

# DynamicBodyModel

5.1

Generated by Doxygen 1.8.5

Mon Jul 31 2023 11:43:44



# 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	Dynamics . . . . .	12
6.2.1	Detailed Description . . . . .	12
6.3	DynBody . . . . .	13
6.3.1	Detailed Description . . . . .	14
6.3.2	Macro Definition Documentation . . . . .	14
6.3.2.1	PATH . . . . .	14
<b>7</b>	<b>Namespace Documentation</b>	<b>15</b>
7.1	jeod Namespace Reference . . . . .	15
7.1.1	Detailed Description . . . . .	16
7.1.2	Function Documentation . . . . .	16
7.1.2.1	accumulate_forces . . . . .	16
7.1.2.2	accumulate_forces . . . . .	17
7.1.2.3	accumulate_torques . . . . .	18
7.1.2.4	accumulate_torques . . . . .	18
7.1.2.5	check_frame_ownership . . . . .	18

7.1.2.6	<a href="#">collect_insert</a>	18
7.1.2.7	<a href="#">collect_push_back</a>	19
7.1.2.8	<a href="#">release_vector</a>	19
<b>8</b>	<b>Data Structure Documentation</b>	<b>21</b>
8.1	<a href="#">jeod::BodyForceCollect Class Reference</a>	21
8.1.1	<a href="#">Detailed Description</a>	22
8.1.2	<a href="#">Constructor &amp; Destructor Documentation</a>	22
8.1.2.1	<a href="#">BodyForceCollect</a>	22
8.1.2.2	<a href="#">BodyForceCollect</a>	22
8.1.2.3	<a href="#">~BodyForceCollect</a>	22
8.1.3	<a href="#">Member Function Documentation</a>	23
8.1.3.1	<a href="#">operator=</a>	23
8.1.4	<a href="#">Field Documentation</a>	23
8.1.4.1	<a href="#">collect_effector_forc</a>	23
8.1.4.2	<a href="#">collect_effector_torq</a>	23
8.1.4.3	<a href="#">collect_envron_forc</a>	23
8.1.4.4	<a href="#">collect_envron_torq</a>	23
8.1.4.5	<a href="#">collect_no_xmit_forc</a>	23
8.1.4.6	<a href="#">collect_no_xmit_torq</a>	24
8.1.4.7	<a href="#">effector_forc</a>	24
8.1.4.8	<a href="#">effector_torq</a>	24
8.1.4.9	<a href="#">envron_forc</a>	24
8.1.4.10	<a href="#">envron_torq</a>	24
8.1.4.11	<a href="#">extern_forc_inrtl</a>	24
8.1.4.12	<a href="#">extern_forc_struct</a>	25
8.1.4.13	<a href="#">extern_torq_body</a>	25
8.1.4.14	<a href="#">extern_torq_struct</a>	25
8.1.4.15	<a href="#">inertial_torq</a>	25
8.1.4.16	<a href="#">no_xmit_forc</a>	25
8.1.4.17	<a href="#">no_xmit_torq</a>	26
8.2	<a href="#">jeod::BodyRefFrame Class Reference</a>	26
8.2.1	<a href="#">Detailed Description</a>	27
8.2.2	<a href="#">Constructor &amp; Destructor Documentation</a>	27
8.2.2.1	<a href="#">BodyRefFrame</a>	27
8.2.2.2	<a href="#">BodyRefFrame</a>	27
8.2.2.3	<a href="#">~BodyRefFrame</a>	27
8.2.3	<a href="#">Member Function Documentation</a>	27
8.2.3.1	<a href="#">operator=</a>	27
8.2.4	<a href="#">Friends And Related Function Documentation</a>	27

8.2.4.1	<a href="#">init_attrjeod__BodyRefFrame</a>	27
8.2.4.2	<a href="#">InputProcessor</a>	27
8.2.5	<a href="#">Field Documentation</a>	27
8.2.5.1	<a href="#">initialized_items</a>	27
8.2.5.2	<a href="#">mass_point</a>	27
8.3	<a href="#">jeod::BodyWrenchCollect Class Reference</a>	28
8.3.1	<a href="#">Detailed Description</a>	28
8.3.2	<a href="#">Constructor &amp; Destructor Documentation</a>	28
8.3.2.1	<a href="#">BodyWrenchCollect</a>	28
8.3.2.2	<a href="#">~BodyWrenchCollect</a>	29
8.3.2.3	<a href="#">BodyWrenchCollect</a>	29
8.3.3	<a href="#">Member Function Documentation</a>	29
8.3.3.1	<a href="#">accumulate</a>	29
8.3.3.2	<a href="#">accumulate</a>	29
8.3.3.3	<a href="#">operator=</a>	29
8.3.4	<a href="#">Field Documentation</a>	29
8.3.4.1	<a href="#">collect_wrench</a>	29
8.4	<a href="#">jeod::CInterfaceForce Class Reference</a>	30
8.4.1	<a href="#">Detailed Description</a>	30
8.4.2	<a href="#">Constructor &amp; Destructor Documentation</a>	31
8.4.2.1	<a href="#">CInterfaceForce</a>	31
8.4.2.2	<a href="#">CInterfaceForce</a>	31
8.4.2.3	<a href="#">~CInterfaceForce</a>	31
8.4.2.4	<a href="#">CInterfaceForce</a>	31
8.4.3	<a href="#">Member Function Documentation</a>	31
8.4.3.1	<a href="#">operator=</a>	31
8.5	<a href="#">jeod::CInterfaceTorque Class Reference</a>	31
8.5.1	<a href="#">Detailed Description</a>	32
8.5.2	<a href="#">Constructor &amp; Destructor Documentation</a>	32
8.5.2.1	<a href="#">CInterfaceTorque</a>	32
8.5.2.2	<a href="#">CInterfaceTorque</a>	32
8.5.2.3	<a href="#">~CInterfaceTorque</a>	33
8.5.2.4	<a href="#">CInterfaceTorque</a>	33
8.5.3	<a href="#">Member Function Documentation</a>	33
8.5.3.1	<a href="#">operator=</a>	33
8.6	<a href="#">jeod::CollectForce Class Reference</a>	33
8.6.1	<a href="#">Detailed Description</a>	34
8.6.2	<a href="#">Constructor &amp; Destructor Documentation</a>	34
8.6.2.1	<a href="#">CollectForce</a>	34
8.6.2.2	<a href="#">CollectForce</a>	35

