryInterface](#).

Definition at line 145 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`  
[override], [virtual]

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

## Returns

Name of the address, if any

## Parameters

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

Implements [jeod::JeodMemoryInterface](#).

Definition at line 124 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

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

Definition at line 200 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` `[override], [virtual]`

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

#### Returns

Name of the address, if any

#### Parameters

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

Implements [jeod::JeodMemoryInterface](#).

Definition at line 88 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 )` `[override], [virtual]`

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

#### Returns

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

Units:

#### Parameters

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

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

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

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

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

The Trick10 memory interface supports checkpoint/restart.

Implements [jeod::JeodMemoryInterface](#).

Definition at line 149 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* ) `[override]`, `[virtual]`

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

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

#### Assumptions and Limitations

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

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

#### Parameters

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

Implements [jeod::JeodMemoryInterface](#).

Definition at line 82 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* ) `[override]`, `[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 501 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 ) `[override]`, `[virtual]`

Restore the checkpointable objects from the checkpoint file.

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

Definition at line 312 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 156 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 196 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 111 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 183 of file trick10\_memory\_interface.hh.

Referenced by JeodTrick10MemoryInterface(), and translate\_addr\_to\_name().

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

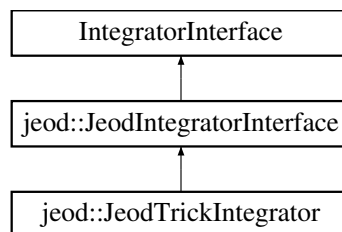
- [trick10\\_memory\\_interface.hh](#)
- [trick10\\_memory\\_interface.cc](#)
- [trick\\_memory\\_interface\\_chkpnt.cc](#)
- [trick\\_memory\\_interface\\_xlate.cc](#)

## 8.12 jeod::JeodTrickIntegrator Class Reference

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

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

Inheritance diagram for jeod::JeodTrickIntegrator:



### Public Member Functions

- [JeodTrickIntegrator](#) ()  
*Default constructor.*
- [~JeodTrickIntegrator](#) () override  
*Destructor.*
- er7\_utils::Integration::Technique [interpret\\_integration\\_type](#) (int integ\_technique) const override  
*Interpret the integration technique.*
- virtual Trick::Integrator\* [get\\_integrator](#) ()  
*Get the simulation engine's integrator.*
- double [get\\_dt](#) () const override  
*Get the integration cycle time step.*
- bool [get\\_first\\_step\\_derivs\\_flag](#) () const override  
*Get the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle.*
- void [set\\_first\\_step\\_derivs\\_flag](#) (bool value) override  
*Set the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle.*
- void [reset\\_first\\_step\\_derivs\\_flag](#) () override  
*Reset the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle.*
- void [restore\\_first\\_step\\_derivs\\_flag](#) () override  
*Restore the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle to its value prior to the most recent call to reset\_first\_step\_derivs\_flag.*
- void [set\\_step\\_number](#) (unsigned int stepno) override

*Set the step number within an integration cycle.*

- void [set\\_time](#) (double sim\_time) override

*Update the time model given the simulation time.*

## 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 119 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 129 of file [jeod\\_trick\\_integrator.hh](#).

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

Destructor.

Definition at line 140 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 `double jeod::JeodTrickIntegrator::get_dt ( ) const [inline], [override]`

Get the integration cycle time step.

##### Returns

Simulation time delta t, in seconds

Definition at line 166 of file `jeod_trick_integrator.hh`.

References `trick_integrator`.

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

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

##### Returns

Value of the first step derivatives flag

Definition at line 176 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 157 of file `jeod_trick_integrator.hh`.

References `trick_integrator`.

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

Interpret the integration technique.

Implements [jeod::JeodIntegratorInterface](#).

Definition at line 145 of file `jeod_trick_integrator.hh`.

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

Not implemented.

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

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

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

Definition at line 197 of file jeod\_trick\_integrator.hh.

References default\_first\_step\_deriv, and trick\_integrator.

#### 8.12.3.7 void jeod::JeodTrickIntegrator::restore\_first\_step\_derivs\_flag ( ) [inline],[override]

Restore the flag that tells the simulation engine to compute derivatives on the initial step of each integration cycle to it's value prior to the most recent call to reset\_first\_step\_derivs\_flag.

Definition at line 208 of file jeod\_trick\_integrator.hh.

References default\_first\_step\_deriv, and trick\_integrator.

#### 8.12.3.8 void jeod::JeodTrickIntegrator::set\_first\_step\_derivs\_flag ( bool *value* ) [inline],[override]

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

Parameters

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

Definition at line 186 of file jeod\_trick\_integrator.hh.

References trick\_integrator.

#### 8.12.3.9 void jeod::JeodTrickIntegrator::set\_step\_number ( unsigned int *stepno* ) [inline],[override]

Set the step number within an integration cycle.

Parameters

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

Definition at line 217 of file jeod\_trick\_integrator.hh.

References trick\_integrator.

#### 8.12.3.10 void jeod::JeodTrickIntegrator::set\_time ( double *sim\_time* ) [inline],[override]

Update the time model given the simulation time.

Parameters

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

Definition at line 226 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 120 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 244 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 239 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:

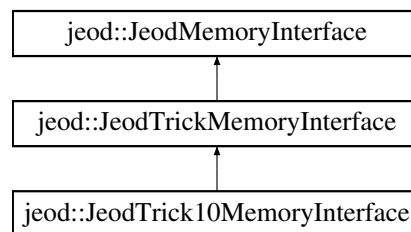
- [jeod\\_trick\\_integrator.hh](#)

## 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.*
- [~JeodTrickMemoryInterface](#) () override  
*[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.*
- struct ATTRIBUTES\_tag \* [find\\_attributes](#) (const std::string &type\_name) const override  
*Find the attributes for a class in the symbol table.*
- struct ATTRIBUTES\_tag \* [find\\_attributes](#) (const std::type\_info &data\_type) const override  
*Find the attributes for a class in the symbol table.*
- struct ATTRIBUTES\_tag [primitive\\_attributes](#) (const std::type\_info &data\_type) const override  
*Create an attributes structure that represents a primitive type.*
- struct ATTRIBUTES\_tag [pointer\\_attributes](#) (const struct ATTRIBUTES\_tag &target\_attr) const override  
*Create an attributes structure that represents a pointer type.*
- struct ATTRIBUTES\_tag [void\\_pointer\\_attributes](#) () const override  
*Create an attributes structure that represents a void\* pointer.*
- struct ATTRIBUTES\_tag [structure\\_attributes](#) (const struct ATTRIBUTES\_tag \*target\_attr, std::size\_t target\_size) const override  
*Create an attributes structure that represents a structured type.*
- bool [register\\_allocation](#) (const void \*addr, const JeodMemoryItem &item, const JeodMemoryTypeDescriptor &tdesc, const char \*file, unsigned int line) override  
*Register newly allocated memory with [Trick](#).*
- void [deregister\\_allocation](#) (const void \*addr, const JeodMemoryItem &item, const JeodMemoryTypeDescriptor &tdesc, const char \*file, unsigned int line) override  
*Delete [Trick](#) information about some pointer – but not the pointer itself.*
- void [register\\_container](#) (const void \*owner, const JeodMemoryTypeDescriptor &owner\_type, const char \*elem\_name, JeodCheckpointable &container) override  
*Register the checkpointable object with [Trick](#).*
- void [deregister\\_container](#) (const void \*owner, const JeodMemoryTypeDescriptor &owner\_type, const char \*elem\_name, JeodCheckpointable &container) override  
*Revoke the registrations performed by [register\\_container](#).*
- const std::string [get\\_name\\_at\\_address](#) (const void \*addr, const JeodMemoryTypeDescriptor \*tdesc) const override  
*Stubbed-out implementation of [get\\_name\\_at\\_address](#) for [Trick](#) implementations that do not fully support JEOD checkpoint/restart requirements.*
- void \* [get\\_address\\_at\\_name](#) (const std::string &name) const override  
*Stubbed-out implementation of [get\\_address\\_at\\_name](#) for [Trick](#) implementations that do not fully support JEOD checkpoint/restart requirements.*
- bool [is\\_checkpoint\\_restart\\_supported](#) () const override  
*The generic [Trick](#) memory interface does not support checkpoint/restart.*
- virtual const std::string [get\\_trick\\_checkpoint\\_file](#) (bool checkpoint)  
*Get the name of the current [Trick](#) checkpoint file.*
- virtual void [checkpoint\\_containers](#) ()  
*Dump the container checkpointable objects to the checkpoint file.*
- virtual void [restore\\_containers](#) ()  
*Restore the container checkpointables objects from the checkpoint file.*
- virtual void [checkpoint\\_allocations](#) ()  
*Dump the allocation information to the checkpoint file.*
- virtual void [restore\\_allocations](#) (JeodMemoryManager &memory\_manager)  
*Restore the allocated data per the checkpoint file.*