8.6.2.3	CollectForce	35
8.6.2.4	CollectForce	35
8.6.2.5	~CollectForce	35
8.6.2.6	CollectForce	35
8.6.3	Member Function Documentation	35
8.6.3.1	create	35
8.6.3.2	create	36
8.6.3.3	create	36
8.6.3.4	create	36
8.6.3.5	create	36
8.6.3.6	is_active	37
8.6.3.7	operator=	37
8.6.3.8	operator==	37
8.6.3.9	operator[]	37
8.6.3.10	operator[]	37
8.6.4	Field Documentation	38
8.6.4.1	active	38
8.6.4.2	force	38
8.7	jeod::CollectTorque Class Reference	38
8.7.1	Detailed Description	39
8.7.2	Constructor & Destructor Documentation	40
8.7.2.1	CollectTorque	40
8.7.2.2	CollectTorque	40
8.7.2.3	CollectTorque	40
8.7.2.4	CollectTorque	40
8.7.2.5	~CollectTorque	40
8.7.2.6	CollectTorque	41
8.7.3	Member Function Documentation	41
8.7.3.1	create	41
8.7.3.2	create	41
8.7.3.3	create	41
8.7.3.4	create	41
8.7.3.5	create	42
8.7.3.6	is_active	42
8.7.3.7	operator=	42
8.7.3.8	operator==	42
8.7.3.9	operator[]	42
8.7.3.10	operator[]	43
8.7.4	Field Documentation	43
8.7.4.1	active	43

8.7.4.2	torque	43
8.8	jeod::DynBody Class Reference	43
8.8.1	Detailed Description	49
8.8.2	Constructor & Destructor Documentation	49
8.8.2.1	DynBody	49
8.8.2.2	~DynBody	49
8.8.2.3	DynBody	50
8.8.3	Member Function Documentation	50
8.8.3.1	activate	50
8.8.3.2	add_control	50
8.8.3.3	add_integrable_object	50
8.8.3.4	add_mass_body	50
8.8.3.5	add_mass_body	50
8.8.3.6	add_mass_body_frames	51
8.8.3.7	add_mass_body_validate	51
8.8.3.8	add_mass_point	51
8.8.3.9	attach_child	51
8.8.3.10	attach_child	52
8.8.3.11	attach_establish_links	52
8.8.3.12	attach_to	52
8.8.3.13	attach_to	53
8.8.3.14	attach_to_frame	53
8.8.3.15	attach_to_frame	53
8.8.3.16	attach_to_frame	53
8.8.3.17	attach_to_frame	54
8.8.3.18	attach_update_properties	54
8.8.3.19	attach_validate_child	54
8.8.3.20	attach_validate_parent	55
8.8.3.21	clear_integrable_objects	55
8.8.3.22	collect_forces_and_torques	56
8.8.3.23	compute_derived_state_forward	56
8.8.3.24	compute_derived_state_reverse	56
8.8.3.25	compute_ref_point_transform	56
8.8.3.26	compute_state_elements_forward	57
8.8.3.27	compute_state_elements_reverse	57
8.8.3.28	compute_vehicle_point_derivatives	57
8.8.3.29	compute_vehicle_point_states	58
8.8.3.30	create_body_integrators	58
8.8.3.31	create_integrators	58
8.8.3.32	deactivate	60

8.8.3.33	<a href="#">destroy_integrators</a>	60
8.8.3.34	<a href="#">detach</a>	60
8.8.3.35	<a href="#">detach</a>	60
8.8.3.36	<a href="#">detach_mass_body_frames</a>	61
8.8.3.37	<a href="#">detach_mass_internal</a>	61
8.8.3.38	<a href="#">find_body_frame</a>	62
8.8.3.39	<a href="#">find_vehicle_point</a>	62
8.8.3.40	<a href="#">get_dynamics_integration_group</a>	62
8.8.3.41	<a href="#">get_initialized_states</a>	63
8.8.3.42	<a href="#">get_integrable_objects</a>	63
8.8.3.43	<a href="#">get_parent_body</a>	63
8.8.3.44	<a href="#">get_parent_body_internal</a>	63
8.8.3.45	<a href="#">get_root_body</a>	63
8.8.3.46	<a href="#">get_root_body_internal</a>	64
8.8.3.47	<a href="#">initialize_controls</a>	64
8.8.3.48	<a href="#">initialize_model</a>	64
8.8.3.49	<a href="#">initialized_states_contains</a>	64
8.8.3.50	<a href="#">integrate</a>	65
8.8.3.51	<a href="#">is_root_body</a>	65
8.8.3.52	<a href="#">migrate_integrable_objects</a>	65
8.8.3.53	<a href="#">operator=</a>	65
8.8.3.54	<a href="#">process_dynamic_attachment</a>	66
8.8.3.55	<a href="#">propagate_state</a>	66
8.8.3.56	<a href="#">propagate_state_from_composite</a>	66
8.8.3.57	<a href="#">propagate_state_from_structure</a>	67
8.8.3.58	<a href="#">remove_integrable_object</a>	67
8.8.3.59	<a href="#">remove_mass_body</a>	67
8.8.3.60	<a href="#">reset_controls</a>	68
8.8.3.61	<a href="#">reset_integrators</a>	68
8.8.3.62	<a href="#">rot_integ</a>	68
8.8.3.63	<a href="#">set_attitude_left_quaternion</a>	68
8.8.3.64	<a href="#">set_attitude_matrix</a>	69
8.8.3.65	<a href="#">set_attitude_rate</a>	69
8.8.3.66	<a href="#">set_attitude_right_quaternion</a>	69
8.8.3.67	<a href="#">set_integ_frame</a>	70
8.8.3.68	<a href="#">set_integ_frame</a>	70
8.8.3.69	<a href="#">set_name</a>	70
8.8.3.70	<a href="#">set_position</a>	71
8.8.3.71	<a href="#">set_state</a>	71
8.8.3.72	<a href="#">set_state_source</a>	71