## Protected Types

- 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 98 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 333 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 338 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 55 of file `trick_memory_interface.cc`.

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

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

[JeodTrickMemoryInterface](#) destructor.

Definition at line 76 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 227 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 211 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 103 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 ) [override], [virtual]`

Delete [Trick](#) information about some pointer – but not the pointer itself.

##### Assumptions and Limitations

- Some other agent must freeing the memory at the input address itself. This function merely deletes [Trick](#)'s knowledge of that pointer.

##### Parameters

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

Implements [jeod::JeodMemoryInterface](#).

Definition at line 138 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 ) [override], [virtual]`

Revoke the registrations performed by `register_container`.

This function is typically called at destruction time via `JEOD_DEREGISTER_CHECKPOINTABLE`. This default implementation does nothing.

##### Parameters

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

Implements [jeod::JeodMemoryInterface](#).

Definition at line 148 of file `trick_memory_interface.cc`.

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

Find the attributes for a class in the symbol table.

#### Returns

Found attributes

#### Parameters

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

Implements [jeod::JeodMemoryInterface](#).

Definition at line 55 of file `trick_memory_interface_attr.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`  
`[override],[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 86 of file `trick_memory_interface_attr.cc`.

References `find_attributes()`.

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

Stubbed-out implementation of `get_address_at_name` for [Trick](#) implementations that do not fully support JEOD checkpoint/restart requirements.

#### Returns

Address of named item in memory

#### Parameters

<i>name</i>	Name of item to be found
-------------	--------------------------

Implements [jeod::JeodMemoryInterface](#).

Definition at line 182 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` `[override],[virtual]`

Stubbed-out implementation of `get_name_at_address` for [Trick](#) implementations that do not fully support JEOD checkpoint/restart requirements.

**Returns**

Name of the address, if any.

**Parameters**

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

Implements [jeod::JeodMemoryInterface](#).

Definition at line 165 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 199 of file `trick_memory_interface.hh`.

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

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

Implements [jeod::JeodMemoryInterface](#).

Definition at line 189 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` `[override], [virtual]`

Create an attributes structure that represents a pointer type.

**Returns**

Constructed pointer attributes.

## Parameters

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

Implements [jeod::JeodMemoryInterface](#).

Definition at line 206 of file `trick_memory_interface_attr.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 [override],[virtual]`

Create an attributes structure that represents a primitive type.

## Returns

Constructed attributes.

## Parameters

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

Implements [jeod::JeodMemoryInterface](#).

Definition at line 100 of file `trick_memory_interface_attr.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 ) [override],[virtual]`

Register newly allocated memory with [Trick](#).

## Assumptions and Limitations

- Memory was indeed allocated.
- The input address is not null.
- The number of elements is positive.

## Returns

True if registered

## Parameters

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

Implements [jeod::JeodMemoryInterface](#).

Definition at line 75 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 ) [override],[virtual]`

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



This function is typically called at construction or initialization time via JEOD\_REGISTER\_CHECKPOINTABLE. This default implementation does nothing.

## Parameters

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

Implements [jeod::JeodMemoryInterface](#).

Definition at line 127 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 236 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 219 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 90 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` `[override], [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 279 of file trick\_memory\_interface\_attrib.cc.

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

Create an attributes structure that represents a void\* pointer.

**Returns**

Constructed pointer attributes.

Implements [jeod::JeodMemoryInterface](#).

Definition at line 254 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 100 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 351 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 356 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 346 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 366 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 361 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 371 of file `trick_memory_interface.hh`.

Referenced by `set_mode()`.

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

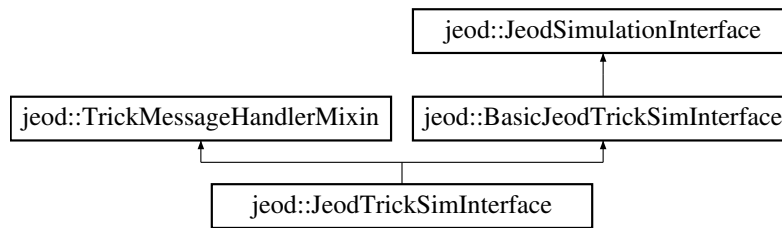
- [trick\\_memory\\_interface.hh](#)
- [trick\\_memory\\_interface.cc](#)
- [trick\\_memory\\_interface\\_alloc.cc](#)
- [trick\\_memory\\_interface\\_attrib.cc](#)

## 8.14 `jeod::JeodTrickSimInterface` Class Reference

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

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

Inheritance diagram for `jeod::JeodTrickSimInterface`:



## Public Member Functions

- [JeodTrickSimInterface](#) ()  
*Non-default constructor.*
- [~JeodTrickSimInterface](#) () override  
*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 296 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 305 of file `trick_sim_interface.hh`.

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

Destructor.

Definition at line 311 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 298 of file `trick_sim_interface.hh`.

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

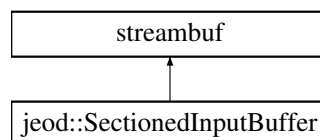
- [trick\\_sim\\_interface.hh](#)

## 8.15 jeod::SectionedInputBuffer Class Reference

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

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

Inheritance diagram for `jeod::SectionedInputBuffer`:



### Public Member Functions

- [~SectionedInputBuffer](#) () override  
*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.*

- `std::streambuf::int_type` `underflow` () override  
*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 89 of file `checkpoint_input_manager.hh`.

## 8.15.2 Constructor & Destructor Documentation

### 8.15.2.1 `jeod::SectionedInputBuffer::~SectionedInputBuffer ( )` `[inline]`, `[override]`

Destructor.

For now, this does nothing.

Definition at line 98 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

<code>in</code>	<code>stream</code>	Checkpoint file input file stream
<code>in</code>	<code>spos</code>	Section data start position
<code>in</code>	<code>epos</code>	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 123 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 105 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 )` `[override], [private]`

Get a character in the case of depletion of the read buffer.