8.8.3.73	<a href="#">set_state_source_internal</a>	72
8.8.3.74	<a href="#">set_velocity</a>	72
8.8.3.75	<a href="#">sort_controls</a>	72
8.8.3.76	<a href="#">switch_integration_frames</a>	72
8.8.3.77	<a href="#">switch_integration_frames</a>	73
8.8.3.78	<a href="#">trans_integ</a>	73
8.8.3.79	<a href="#">update_integrated_state</a>	74
8.8.4	<a href="#">Friends And Related Function Documentation</a>	74
8.8.4.1	<a href="#">init_attrjeod__DynBody</a>	74
8.8.4.2	<a href="#">InputProcessor</a>	74
8.8.5	<a href="#">Field Documentation</a>	74
8.8.5.1	<a href="#">associated_integrable_objects</a>	74
8.8.5.2	<a href="#">attitude_source</a>	74
8.8.5.3	<a href="#">autoupdate_vehicle_points</a>	74
8.8.5.4	<a href="#">collect</a>	75
8.8.5.5	<a href="#">composite_body</a>	75
8.8.5.6	<a href="#">compute_point_derivative</a>	75
8.8.5.7	<a href="#">core_body</a>	75
8.8.5.8	<a href="#">derivs</a>	75
8.8.5.9	<a href="#">dyn_children</a>	76
8.8.5.10	<a href="#">dyn_manager</a>	76
8.8.5.11	<a href="#">dyn_parent</a>	76
8.8.5.12	<a href="#">frame_attach</a>	76
8.8.5.13	<a href="#">grav_interaction</a>	76
8.8.5.14	<a href="#">initialized_states</a>	77
8.8.5.15	<a href="#">integ_frame</a>	77
8.8.5.16	<a href="#">integ_frame_name</a>	77
8.8.5.17	<a href="#">integ_results_merger</a>	77
8.8.5.18	<a href="#">integrated_frame</a>	77
8.8.5.19	<a href="#">mass</a>	78
8.8.5.20	<a href="#">mass_children</a>	78
8.8.5.21	<a href="#">name</a>	78
8.8.5.22	<a href="#">position_source</a>	78
8.8.5.23	<a href="#">rate_source</a>	78
8.8.5.24	<a href="#">rot_integrator</a>	79
8.8.5.25	<a href="#">rotation_integration</a>	79
8.8.5.26	<a href="#">rotational_dynamics</a>	79
8.8.5.27	<a href="#">structure</a>	79
8.8.5.28	<a href="#">three_dof</a>	80
8.8.5.29	<a href="#">time_manager</a>	80

8.8.5.30	<a href="#">trans_integrator</a>	80
8.8.5.31	<a href="#">translational_dynamics</a>	80
8.8.5.32	<a href="#">vehicle_points</a>	80
8.8.5.33	<a href="#">velocity_source</a>	80
8.9	<a href="#">jeod::DynBodyGenericFrameAttachment Class Reference</a>	81
8.9.1	<a href="#">Detailed Description</a>	82
8.9.2	<a href="#">Constructor &amp; Destructor Documentation</a>	82
8.9.2.1	<a href="#">DynBodyGenericFrameAttachment</a>	82
8.9.3	<a href="#">Member Function Documentation</a>	82
8.9.3.1	<a href="#">clear_attachment</a>	82
8.9.3.2	<a href="#">get_attach_offset</a>	82
8.9.3.3	<a href="#">get_parent_frame</a>	82
8.9.3.4	<a href="#">initialize_attachment</a>	82
8.9.3.5	<a href="#">isAttached</a>	83
8.9.4	<a href="#">Friends And Related Function Documentation</a>	83
8.9.4.1	<a href="#">init_attrjeod__DynBodyGenericFrameAttachment</a>	83
8.9.4.2	<a href="#">InputProcessor</a>	83
8.9.5	<a href="#">Field Documentation</a>	83
8.9.5.1	<a href="#">active</a>	83
8.9.5.2	<a href="#">rigid_attach_parent</a>	83
8.9.5.3	<a href="#">rigid_attach_state</a>	83
8.10	<a href="#">jeod::DynBodyMessages Class Reference</a>	83
8.10.1	<a href="#">Detailed Description</a>	84
8.10.2	<a href="#">Constructor &amp; Destructor Documentation</a>	84
8.10.2.1	<a href="#">DynBodyMessages</a>	84
8.10.2.2	<a href="#">DynBodyMessages</a>	84
8.10.3	<a href="#">Member Function Documentation</a>	84
8.10.3.1	<a href="#">operator=</a>	84
8.10.4	<a href="#">Friends And Related Function Documentation</a>	84
8.10.4.1	<a href="#">init_attrjeod__DynBodyMessages</a>	84
8.10.4.2	<a href="#">InputProcessor</a>	84
8.10.5	<a href="#">Field Documentation</a>	85
8.10.5.1	<a href="#">internal_error</a>	85
8.10.5.2	<a href="#">invalid_attachment</a>	85
8.10.5.3	<a href="#">invalid_body</a>	85
8.10.5.4	<a href="#">invalid_frame</a>	85
8.10.5.5	<a href="#">invalid_group</a>	86
8.10.5.6	<a href="#">invalid_name</a>	86
8.10.5.7	<a href="#">invalid_technique</a>	86
8.10.5.8	<a href="#">not_dyn_body</a>	86