For now, the buffer is always depleted.

#### Returns

Character read from the underlying file.

Definition at line 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 90 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 159 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 164 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 154 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 149 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 137 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 143 of file `checkpoint_input_manager.hh`.

Referenced by `activate()`.

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

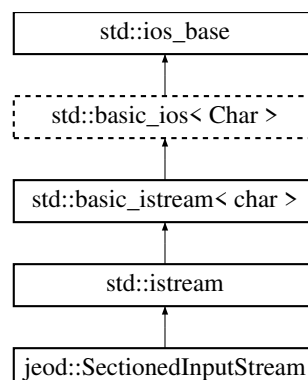
- [checkpoint\\_input\\_manager.hh](#)
- [checkpoint\\_input\\_manager.cc](#)

## 8.16 `jeod::SectionedInputStream` Class Reference

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

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

Inheritance diagram for `jeod::SectionedInputStream`:



### Public Member Functions

- [SectionedInputStream \(\)](#)

Construct a [SectionedInputStream](#) object.

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

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

- [~SectionedInputStream](#) () override

Destruct a [SectionedInputStream](#) object.

- 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 309 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 ) [override]

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 244 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 289 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 346 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 337 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 310 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 387 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 399 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 393 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 370 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 365 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 381 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 375 of file checkpoint\_input\_manager.hh.

Referenced by activate(), deactivate(), is\_activatable(), operator!(), and SectionedInputStream().

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

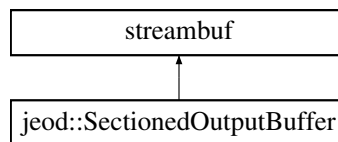
- [checkpoint\\_input\\_manager.hh](#)
- [checkpoint\\_input\\_manager.cc](#)

## 8.17 jeod::SectionedOutputBuffer Class Reference

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

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

Inheritance diagram for jeod::SectionedOutputBuffer:



### Public Member Functions

- [~SectionedOutputBuffer](#) () override  
*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.*
- std::streambuf::int\_type [overflow](#) (std::streambuf::int\_type c) override  
*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 86 of file `checkpoint_output_manager.hh`.

### 8.17.2 Constructor & Destructor Documentation

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

Destructor.

For now, this does nothing.

Definition at line 95 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 121 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 102 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* ) [override], [private]

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

For now, the buffer always overflows.

**Returns**

Status: EOF => failed

**Parameters**

<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 87 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 137 of file checkpoint\_output\_manager.hh.

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

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

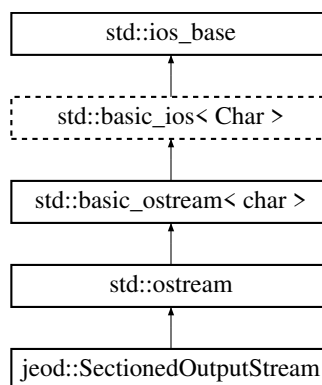
- [checkpoint\\_output\\_manager.hh](#)
- [checkpoint\\_output\\_manager.cc](#)

## 8.18 jeod::SectionedOutputStream Class Reference

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

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

Inheritance diagram for `jeod::SectionedOutputStream`:



## Public Member Functions

- [SectionedOutputStream](#) ()  
Construct a [SectionedOutputStream](#) object.
- [SectionedOutputStream](#) (const [SectionedOutputStream](#) &)  
Construct a [SectionedOutputStream](#) object by copying from another.
- [~SectionedOutputStream](#) () override  
Destruct a [SectionedOutputStream](#) object.
- 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 &section\_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 168 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 ) [override]

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 235 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 288 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 204 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 195 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 169 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 261 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 255 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 229 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 224 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.

Definition at line 244 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.

Definition at line 239 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 234 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.

Definition at line 249 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 472 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

<code>in</code>	<code>start</code>	Start position
<code>in</code>	<code>end</code>	End position

Definition at line 488 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 481 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 476 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 79 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 109 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 104 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 94 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::JeodTrick10-`

MemoryInterface::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 99 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 89 of file `sim_interface_messages.hh`.

Referenced by `jeod::JeodSimulationInterface::create_integrator_interface()`, `jeod::JeodSimulationInterface::get_address_at_name()`, `jeod::JeodSimulationInterface::get_checkpoint_reader()`, `jeod::JeodSimulationInterface::get_checkpoint_writer()`, `jeod::JeodSimulationInterface::get_job_cycle()`, `jeod::JeodSimulationInterface::get_memory_interface()`, `jeod::JeodSimulationInterface::get_name_at_address()`, and `jeod::JeodSimulationInterface::JeodSimulationInterface()`.

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

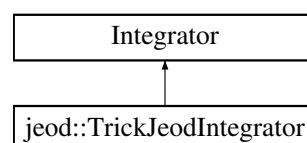
- [sim\\_interface\\_messages.hh](#)
- [sim\\_interface\\_messages.cc](#)

## 8.21 jeod::TrickJeodIntegrator Class Reference

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

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

Inheritance diagram for `jeod::TrickJeodIntegrator`:



### Public Member Functions

- [~TrickJeodIntegrator](#) () override  
*Destructor.*
- `int integrate` () override  
*Does nothing.*
- `void initialize` (int, double) override  
*Does nothing.*

### 8.21.1 Detailed Description

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

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

### 8.21.2 Constructor & Destructor Documentation

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

Destructor.

Definition at line 96 of file `jeod_trick_integrator.hh`.

### 8.21.3 Member Function Documentation

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

Does nothing.

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

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

Does nothing.

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

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

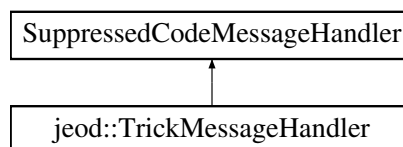
- [jeod\\_trick\\_integrator.hh](#)

## 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.*
- [~TrickMessageHandler](#) (void) override  
*Destructor.*
- void [register\\_contents](#) (void) override  
*Register the [TrickMessageHandler](#)'s checkpointable contents.*

## Protected Member Functions

- void [process\\_message](#) (int severity, const char \*prefix, const char \*file, unsigned int line, const char \*msg\_code, const char \*format, va\_list args) const override

*Handle a message.*

## Private Member Functions

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

*Not implemented.*

*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 91 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 108 of file `trick_message_handler.hh`.

#### 8.22.2.2 jeod::TrickMessageHandler::~~TrickMessageHandler ( void ) [inline],[override]

Destructor.

Definition at line 113 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` `[override]`, `[protected]`

Handle a message.

All calls to the message-generating MessageHandler methods eventually result in a call to thisTrickMessageHandler::process\_message method. This method uses the [Trick](#) function `exec_terminate` to process fatal errors. The [Trick](#) function `send_hs` is used for all non-fatal messages, but only if the message severity is at or below the message suppression level.

#### Parameters

in	<i>severity</i>	Severity level
in	<i>prefix</i>	Message prefix (e.g., Error)
in	<i>file</i>	Typically <b>FILE</b>
in	<i>line</i>	Typically <b>LINE</b>
in	<i>msg_code</i>	Message code
in	<i>format</i>	sprintf format
in	<i>args</i>	Arguments

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

References `MAX_MSG_SIZE`.

**8.22.3.3** `void jeod::TrickMessageHandler::register_contents ( void )` `[override]`

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

Definition at line 62 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 92 of file `trick_message_handler.hh`.