8.11	<a href="#">jeod::Force Class Reference</a>	87
8.11.1	<a href="#">Detailed Description</a>	87
8.11.2	<a href="#">Constructor &amp; Destructor Documentation</a>	88
8.11.2.1	<a href="#">Force</a>	88
8.11.2.2	<a href="#">~Force</a>	88
8.11.2.3	<a href="#">Force</a>	88
8.11.3	<a href="#">Member Function Documentation</a>	88
8.11.3.1	<a href="#">operator=</a>	88
8.11.3.2	<a href="#">operator[]</a>	88
8.11.3.3	<a href="#">operator[]</a>	88
8.11.4	<a href="#">Field Documentation</a>	89
8.11.4.1	<a href="#">active</a>	89
8.11.4.2	<a href="#">force</a>	89
8.12	<a href="#">jeod::FrameDerivs Class Reference</a>	89
8.12.1	<a href="#">Detailed Description</a>	90
8.12.2	<a href="#">Constructor &amp; Destructor Documentation</a>	90
8.12.2.1	<a href="#">FrameDerivs</a>	90
8.12.3	<a href="#">Field Documentation</a>	90
8.12.3.1	<a href="#">non_grav_accel</a>	90
8.12.3.2	<a href="#">Qdot_parent_this</a>	90
8.12.3.3	<a href="#">rot_accel</a>	90
8.12.3.4	<a href="#">trans_accel</a>	90
8.13	<a href="#">jeod::JPVCollectForce Class Reference</a>	91
8.13.1	<a href="#">Detailed Description</a>	91
8.13.2	<a href="#">Member Function Documentation</a>	91
8.13.2.1	<a href="#">perform_insert_action</a>	91
8.13.2.2	<a href="#">push_back</a>	92
8.13.3	<a href="#">Friends And Related Function Documentation</a>	92
8.13.3.1	<a href="#">collect_insert</a>	92
8.13.3.2	<a href="#">collect_push_back</a>	92
8.14	<a href="#">jeod::JPVCollectTorque Class Reference</a>	92
8.14.1	<a href="#">Detailed Description</a>	93
8.14.2	<a href="#">Member Function Documentation</a>	93
8.14.2.1	<a href="#">perform_insert_action</a>	93
8.14.2.2	<a href="#">push_back</a>	93
8.14.3	<a href="#">Friends And Related Function Documentation</a>	93
8.14.3.1	<a href="#">collect_insert</a>	93
8.14.3.2	<a href="#">collect_push_back</a>	93
8.15	<a href="#">jeod::StructureIntegratedDynBody Class Reference</a>	94
8.15.1	<a href="#">Detailed Description</a>	96

8.15.2	Constructor & Destructor Documentation	96
8.15.2.1	StructureIntegratedDynBody	96
8.15.2.2	~StructureIntegratedDynBody	96
8.15.2.3	StructureIntegratedDynBody	96
8.15.3	Member Function Documentation	96
8.15.3.1	add_constraint	96
8.15.3.2	attach_update_properties	96
8.15.3.3	collect_forces_and_torques	97
8.15.3.4	collect_local_forces_and_torques	97
8.15.3.5	complete_translational_acceleration	97
8.15.3.6	compute_inertial_torque	98
8.15.3.7	compute_rotational_acceleration	98
8.15.3.8	compute_translational_acceleration	98
8.15.3.9	compute_vehicle_point_derivatives	98
8.15.3.10	detach	99
8.15.3.11	get_vehicle_properties	100
8.15.3.12	operator=	100
8.15.3.13	PropagateForcesAndTorques	100
8.15.3.14	rot_integ	100
8.15.3.15	set_solver	101
8.15.3.16	solve_constraints	101
8.15.3.17	trans_integ	101
8.15.4	Friends And Related Function Documentation	101
8.15.4.1	DynBodyConstraintsSolver	101
8.15.4.2	init_attrjeod__StructureIntegratedDynBody	101
8.15.4.3	InputProcessor	101
8.15.5	Field Documentation	102
8.15.5.1	constraints_solver	102
8.15.5.2	effector_wrench	102
8.15.5.3	effector_wrench_collection	102
8.15.5.4	inertial_accel_inrtl	102
8.15.5.5	inertial_accel_struct	102
8.15.5.6	inertial_accel_struct_omega	103
8.15.5.7	inertial_accel_struct_omega_dot	103
8.15.5.8	non_grav_state	103
8.15.5.9	struct_derivs	103
8.15.5.10	vehicle_properties	103
8.16	jeod::Torque Class Reference	104
8.16.1	Detailed Description	104
8.16.2	Constructor & Destructor Documentation	104

8.16.2.1	Torque	104
8.16.2.2	~Torque	105
8.16.2.3	Torque	105
8.16.3	Member Function Documentation	105
8.16.3.1	operator=	105
8.16.3.2	operator[]	105
8.16.3.3	operator[]	105
8.16.4	Field Documentation	105
8.16.4.1	active	105
8.16.4.2	torque	106
8.17	jeod::VehicleNonGravState Class Reference	106
8.17.1	Detailed Description	106
8.17.2	Friends And Related Function Documentation	106
8.17.2.1	init_attrjeod__VehicleNonGravState	106
8.17.2.2	InputProcessor	106
8.17.3	Field Documentation	107
8.17.3.1	accel_struct	107
8.17.3.2	inertial_torque_struct	107
8.17.3.3	omega_body	107
8.17.3.4	omega_dot_body	107
8.17.3.5	omega_dot_struct	107
8.17.3.6	omega_struct	107
8.18	jeod::VehicleProperties Class Reference	108
8.18.1	Detailed Description	109
8.18.2	Constructor & Destructor Documentation	109
8.18.2.1	VehicleProperties	109
8.18.2.2	VehicleProperties	109
8.18.3	Member Function Documentation	109
8.18.3.1	get_inertia	109
8.18.3.2	get_inverse_inertia	110
8.18.3.3	get_inverse_mass	110
8.18.3.4	get_mass	110
8.18.3.5	get_parent_to_structure_offset	110
8.18.3.6	get_parent_to_structure_transform	110
8.18.3.7	get_structure_to_body_offset	110
8.18.3.8	get_structure_to_body_transform	111
8.18.4	Friends And Related Function Documentation	111
8.18.4.1	init_attrjeod__VehicleProperties	111
8.18.4.2	InputProcessor	111
8.18.5	Field Documentation	111

8.18.5.1	<a href="#">inertia</a>	111
8.18.5.2	<a href="#">inverse_inertia</a>	111
8.18.5.3	<a href="#">inverse_mass</a>	111
8.18.5.4	<a href="#">mass</a>	112
8.18.5.5	<a href="#">parent_to_structure_offset</a>	112
8.18.5.6	<a href="#">parent_to_structure_transform</a>	112
8.18.5.7	<a href="#">structure_to_body_offset</a>	112
8.18.5.8	<a href="#">structure_to_body_transform</a>	112
8.19	<a href="#">jeod::Wrench Class Reference</a>	112
8.19.1	<a href="#">Detailed Description</a>	114
8.19.2	<a href="#">Constructor &amp; Destructor Documentation</a>	114
8.19.2.1	<a href="#">Wrench</a>	114
8.19.2.2	<a href="#">Wrench</a>	115
8.19.2.3	<a href="#">Wrench</a>	115
8.19.2.4	<a href="#">~Wrench</a>	115
8.19.2.5	<a href="#">Wrench</a>	115
8.19.2.6	<a href="#">Wrench</a>	115
8.19.3	<a href="#">Member Function Documentation</a>	115
8.19.3.1	<a href="#">accumulate</a>	115
8.19.3.2	<a href="#">accumulate</a>	116
8.19.3.3	<a href="#">activate</a>	116
8.19.3.4	<a href="#">deactivate</a>	116
8.19.3.5	<a href="#">get_force</a>	116
8.19.3.6	<a href="#">get_point</a>	116
8.19.3.7	<a href="#">get_torque</a>	116
8.19.3.8	<a href="#">is_active</a>	117
8.19.3.9	<a href="#">operator+=</a>	117
8.19.3.10	<a href="#">operator=</a>	117
8.19.3.11	<a href="#">operator=</a>	117
8.19.3.12	<a href="#">reset_force</a>	117
8.19.3.13	<a href="#">reset_force_and_torque</a>	117
8.19.3.14	<a href="#">reset_point</a>	118
8.19.3.15	<a href="#">reset_torque</a>	118
8.19.3.16	<a href="#">scale_force</a>	118
8.19.3.17	<a href="#">scale_torque</a>	118
8.19.3.18	<a href="#">set</a>	118
8.19.3.19	<a href="#">set_force</a>	118
8.19.3.20	<a href="#">set_force</a>	119
8.19.3.21	<a href="#">set_point</a>	119
8.19.3.22	<a href="#">set_torque</a>	119

8.19.3.23 transform_to_parent . . . . .	119
8.19.3.24 transform_to_point . . . . .	119
8.19.4 Friends And Related Function Documentation . . . . .	120
8.19.4.1 init_attrjeod__Wrench . . . . .	120
8.19.4.2 InputProcessor . . . . .	120
8.19.5 Field Documentation . . . . .	120
8.19.5.1 active . . . . .	120
8.19.5.2 force . . . . .	120
8.19.5.3 point . . . . .	120
8.19.5.4 torque . . . . .	120
<b>9 File Documentation</b> . . . . .	<b>123</b>
9.1 aux_classes.cc File Reference . . . . .	123
9.1.1 Detailed Description . . . . .	123
9.2 body_force_collect.hh File Reference . . . . .	123
9.2.1 Detailed Description . . . . .	124
9.3 body_ref_frame.hh File Reference . . . . .	124
9.3.1 Detailed Description . . . . .	124
9.4 body_wrench_collect.cc File Reference . . . . .	125
9.4.1 Detailed Description . . . . .	125
9.5 body_wrench_collect.hh File Reference . . . . .	125
9.5.1 Detailed Description . . . . .	125
9.6 class_declarations.hh File Reference . . . . .	125
9.6.1 Detailed Description . . . . .	126
9.7 dyn_body.cc File Reference . . . . .	126
9.7.1 Detailed Description . . . . .	126
9.8 dyn_body.hh File Reference . . . . .	126
9.8.1 Detailed Description . . . . .	127
9.9 dyn_body_attach.cc File Reference . . . . .	127
9.9.1 Detailed Description . . . . .	127
9.10 dyn_body_collect.cc File Reference . . . . .	127
9.10.1 Detailed Description . . . . .	128
9.11 dyn_body_detach.cc File Reference . . . . .	128
9.11.1 Detailed Description . . . . .	128
9.12 dyn_body_find_body_frame.cc File Reference . . . . .	129
9.12.1 Detailed Description . . . . .	129
9.13 dyn_body_generic_rigid_attach.hh File Reference . . . . .	129
9.13.1 Detailed Description . . . . .	129
9.14 dyn_body_initialize_model.cc File Reference . . . . .	129
9.14.1 Detailed Description . . . . .	130