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

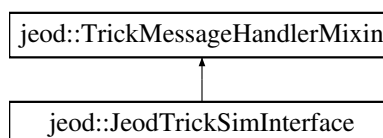
- [trick\\_message\\_handler.hh](#)
- [trick\\_message\\_handler.cc](#)

## 8.23 jeod::TrickMessageHandlerMixin Class Reference

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

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

Inheritance diagram for `jeod::TrickMessageHandlerMixin`:





## Public Member Functions

- [TrickMessageHandlerMixin](#) ()  
*Default 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 247 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 255 of file `trick_sim_interface.hh`.

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

Destructor.

Definition at line 260 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 248 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 269 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 109 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 99 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 <cstddef>
#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, [44](#)
- ~JeodIntegratorInterface
  - jeod::JeodIntegratorInterface, [52](#)
- ~JeodMemoryInterface
  - jeod::JeodMemoryInterface, [54](#)
- ~JeodSimulationInterface
  - jeod::JeodSimulationInterface, [61](#)
- ~JeodTrick10MemoryInterface
  - jeod::JeodTrick10MemoryInterface, [70](#)
- ~JeodTrickIntegrator
  - jeod::JeodTrickIntegrator, [76](#)
- ~JeodTrickMemoryInterface
  - jeod::JeodTrickMemoryInterface, [82](#)
- ~JeodTrickSimInterface
  - jeod::JeodTrickSimInterface, [91](#)
- ~SectionedInputBuffer
  - jeod::SectionedInputBuffer, [93](#)
- ~SectionedInputStream
  - jeod::SectionedInputStream, [100](#)
- ~SectionedOutputBuffer
  - jeod::SectionedOutputBuffer, [104](#)
- ~SectionedOutputStream
  - jeod::SectionedOutputStream, [108](#)
- ~TrickJeodIntegrator
  - jeod::TrickJeodIntegrator, [116](#)
- ~TrickMessageHandler
  - jeod::TrickMessageHandler, [117](#)
- ~TrickMessageHandlerMixin
  - jeod::TrickMessageHandlerMixin, [119](#)
- activate
  - jeod::SectionedInputBuffer, [94](#)
  - jeod::SectionedInputStream, [100](#)
  - jeod::SectionedOutputBuffer, [104](#)
  - jeod::SectionedOutputStream, [108](#)
- add\_integrable\_object
  - jeod::JeodDynbodyIntegrationLoop, [45](#)
- add\_sim\_object
  - jeod::JeodDynbodyIntegrationLoop, [46](#)
- add\_sim\_object\_bodies
  - jeod::JeodDynbodyIntegrationLoop, [46](#)
- allocation\_map
  - jeod::JeodTrickMemoryInterface, [89](#)
- AllocationMap
  - jeod::JeodTrickMemoryInterface, [81](#)
- AllocationMapEntry
  - jeod::JeodTrickMemoryInterface::AllocationMapEntry, [21](#)
- at\_eof
  - jeod::SectionedInputBuffer, [95](#)
- BasicJeodTrickSimInterface
  - jeod::BasicJeodTrickSimInterface, [24](#), [25](#)
- buf
  - jeod::SectionedInputBuffer, [95](#)
- CLASS
  - SimInterface, [15](#)
- CheckPointInputManager
  - jeod::CheckPointInputManager, [31](#)
  - jeod::SectionedInputStream, [101](#)
- CheckPointOutputManager
  - jeod::CheckPointOutputManager, [36](#)
  - jeod::SectionedOutputStream, [110](#)
- Checkpoint
  - jeod::JeodSimulationInterface, [61](#)
- checkpoint\_allocations
  - jeod::BasicJeodTrickSimInterface, [25](#)
  - jeod::JeodTrick10MemoryInterface, [70](#)
  - jeod::JeodTrickMemoryInterface, [82](#)
- checkpoint\_containers
  - jeod::BasicJeodTrickSimInterface, [25](#)
  - jeod::JeodTrick10MemoryInterface, [70](#)
  - jeod::JeodTrickMemoryInterface, [82](#)
- checkpoint\_file\_name
  - jeod::BasicJeodTrickSimInterface, [28](#)
- checkpoint\_input\_manager.cc, [121](#)
- checkpoint\_input\_manager.hh, [121](#)
- checkpoint\_output\_manager.cc, [122](#)
- checkpoint\_output\_manager.hh, [122](#)
- checkpoint\_reader
  - jeod::BasicJeodTrickSimInterface, [28](#)
- checkpoint\_writer
  - jeod::BasicJeodTrickSimInterface, [29](#)
- class\_declarations.hh, [123](#)
- close\_checkpoint\_file
  - jeod::BasicJeodTrickSimInterface, [25](#)
- close\_restart\_file
  - jeod::BasicJeodTrickSimInterface, [25](#)
- collect\_derivatives
  - jeod::JeodDynbodyIntegrationLoop, [46](#)
- config.hh, [123](#)
- config\_test\_harness.hh, [124](#)
- config\_trick10.hh, [124](#)
- configure
  - jeod::JeodSimulationInterface, [62](#)