9.15	<a href="#">dyn_body_integration.cc File Reference</a>	130
9.15.1	<a href="#">Detailed Description</a>	130
9.16	<a href="#">dyn_body_messages.cc File Reference</a>	131
9.16.1	<a href="#">Detailed Description</a>	131
9.17	<a href="#">dyn_body_messages.hh File Reference</a>	131
9.17.1	<a href="#">Detailed Description</a>	131
9.18	<a href="#">dyn_body_propagate_state.cc File Reference</a>	131
9.18.1	<a href="#">Detailed Description</a>	132
9.19	<a href="#">dyn_body_set_state.cc File Reference</a>	132
9.19.1	<a href="#">Detailed Description</a>	132
9.20	<a href="#">dyn_body_vehicle_point.cc File Reference</a>	133
9.20.1	<a href="#">Detailed Description</a>	133
9.21	<a href="#">force.cc File Reference</a>	133
9.21.1	<a href="#">Detailed Description</a>	133
9.22	<a href="#">force.hh File Reference</a>	133
9.22.1	<a href="#">Detailed Description</a>	134
9.23	<a href="#">force_inline.hh File Reference</a>	134
9.23.1	<a href="#">Detailed Description</a>	134
9.24	<a href="#">frame_derivs.hh File Reference</a>	134
9.24.1	<a href="#">Detailed Description</a>	135
9.25	<a href="#">structure_integrated_dyn_body.cc File Reference</a>	135
9.25.1	<a href="#">Detailed Description</a>	135
9.26	<a href="#">structure_integrated_dyn_body.hh File Reference</a>	135
9.26.1	<a href="#">Detailed Description</a>	136
9.27	<a href="#">structure_integrated_dyn_body_collect.cc File Reference</a>	136
9.27.1	<a href="#">Detailed Description</a>	136
9.28	<a href="#">structure_integrated_dyn_body_integration.cc File Reference</a>	136
9.28.1	<a href="#">Detailed Description</a>	137
9.29	<a href="#">structure_integrated_dyn_body_pt_accel.cc File Reference</a>	137
9.29.1	<a href="#">Detailed Description</a>	137
9.30	<a href="#">structure_integrated_dyn_body_solve.cc File Reference</a>	137
9.30.1	<a href="#">Detailed Description</a>	138
9.31	<a href="#">torque.cc File Reference</a>	138
9.31.1	<a href="#">Detailed Description</a>	138
9.32	<a href="#">torque.hh File Reference</a>	138
9.32.1	<a href="#">Detailed Description</a>	139
9.33	<a href="#">torque_inline.hh File Reference</a>	139
9.33.1	<a href="#">Detailed Description</a>	139
9.34	<a href="#">vehicle_non_grav_state.hh File Reference</a>	139
9.34.1	<a href="#">Detailed Description</a>	139



---

9.35 vehicle_properties.hh File Reference . . . . .	140
9.35.1 Detailed Description . . . . .	140
9.36 wrench.hh File Reference . . . . .	140
9.36.1 Detailed Description . . . . .	140
 <b>Index</b>	 <b>141</b>



# Chapter 1

## Module Index

### 1.1 Modules

Here is a list of all modules:

Models . . . . .	<a href="#">11</a>
Dynamics . . . . .	<a href="#">12</a>
DynBody . . . . .	<a href="#">13</a>



## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">jeod</a>	Namespace jeod . . . . .	15
----------------------	--------------------------	----



## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

jeod::BodyForceCollect . . . . .	21
jeod::BodyWrenchCollect . . . . .	28
jeod::CollectForce . . . . .	33
jeod::CInterfaceForce . . . . .	30
jeod::CollectTorque . . . . .	38
jeod::CInterfaceTorque . . . . .	31
jeod::DynBodyGenericFrameAttachment . . . . .	81
jeod::DynBodyMessages . . . . .	83
jeod::Force . . . . .	87
jeod::FrameDerivs . . . . .	89
IntegrableObject	
jeod::DynBody . . . . .	43
jeod::StructureIntegratedDynBody . . . . .	94
RefFrame	
jeod::BodyRefFrame . . . . .	26
RefFrameOwner	
jeod::DynBody . . . . .	43
jeod::Torque . . . . .	104
type	
jeod::JPVCollectForce . . . . .	91
jeod::JPVCollectTorque . . . . .	92
jeod::VehicleNonGravState . . . . .	106
jeod::VehicleProperties . . . . .	108
jeod::Wrench . . . . .	112





## Chapter 4

# Data Structure Index

### 4.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">jeod::BodyForceCollect</a>	Serves as the collection point for forces and torques that act on a vehicle . . . . .	21
<a href="#">jeod::BodyRefFrame</a>	Extend RefFrame to add coupling between the reference frame tree and the mass tree and to keep track of which state items have been set . . . . .	26
<a href="#">jeod::BodyWrenchCollect</a>	Serves as the collection point for wrenches that act on a vehicle . . . . .	28
<a href="#">jeod::CInterfaceForce</a>	This class is deprecated . . . . .	30
<a href="#">jeod::CInterfaceTorque</a>	This class is deprecated . . . . .	31
<a href="#">jeod::CollectForce</a>	A <a href="#">CollectForce</a> represents a collected force that acts on a vehicle . . . . .	33
<a href="#">jeod::CollectTorque</a>	A <a href="#">CollectTorque</a> represents a collected torque that acts on a vehicle . . . . .	38
<a href="#">jeod::DynBody</a>	Class <a href="#">DynBody</a> is the base class for all dynamic bodies . . . . .	43
<a href="#">jeod::DynBodyGenericFrameAttachment</a>	A wrench comprises a torque and a force applied at a point on a <a href="#">DynBody</a> . . . . .	81
<a href="#">jeod::DynBodyMessages</a>	Specify the message IDs used in the <a href="#">DynBody</a> model . . . . .	83
<a href="#">jeod::Force</a>	A <a href="#">Force</a> represents a Newtonian force that acts on a <a href="#">DynBody</a> . . . . .	87
<a href="#">jeod::FrameDerivs</a>	Contains translational and rotational second derivatives . . . . .	89
<a href="#">jeod::JPVCollectForce</a>	This is a derived version of the template class <code>JeodPointerVector&lt;CollectForce&gt;::type</code> with an implementation of the method <code>perform_cleanup_action</code> which frees and clears stale data following a restore . . . . .	91
<a href="#">jeod::JPVCollectTorque</a>	This is a derived version of the template class <code>JeodPointerVector&lt;CollectTorque&gt;::type</code> with an implementation of the method <code>perform_cleanup_action</code> which frees and clears stale data following a restore . . . . .	92
<a href="#">jeod::StructureIntegratedDynBody</a>	Extends <a href="#">DynBody</a> to integrate an object's structural reference frame as opposed to its center of mass . . . . .	94
<a href="#">jeod::Torque</a>	A <a href="#">Torque</a> represents a Newtonian torque that acts on a <a href="#">DynBody</a> . . . . .	104