- construct\_identifier
  - jeod::JeodTrickMemoryInterface, [82](#)
- Construction
  - jeod::JeodSimulationInterface, [61](#)
- container
  - jeod::JeodTrickMemoryInterface::ContainerListEntry, [41](#)
- container\_list
  - jeod::JeodTrickMemoryInterface, [89](#)
- ContainerList
  - jeod::JeodTrickMemoryInterface, [81](#)
- ContainerListEntry
  - jeod::JeodTrickMemoryInterface::ContainerListEntry, [40](#)
- create\_integrator\_interface
  - jeod::JeodSimulationInterface, [62](#)
- create\_integrator\_internal
  - jeod::BasicJeodTrickSimInterface, [25](#)
  - jeod::JeodSimulationInterface, [62](#)
- create\_section\_reader
  - jeod::CheckPointInputManager, [31](#), [32](#)
- create\_section\_writer
  - jeod::CheckPointOutputManager, [36](#), [37](#)
- create\_trick\_section\_reader
  - jeod::CheckPointInputManager, [32](#)
- create\_trick\_section\_writer
  - jeod::CheckPointOutputManager, [37](#)
- curr\_pos
  - jeod::SectionedInputBuffer, [95](#)
- current\_reader
  - jeod::CheckPointInputManager, [34](#)
- current\_writer
  - jeod::CheckPointOutputManager, [39](#)
- deactivate
  - jeod::SectionedInputBuffer, [94](#)
  - jeod::SectionedInputStream, [100](#)
  - jeod::SectionedOutputBuffer, [105](#)
  - jeod::SectionedOutputStream, [109](#)
- Dead
  - jeod::JeodSimulationInterface, [61](#)
- default\_first\_step\_deriv
  - jeod::JeodTrickIntegrator, [78](#)
- deregister\_allocation
  - jeod::JeodMemoryInterface, [55](#)
  - jeod::JeodTrickMemoryInterface, [83](#)
- deregister\_container
  - jeod::JeodMemoryInterface, [55](#)
  - jeod::JeodTrick10MemoryInterface, [70](#)
  - jeod::JeodTrickMemoryInterface, [83](#)
- deregister\_reader
  - jeod::CheckPointInputManager, [32](#)
- deregister\_writer
  - jeod::CheckPointOutputManager, [37](#)
- deriv\_ephem\_update
  - jeod::JeodDynbodyIntegrationLoop, [50](#)
- dlhandle
  - jeod::JeodTrickMemoryInterface, [89](#)
- dyn\_manager
  - jeod::JeodDynbodyIntegrationLoop, [50](#)
- ER7\_UTILS\_RESTRICT
  - SimInterface, [15](#)
- ER7\_UTILS\_UNUSED
  - SimInterface, [15](#)
- elem\_name
  - jeod::JeodTrickMemoryInterface::ContainerListEntry, [41](#)
- end\_pos
  - jeod::CheckPointInputManager::SectionInfo, [113](#)
  - jeod::SectionedInputBuffer, [95](#)
  - jeod::SectionedInputStream, [102](#)
- er7\_utils, [19](#)
- file\_buf
  - jeod::SectionedInputBuffer, [96](#)
  - jeod::SectionedOutputBuffer, [106](#)
- filename
  - jeod::CheckPointInputManager, [34](#)
  - jeod::CheckPointOutputManager, [39](#)
- find\_attributes
  - jeod::JeodMemoryInterface, [55](#), [56](#)
  - jeod::JeodTrickMemoryInterface, [83](#), [84](#)
- find\_containing\_sim\_object
  - jeod::JeodDynbodyIntegrationLoop, [46](#)
- generic\_message\_handler
  - jeod::BasicJeodTrickSimInterface, [29](#)
- get\_address\_at\_name
  - jeod::JeodMemoryInterface, [56](#)
  - jeod::JeodSimulationInterface, [62](#)
  - jeod::JeodTrick10MemoryInterface, [71](#)
  - jeod::JeodTrickMemoryInterface, [84](#)
- get\_checkpoint\_file\_name
  - jeod::BasicJeodTrickSimInterface, [26](#)
- get\_checkpoint\_reader
  - jeod::JeodSimulationInterface, [63](#)
- get\_checkpoint\_reader\_internal
  - jeod::BasicJeodTrickSimInterface, [26](#)
  - jeod::JeodSimulationInterface, [63](#)
- get\_checkpoint\_writer
  - jeod::JeodSimulationInterface, [63](#)
- get\_checkpoint\_writer\_internal
  - jeod::BasicJeodTrickSimInterface, [26](#)
  - jeod::JeodSimulationInterface, [63](#)
- get\_container\_id
  - jeod::JeodTrick10MemoryInterface, [71](#)
- get\_dt
  - jeod::JeodTrickIntegrator, [77](#)
- get\_first\_step\_derivs\_flag
  - jeod::JeodTrickIntegrator, [77](#)
- get\_integrator
  - jeod::JeodIntegratorInterface, [53](#)
  - jeod::JeodTrickIntegrator, [77](#)
- get\_job\_cycle
  - jeod::JeodSimulationInterface, [64](#)
- get\_job\_cycle\_internal
  - jeod::BasicJeodTrickSimInterface, [26](#)

- jeod::JeodSimulationInterface, 64
- get\_memory\_interface
  - jeod::JeodSimulationInterface, 64
- get\_memory\_interface\_internal
  - jeod::BasicJeodTrickSimInterface, 27
  - jeod::JeodSimulationInterface, 64
- get\_mode
  - jeod::JeodSimulationInterface, 64
- get\_name\_at\_address
  - jeod::JeodMemoryInterface, 56
  - jeod::JeodSimulationInterface, 65
  - jeod::JeodTrick10MemoryInterface, 71
  - jeod::JeodTrickMemoryInterface, 84
- get\_trick\_checkpoint\_file
  - jeod::JeodTrick10MemoryInterface, 72
  - jeod::JeodTrickMemoryInterface, 85
- gravitation
  - jeod::JeodDynbodyIntegrationLoop, 48
- gravity\_manager
  - jeod::JeodDynbodyIntegrationLoop, 50
- have\_active\_reader
  - jeod::CheckPointInputManager, 33
- have\_active\_writer
  - jeod::CheckPointOutputManager, 38
- id\_length
  - jeod::JeodTrickMemoryInterface, 90
- id\_prefix
  - jeod::JeodTrickMemoryInterface, 90
- implementation\_error
  - jeod::SimInterfaceMessages, 114
- init\_attrjeod\_\_BasicJeodTrickSimInterface
  - jeod::BasicJeodTrickSimInterface, 28
- init\_attrjeod\_\_JeodDynbodyIntegrationLoop
  - jeod::JeodDynbodyIntegrationLoop, 50
- init\_attrjeod\_\_JeodIntegratorInterface
  - jeod::JeodIntegratorInterface, 53
- init\_attrjeod\_\_JeodMemoryInterface
  - jeod::JeodMemoryInterface, 59
- init\_attrjeod\_\_JeodSimulationInterface
  - jeod::JeodSimulationInterface, 66
- init\_attrjeod\_\_JeodTrick10MemoryInterface
  - jeod::JeodTrick10MemoryInterface, 74
- init\_attrjeod\_\_JeodTrickIntegrator
  - jeod::JeodTrickIntegrator, 78
- init\_attrjeod\_\_JeodTrickMemoryInterface
  - jeod::JeodTrickMemoryInterface, 89
- init\_attrjeod\_\_JeodTrickSimInterface
  - jeod::JeodTrickSimInterface, 92
- init\_attrjeod\_\_TrickMessageHandler
  - jeod::TrickMessageHandler, 118
- init\_attrjeod\_\_TrickMessageHandlerMixin
  - jeod::TrickMessageHandlerMixin, 120
- Initialization
  - jeod::JeodSimulationInterface, 61
- initialize
  - jeod::CheckPointInputManager, 33
  - jeod::TrickJeodIntegrator, 116
- initialize\_integ\_loop
  - jeod::JeodDynbodyIntegrationLoop, 48
- InputProcessor
  - jeod::BasicJeodTrickSimInterface, 28
  - jeod::JeodDynbodyIntegrationLoop, 50
  - jeod::JeodIntegratorInterface, 53
  - jeod::JeodMemoryInterface, 59
  - jeod::JeodSimulationInterface, 66
  - jeod::JeodTrick10MemoryInterface, 74
  - jeod::JeodTrickIntegrator, 78
  - jeod::JeodTrickMemoryInterface, 89
  - jeod::JeodTrickSimInterface, 92
  - jeod::TrickMessageHandler, 118
  - jeod::TrickMessageHandlerMixin, 120
- integ\_constructor
  - jeod::JeodDynbodyIntegrationLoop, 50
- integ\_group
  - jeod::JeodDynbodyIntegrationLoop, 51
- integ\_group\_factory
  - jeod::JeodDynbodyIntegrationLoop, 51
- integ\_interface
  - jeod::JeodDynbodyIntegrationLoop, 51
- integrate
  - jeod::TrickJeodIntegrator, 116
- integrate\_dt
  - jeod::JeodDynbodyIntegrationLoop, 48
- integration\_error
  - jeod::SimInterfaceMessages, 114
- interface\_error
  - jeod::SimInterfaceMessages, 114
- interpret\_integration\_type
  - jeod::JeodIntegratorInterface, 53
  - jeod::JeodTrickIntegrator, 77
- is\_activatable
  - jeod::SectionedInputStream, 101
  - jeod::SectionedOutputStream, 109
- is\_active
  - jeod::SectionedInputStream, 102
  - jeod::SectionedOutputStream, 110
- is\_array
  - jeod::JeodTrickMemoryInterface::AllocationMap-Entry, 22
- is\_checkpoint\_restart\_supported
  - jeod::JeodMemoryInterface, 57
  - jeod::JeodTrick10MemoryInterface, 72
  - jeod::JeodTrickMemoryInterface, 85
- is\_copy
  - jeod::SectionedInputStream, 102
  - jeod::SectionedOutputStream, 110
- is\_open
  - jeod::CheckPointInputManager, 34
  - jeod::CheckPointOutputManager, 39
- JEOD\_ATTRIBUTES
  - memory\_attributes.hh, 127
- 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, 61
  - Construction, 61
  - Dead, 61
  - Initialization, 61
  - NumModes, 61
  - Operational, 61
  - PostCheckpoint, 61
  - PreCheckpoint, 61
  - Restart, 61
  - Restore, 61
  - Shutdown, 61
- 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, 34
  - stream, 35
- jeod::CheckPointInputManager::SectionInfo, 111
  - end\_pos, 113
  - SectionInfo, 112
  - start\_pos, 113
- jeod::CheckPointOutputManager, 35
  - CheckPointOutputManager, 36
  - create\_section\_writer, 36, 37
  - create\_trick\_section\_writer, 37
  - current\_writer, 39
  - deregister\_writer, 37
  - filename, 39
  - have\_active\_writer, 38
  - is\_open, 39
  - MemoryManagerWrapper, 39
  - operator=, 38
  - register\_writer, 38
  - section\_end, 39
  - section\_start, 39
  - stream, 39
- jeod::JeodDynbodyIntegrationLoop, 41
  - ~JeodDynbodyIntegrationLoop, 44
  - add\_integrable\_object, 45
  - add\_sim\_object, 46
  - add\_sim\_object\_bodies, 46
  - collect\_derivatives, 46
  - deriv\_ephem\_update, 50
  - dyn\_manager, 50
  - find\_containing\_sim\_object, 46
  - gravitation, 48
  - gravity\_manager, 50
  - init\_attrjeod\_\_JeodDynbodyIntegrationLoop, 50
  - initialize\_integ\_loop, 48
  - InputProcessor, 50
  - integ\_constructor, 50
  - integ\_group, 51
  - integ\_group\_factory, 51
  - integ\_interface, 51
  - integrate\_dt, 48
  - JeodDynbodyIntegrationLoop, 44
  - loop\_sim\_object, 51
  - operator=, 48
  - remove\_integrable\_object, 48
  - remove\_sim\_object, 49
  - remove\_sim\_object\_bodies, 49
  - set\_deriv\_ephem\_update, 49
  - set\_time\_to\_loop\_start, 49
  - time\_manager, 51
  - update\_integration\_group, 49
- jeod::JeodIntegratorInterface, 52
  - ~JeodIntegratorInterface, 52
  - get\_integrator, 53