<a href="#">jeod::VehicleNonGravState</a>	
Encapsulates various aspects of a vehicle's state with respect to inertial	106
<a href="#">jeod::VehicleProperties</a>	
Captures pointers to various vehicle properties that are commonly used in the constraint concept	108
<a href="#">jeod::Wrench</a>	
A wrench comprises a torque and a force applied at a point on a <a href="#">DynBody</a>	112

## Chapter 5

# File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

<a href="#">aux_classes.cc</a>	Define base methods for various small JEOD DynBody classes . . . . .	123
<a href="#">body_force_collect.hh</a>	Define the class BodyForceCollect . . . . .	123
<a href="#">body_ref_frame.hh</a>	Define the class BodyRefFrame . . . . .	124
<a href="#">body_wrench_collect.cc</a>	Define BodyWrenchCollect member functions . . . . .	125
<a href="#">body_wrench_collect.hh</a>	Defines the class BodyWrenchCollect . . . . .	125
<a href="#">class_declarations.hh</a>	Forward declarations of classes defined in <a href="#">dyn_body.hh</a> . . . . .	125
<a href="#">dyn_body.cc</a>	Define base methods for the DynBody class . . . . .	126
<a href="#">dyn_body.hh</a>	Define the class DynBody . . . . .	126
<a href="#">dyn_body_attach.cc</a>	Define DynBody attachment methods . . . . .	127
<a href="#">dyn_body_collect.cc</a>	Define DynBody methods related to force and torque accumulation and propagation . . . . .	127
<a href="#">dyn_body_detach.cc</a>	Define DynBody detachment methods . . . . .	128
<a href="#">dyn_body_find_body_frame.cc</a>	Define DynBody::find_body_frame . . . . .	129
<a href="#">dyn_body_generic_rigid_attach.hh</a>	Define the class Wrench . . . . .	129
<a href="#">dyn_body_initialize_model.cc</a>	Define DynBody::initialize_model . . . . .	129
<a href="#">dyn_body_integration.cc</a>	Define methods for frame switching . . . . .	130
<a href="#">dyn_body_messages.cc</a>	Implement the class De4xxMessages . . . . .	131
<a href="#">dyn_body_messages.hh</a>	Define the class DynBodyMessages . . . . .	131
<a href="#">dyn_body_propagate_state.cc</a>	Define DynBody state propagation / update methods . . . . .	131
<a href="#">dyn_body_set_state.cc</a>	Define methods related to setting aspects of a vehicle's state . . . . .	132

<a href="#">dyn_body_vehicle_point.cc</a>	Define methods that support vehicle points . . . . .	133
<a href="#">force.cc</a>	Define force model member functions . . . . .	133
<a href="#">force.hh</a>	Define the JEOD force model . . . . .	133
<a href="#">force_inline.hh</a>	Inline functions for the JEOD force model . . . . .	134
<a href="#">frame_derivs.hh</a>	Define the FrameDerivs class . . . . .	134
<a href="#">structure_integrated_dyn_body.cc</a>	Define base member functions for StructureIntegratedDynBody . . . . .	135
<a href="#">structure_integrated_dyn_body.hh</a>	Define the class StructureIntegratedDynBody, which integrates a DynBody object's structural state . . . . .	135
<a href="#">structure_integrated_dyn_body_collect.cc</a>	Define StructureIntegratedDynBody methods related to force and torque accumulation and propagation . . . . .	136
<a href="#">structure_integrated_dyn_body_integration.cc</a>	Define StructureIntegratedDynBody member functions related to state integration . . . . .	136
<a href="#">structure_integrated_dyn_body_pt_accel.cc</a>	Define StructureIntegratedDynBody::compute_vehicle_point_derivatives . . . . .	137
<a href="#">structure_integrated_dyn_body_solve.cc</a>	Define StructureIntegratedDynBody methods related to force and torque accumulation and propagation . . . . .	137
<a href="#">torque.cc</a>	Define torque model member functions . . . . .	138
<a href="#">torque.hh</a>	Define the JEOD torque model . . . . .	138
<a href="#">torque_inline.hh</a>	Define the JEOD torque model . . . . .	139
<a href="#">vehicle_non_grav_state.hh</a>	Define the class VehicleNonGravState . . . . .	139
<a href="#">vehicle_properties.hh</a>	Define the class VehicleProperties . . . . .	140
<a href="#">wrench.hh</a>	Define the class Wrench . . . . .	140

## Chapter 6

# Module Documentation

### 6.1 Models

#### Modules

- [Dynamics](#)

#### 6.1.1 Detailed Description

## 6.2 Dynamics

### Modules

- [DynBody](#)