- init\_attrjeod\_\_JeodIntegratorInterface, 53
- InputProcessor, 53
- interpret\_integration\_type, 53
- jeod::JeodMemoryInterface, 53
  - ~JeodMemoryInterface, 54
  - deregister\_allocation, 55
  - deregister\_container, 55
  - find\_attributes, 55, 56
  - get\_address\_at\_name, 56
  - get\_name\_at\_address, 56
  - init\_attrjeod\_\_JeodMemoryInterface, 59
  - InputProcessor, 59
  - is\_checkpoint\_restart\_supported, 57
  - JeodMemoryInterface, 54, 55
  - operator=, 57
  - pointer\_attributes, 57
  - primitive\_attributes, 57
  - register\_allocation, 58
  - register\_container, 58
  - structure\_attributes, 58
  - void\_pointer\_attributes, 58
- jeod::JeodSimulationInterface, 59
  - ~JeodSimulationInterface, 61
  - configure, 62
  - create\_integrator\_interface, 62
  - create\_integrator\_internal, 62
  - get\_address\_at\_name, 62
  - get\_checkpoint\_reader, 63
  - get\_checkpoint\_reader\_internal, 63
  - get\_checkpoint\_writer, 63
  - get\_checkpoint\_writer\_internal, 63
  - get\_job\_cycle, 64
  - get\_job\_cycle\_internal, 64
  - get\_memory\_interface, 64
  - get\_memory\_interface\_internal, 64
  - get\_mode, 64
  - get\_name\_at\_address, 65
  - init\_attrjeod\_\_JeodSimulationInterface, 66
  - InputProcessor, 66
  - JeodSimulationInterface, 61
  - Mode, 61
  - mode, 66
  - operator=, 65
  - saved\_mode, 66
  - set\_mode, 65
  - sim\_interface, 66
- jeod::JeodSimulationInterfaceInit, 66
  - JeodSimulationInterfaceInit, 67
  - memory\_debug\_level, 67
  - message\_suppress\_id, 67
  - message\_suppress\_location, 67
  - message\_suppression\_level, 68
- jeod::JeodTrick10MemoryInterface, 68
  - ~JeodTrick10MemoryInterface, 70
  - checkpoint\_allocations, 70
  - checkpoint\_containers, 70
  - deregister\_container, 70
  - get\_address\_at\_name, 71
  - get\_container\_id, 71
  - get\_name\_at\_address, 71
  - get\_trick\_checkpoint\_file, 72
  - init\_attrjeod\_\_JeodTrick10MemoryInterface, 74
  - InputProcessor, 74
  - is\_checkpoint\_restart\_supported, 72
  - JeodTrick10MemoryInterface, 70
  - operator=, 72
  - register\_container, 72
  - restore\_allocations, 73
  - restore\_containers, 73
  - translate\_addr\_to\_name, 73
  - translate\_name\_to\_addr, 74
  - trick\_checkpoint\_agent, 74
- jeod::JeodTrickIntegrator, 75
  - ~JeodTrickIntegrator, 76
  - default\_first\_step\_deriv, 78
  - get\_dt, 77
  - get\_first\_step\_derivs\_flag, 77
  - get\_integrator, 77
  - init\_attrjeod\_\_JeodTrickIntegrator, 78
  - InputProcessor, 78
  - interpret\_integration\_type, 77
  - JeodTrickIntegrator, 76
  - operator=, 77
  - reset\_first\_step\_derivs\_flag, 77
  - restore\_first\_step\_derivs\_flag, 78
  - set\_first\_step\_derivs\_flag, 78
  - set\_step\_number, 78
  - set\_time, 78
  - trick\_integrator, 79
- jeod::JeodTrickMemoryInterface, 79
  - ~JeodTrickMemoryInterface, 82
  - allocation\_map, 89
  - AllocationMap, 81
  - checkpoint\_allocations, 82
  - checkpoint\_containers, 82
  - construct\_identifier, 82
  - container\_list, 89
  - ContainerList, 81
  - deregister\_allocation, 83
  - deregister\_container, 83
  - dlhandle, 89
  - find\_attributes, 83, 84
  - get\_address\_at\_name, 84
  - get\_name\_at\_address, 84
  - get\_trick\_checkpoint\_file, 85
  - id\_length, 90
  - id\_prefix, 90
  - init\_attrjeod\_\_JeodTrickMemoryInterface, 89
  - InputProcessor, 89
  - is\_checkpoint\_restart\_supported, 85
  - JeodTrickMemoryInterface, 82
  - mode, 90
  - operator=, 85
  - pointer\_attributes, 85
  - primitive\_attributes, 86
  - register\_allocation, 86

- register\_container, 86
- restore\_allocations, 88
- restore\_containers, 88
- set\_mode, 88
- structure\_attributes, 88
- void\_pointer\_attributes, 89
- jeod::JeodTrickMemoryInterface::AllocationMapEntry, 21
  - AllocationMapEntry, 21
  - is\_array, 22
  - nelements, 22
  - typeid\_info, 22
- jeod::JeodTrickMemoryInterface::ContainerListEntry, 40
  - container, 41
  - ContainerListEntry, 40
  - elem\_name, 41
  - owner, 41
  - owner\_type, 41
- jeod::JeodTrickSimInterface, 90
  - ~JeodTrickSimInterface, 91
  - init\_attrjeod\_\_JeodTrickSimInterface, 92
  - InputProcessor, 92
  - JeodTrickSimInterface, 91
  - operator=, 92
- jeod::SectionedInputBuffer, 92
  - ~SectionedInputBuffer, 93
  - activate, 94
  - at\_eof, 95
  - buf, 95
  - curr\_pos, 95
  - deactivate, 94
  - end\_pos, 95
  - file\_buf, 96
  - operator=, 94
  - SectionedInputBuffer, 93, 94
  - SectionedInputStream, 95
  - start\_pos, 96
  - underflow, 95
- jeod::SectionedInputStream, 96
  - ~SectionedInputStream, 100
  - activate, 100
  - CheckPointInputManager, 101
  - deactivate, 100
  - end\_pos, 102
  - is\_activatable, 101
  - is\_active, 102
  - is\_copy, 102
  - manager, 102
  - operator void \*, 101
  - operator=, 101
  - sectbuf, 102
  - SectionedInputStream, 99, 100
  - start\_pos, 102
  - stream, 102
- jeod::SectionedOutputBuffer, 103
  - ~SectionedOutputBuffer, 104
  - activate, 104
  - deactivate, 105
  - file\_buf, 106
  - operator=, 105
  - overflow, 105
  - SectionedOutputBuffer, 104
  - SectionedOutputStream, 106
- jeod::SectionedOutputStream, 106
  - ~SectionedOutputStream, 108
  - activate, 108
  - CheckPointOutputManager, 110
  - deactivate, 109
  - is\_activatable, 109
  - is\_active, 110
  - is\_copy, 110
  - manager, 110
  - operator void \*, 109
  - operator=, 109
  - sectbuf, 110
  - section\_end, 110
  - section\_start, 110
  - SectionedOutputStream, 108
  - stream, 111
  - tag, 111
- jeod::SimInterfaceMessages, 113
  - implementation\_error, 114
  - integration\_error, 114
  - interface\_error, 114
  - operator=, 114
  - phasing\_error, 115
  - SimInterfaceMessages, 114
  - singleton\_error, 115
- jeod::TrickJeodIntegrator, 115
  - ~TrickJeodIntegrator, 116
  - initialize, 116
  - integrate, 116
- jeod::TrickMessageHandler, 116
  - ~TrickMessageHandler, 117
  - init\_attrjeod\_\_TrickMessageHandler, 118
  - InputProcessor, 118
  - operator=, 117
  - process\_message, 117
  - register\_contents, 118
  - TrickMessageHandler, 117
- jeod::TrickMessageHandlerMixin, 118
  - ~TrickMessageHandlerMixin, 119
  - init\_attrjeod\_\_TrickMessageHandlerMixin, 120
  - InputProcessor, 120
  - message\_handler, 120
  - operator=, 120
  - TrickMessageHandlerMixin, 119
- jeod\_class.hh, 125
- jeod\_integrator\_interface.hh, 125
- jeod\_trick\_integrator.hh, 126
- JeodDynbodyIntegrationLoop
  - jeod::JeodDynbodyIntegrationLoop, 44
- JeodMemoryInterface
  - jeod::JeodMemoryInterface, 54, 55
- JeodSimulationInterface
  - jeod::JeodSimulationInterface, 61

- JeodSimulationInterfaceInit
  - jeod::JeodSimulationInterfaceInit, [67](#)
- JeodTrick10MemoryInterface
  - jeod::JeodTrick10MemoryInterface, [70](#)
- JeodTrickIntegrator
  - jeod::JeodTrickIntegrator, [76](#)
- JeodTrickMemoryInterface
  - jeod::JeodTrickMemoryInterface, [82](#)
- JeodTrickSimInterface
  - jeod::JeodTrickSimInterface, [91](#)
- loop\_sim\_object
  - jeod::JeodDynbodyIntegrationLoop, [51](#)
- MAKE\_MESSAGE\_CODE
  - SimInterface, [17](#)
- MAX\_MSG\_SIZE
  - SimInterface, [17](#)
- manager
  - jeod::SectionedInputStream, [102](#)
  - jeod::SectionedOutputStream, [110](#)
- memory\_attributes.hh, [126](#)
  - JEOD\_ATTRIBUTES, [127](#)
- memory\_debug\_level
  - jeod::JeodSimulationInterfaceInit, [67](#)
- memory\_interface.cc, [127](#)
- memory\_interface.hh, [128](#)
- memory\_manager
  - jeod::BasicJeodTrickSimInterface, [29](#)
- MemoryManagerWrapper
  - jeod::CheckPointOutputManager, [39](#)
- message\_handler
  - jeod::TrickMessageHandlerMixin, [120](#)
- message\_suppress\_id
  - jeod::JeodSimulationInterfaceInit, [67](#)
- message\_suppress\_location
  - jeod::JeodSimulationInterfaceInit, [67](#)
- message\_suppression\_level
  - jeod::JeodSimulationInterfaceInit, [68](#)
- Mode
  - jeod::JeodSimulationInterface, [61](#)
- mode
  - jeod::JeodSimulationInterface, [66](#)
  - jeod::JeodTrickMemoryInterface, [90](#)
- Models, [11](#)
- nelements
  - jeod::JeodTrickMemoryInterface::AllocationMap-Entry, [22](#)
- NumModes
  - jeod::JeodSimulationInterface, [61](#)
- open\_checkpoint\_file
  - jeod::BasicJeodTrickSimInterface, [27](#)
- open\_restart\_file
  - jeod::BasicJeodTrickSimInterface, [27](#)
- Operational
  - jeod::JeodSimulationInterface, [61](#)
- operator void \*
- jeod::SectionedInputStream, [101](#)
- jeod::SectionedOutputStream, [109](#)
- operator=
  - jeod::BasicJeodTrickSimInterface, [27](#)
  - jeod::CheckPointInputManager, [33](#)
  - jeod::CheckPointOutputManager, [38](#)
  - jeod::JeodDynbodyIntegrationLoop, [48](#)
  - jeod::JeodMemoryInterface, [57](#)
  - jeod::JeodSimulationInterface, [65](#)
  - jeod::JeodTrick10MemoryInterface, [72](#)
  - jeod::JeodTrickIntegrator, [77](#)
  - jeod::JeodTrickMemoryInterface, [85](#)
  - jeod::JeodTrickSimInterface, [92](#)
  - jeod::SectionedInputBuffer, [94](#)
  - jeod::SectionedInputStream, [101](#)
  - jeod::SectionedOutputBuffer, [105](#)
  - jeod::SectionedOutputStream, [109](#)
  - jeod::SimInterfaceMessages, [114](#)
  - jeod::TrickMessageHandler, [117](#)
  - jeod::TrickMessageHandlerMixin, [120](#)
- overflow
  - jeod::SectionedOutputBuffer, [105](#)
- owner
  - jeod::JeodTrickMemoryInterface::ContainerList-Entry, [41](#)
- owner\_type
  - jeod::JeodTrickMemoryInterface::ContainerList-Entry, [41](#)
- PATH
  - SimInterface, [17](#)
- phasing\_error
  - jeod::SimInterfaceMessages, [115](#)
- pointer\_attributes
  - jeod::JeodMemoryInterface, [57](#)
  - jeod::JeodTrickMemoryInterface, [85](#)
- PostCheckpoint
  - jeod::JeodSimulationInterface, [61](#)
- PreCheckpoint
  - jeod::JeodSimulationInterface, [61](#)
- primitive\_attributes
  - jeod::JeodMemoryInterface, [57](#)
  - jeod::JeodTrickMemoryInterface, [86](#)
- process\_message
  - jeod::TrickMessageHandler, [117](#)
- register\_allocation
  - jeod::JeodMemoryInterface, [58](#)
  - jeod::JeodTrickMemoryInterface, [86](#)
- register\_container
  - jeod::JeodMemoryInterface, [58](#)
  - jeod::JeodTrick10MemoryInterface, [72](#)
  - jeod::JeodTrickMemoryInterface, [86](#)
- register\_contents
  - jeod::TrickMessageHandler, [118](#)
- register\_reader
  - jeod::CheckPointInputManager, [33](#)
- register\_writer
  - jeod::CheckPointOutputManager, [38](#)

- remove\_integrable\_object
  - jeod::JeodDynbodyIntegrationLoop, 48
- remove\_sim\_object
  - jeod::JeodDynbodyIntegrationLoop, 49
- remove\_sim\_object\_bodies
  - jeod::JeodDynbodyIntegrationLoop, 49
- reset\_first\_step\_derivs\_flag
  - jeod::JeodTrickIntegrator, 77
- Restart
  - jeod::JeodSimulationInterface, 61
- Restore
  - jeod::JeodSimulationInterface, 61
- restore\_allocations
  - jeod::BasicJeodTrickSimInterface, 27
  - jeod::JeodTrick10MemoryInterface, 73
  - jeod::JeodTrickMemoryInterface, 88
- restore\_containers
  - jeod::BasicJeodTrickSimInterface, 27
  - jeod::JeodTrick10MemoryInterface, 73
  - jeod::JeodTrickMemoryInterface, 88
- restore\_first\_step\_derivs\_flag
  - jeod::JeodTrickIntegrator, 78
- saved\_mode
  - jeod::JeodSimulationInterface, 66
- sectbuf
  - jeod::SectionedInputStream, 102
  - jeod::SectionedOutputStream, 110
- section\_end
  - jeod::BasicJeodTrickSimInterface, 29
  - jeod::CheckPointInputManager, 34
  - jeod::CheckPointOutputManager, 39
  - jeod::SectionedOutputStream, 110
- section\_start
  - jeod::BasicJeodTrickSimInterface, 29
  - jeod::CheckPointInputManager, 34
  - jeod::CheckPointOutputManager, 39
  - jeod::SectionedOutputStream, 110
- SectionInfo
  - jeod::CheckPointInputManager::SectionInfo, 112
- SectionedInputBuffer
  - jeod::SectionedInputBuffer, 93, 94
- SectionedInputStream
  - jeod::SectionedInputBuffer, 95
  - jeod::SectionedInputStream, 99, 100
- SectionedOutputBuffer
  - jeod::SectionedOutputBuffer, 104
- SectionedOutputStream
  - jeod::SectionedOutputBuffer, 106
  - jeod::SectionedOutputStream, 108
- sections
  - jeod::CheckPointInputManager, 34
- set\_checkpoint\_file\_name
  - jeod::BasicJeodTrickSimInterface, 28
- set\_deriv\_ephem\_update
  - jeod::JeodDynbodyIntegrationLoop, 49
- set\_first\_step\_derivs\_flag
  - jeod::JeodTrickIntegrator, 78
- set\_mode
  - jeod::BasicJeodTrickSimInterface, 28
  - jeod::JeodSimulationInterface, 65
  - jeod::JeodTrickMemoryInterface, 88
- set\_step\_number
  - jeod::JeodTrickIntegrator, 78
- set\_time
  - jeod::JeodTrickIntegrator, 78
- set\_time\_to\_loop\_start
  - jeod::JeodDynbodyIntegrationLoop, 49
- Shutdown
  - jeod::JeodSimulationInterface, 61
- sim\_interface
  - jeod::JeodSimulationInterface, 66
- sim\_interface\_messages.cc, 128
- sim\_interface\_messages.hh, 129
- 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, 114
- simulation\_interface.cc, 129
- simulation\_interface.hh, 130
- singleton\_error
  - jeod::SimInterfaceMessages, 115
- start\_pos
  - jeod::CheckPointInputManager::SectionInfo, 113
  - jeod::SectionedInputBuffer, 96
  - jeod::SectionedInputStream, 102
- stream
  - jeod::CheckPointInputManager, 35
  - jeod::CheckPointOutputManager, 39
  - jeod::SectionedInputStream, 102
  - jeod::SectionedOutputStream, 111
- structure\_attributes
  - jeod::JeodMemoryInterface, 58
  - jeod::JeodTrickMemoryInterface, 88
- tag
  - jeod::SectionedOutputStream, 111
- time\_manager
  - jeod::JeodDynbodyIntegrationLoop, 51
- translate\_addr\_to\_name
  - jeod::JeodTrick10MemoryInterface, 73
- translate\_name\_to\_addr
  - jeod::JeodTrick10MemoryInterface, 74
- Trick, 20
- trick10\_memory\_interface.cc, 130
- trick10\_memory\_interface.hh, 131

- trick\_MM
  - SimInterface, [17](#), [18](#)
- trick\_checkpoint\_agent
  - jeod::JeodTrick10MemoryInterface, [74](#)
- trick\_curr\_integ
  - SimInterface, [17](#)
- trick\_dynbody\_integ\_loop.cc, [132](#)
- trick\_dynbody\_integ\_loop.hh, [132](#)
- trick\_integrator
  - jeod::JeodTrickIntegrator, [79](#)
- trick\_memory\_interface
  - jeod::BasicJeodTrickSimInterface, [29](#)
- trick\_memory\_interface.cc, [133](#)
- trick\_memory\_interface.hh, [133](#)
- trick\_memory\_interface\_alloc.cc, [134](#)
- trick\_memory\_interface\_attr.cc, [135](#)
- trick\_memory\_interface\_chkpt.cc, [135](#)
- trick\_memory\_interface\_xlate.cc, [136](#)
- trick\_message\_handler.cc, [137](#)
- trick\_message\_handler.hh, [137](#)
- trick\_sim\_interface.cc, [138](#)
- trick\_sim\_interface.hh, [138](#)
- TrickMessageHandler
  - jeod::TrickMessageHandler, [117](#)
- TrickMessageHandlerMixin
  - jeod::TrickMessageHandlerMixin, [119](#)
- typeid\_info
  - jeod::JeodTrickMemoryInterface::AllocationMapEntry, [22](#)
- underflow
  - jeod::SectionedInputBuffer, [95](#)
- update\_integration\_group
  - jeod::JeodDynbodyIntegrationLoop, [49](#)
- Utils, [12](#)
- void\_pointer\_attributes
  - jeod::JeodMemoryInterface, [58](#)
  - jeod::JeodTrickMemoryInterface, [89](#)