### 6.2.1 Detailed Description

## 6.3 DynBody

### Files

- file [body\\_force\\_collect.hh](#)  
*Define the class BodyForceCollect.*
- file [body\\_ref\\_frame.hh](#)  
*Define the class BodyRefFrame.*
- file [body\\_wrench\\_collect.hh](#)  
*Defines the class BodyWrenchCollect.*
- file [class\\_declarations.hh](#)  
*Forward declarations of classes defined in [dyn\\_body.hh](#).*
- file [dyn\\_body.hh](#)  
*Define the class DynBody.*
- file [dyn\\_body\\_generic\\_rigid\\_attach.hh](#)  
*Define the class Wrench.*
- file [dyn\\_body\\_messages.hh](#)  
*Define the class DynBodyMessages.*
- file [force.hh](#)  
*Define the JEOD force model.*
- file [force\\_inline.hh](#)  
*Inline functions for the JEOD force model.*
- file [frame\\_derivs.hh](#)  
*Define the FrameDerivs class.*
- file [structure\\_integrated\\_dyn\\_body.hh](#)  
*Define the class StructureIntegratedDynBody, which integrates a DynBody object's structural state.*
- file [torque.hh](#)  
*Define the JEOD torque model.*
- file [torque\\_inline.hh](#)  
*Define the JEOD torque model.*
- file [vehicle\\_non\\_grav\\_state.hh](#)  
*Define the class VehicleNonGravState.*
- file [vehicle\\_properties.hh](#)  
*Define the class VehicleProperties.*
- file [wrench.hh](#)  
*Define the class Wrench.*
- file [aux\\_classes.cc](#)  
*Define base methods for various small JEOD DynBody classes.*
- file [body\\_wrench\\_collect.cc](#)  
*Define BodyWrenchCollect member functions.*
- file [dyn\\_body.cc](#)  
*Define base methods for the DynBody class.*
- file [dyn\\_body\\_attach.cc](#)  
*Define DynBody attachment methods.*
- file [dyn\\_body\\_collect.cc](#)  
*Define DynBody methods related to force and torque accumulation and propagation.*
- file [dyn\\_body\\_detach.cc](#)  
*Define DynBody detachment methods.*
- file [dyn\\_body\\_find\\_body\\_frame.cc](#)  
*Define DynBody::find\_body\_frame.*
- file [dyn\\_body\\_initialize\\_model.cc](#)

- Define `DynBody::initialize_model`.
- file [dyn\\_body\\_integration.cc](#)
  - Define methods for frame switching.
- file [dyn\\_body\\_messages.cc](#)
  - Implement the class `De4xxMessages`.
- file [dyn\\_body\\_propagate\\_state.cc](#)
  - Define `DynBody` state propagation / update methods.
- file [dyn\\_body\\_set\\_state.cc](#)
  - Define methods related to setting aspects of a vehicle's state.
- file [dyn\\_body\\_vehicle\\_point.cc](#)
  - Define methods that support vehicle points.
- file [force.cc](#)
  - Define force model member functions.
- file [structure\\_integrated\\_dyn\\_body.cc](#)
  - Define base member functions for `StructureIntegratedDynBody`.
- file [structure\\_integrated\\_dyn\\_body\\_collect.cc](#)
  - Define `StructureIntegratedDynBody` methods related to force and torque accumulation and propagation.
- file [structure\\_integrated\\_dyn\\_body\\_integration.cc](#)
  - Define `StructureIntegratedDynBody` member functions related to state integration.
- file [structure\\_integrated\\_dyn\\_body\\_pt\\_accel.cc](#)
  - Define `StructureIntegratedDynBody::compute_vehicle_point_derivatives`.
- file [structure\\_integrated\\_dyn\\_body\\_solve.cc](#)
  - Define `StructureIntegratedDynBody` methods related to force and torque accumulation and propagation.
- file [torque.cc](#)
  - Define torque model member functions.

## Namespaces

- [jeod](#)
  - Namespace `jeod`.

## Macros

- `#define` [PATH](#) "dynamics/dyn\_body/"

### 6.3.1 Detailed Description

### 6.3.2 Macro Definition Documentation

#### 6.3.2.1 `#define` [PATH](#) "dynamics/dyn\_body/"

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



## Chapter 7

# Namespace Documentation

### 7.1 jeod Namespace Reference

Namespace jeod.

#### Data Structures

- class [JPVCollectForce](#)  
*This is a derived version of the template class `JeodPointerVector<CollectForce>::type` with an implementation of the method `perform_cleanup_action` which frees and clears stale data following a restore.*
- class [JPVCollectTorque](#)  
*This is a derived version of the template class `JeodPointerVector<CollectTorque>::type` with an implementation of the method `perform_cleanup_action` which frees and clears stale data following a restore.*
- class [BodyForceCollect](#)  
*Serves as the collection point for forces and torques that act on a vehicle.*
- class [BodyRefFrame](#)  
*Extend `RefFrame` to add coupling between the reference frame tree and the mass tree and to keep track of which state items have been set.*
- class [BodyWrenchCollect](#)  
*Serves as the collection point for wrenches that act on a vehicle.*
- class [DynBody](#)  
*Class `DynBody` is the base class for all dynamic bodies.*
- class [DynBodyGenericFrameAttachment](#)  
*A wrench comprises a torque and a force applied at a point on a `DynBody`.*
- class [DynBodyMessages](#)  
*Specify the message IDs used in the `DynBody` model.*
- class [Force](#)  
*A `Force` represents a Newtonian force that acts on a `DynBody`.*
- class [CollectForce](#)  
*A `CollectForce` represents a collected force that acts on a vehicle.*
- class [CInterfaceForce](#)  
*This class is deprecated.*
- class [FrameDerivs](#)  
*Contains translational and rotational second derivatives.*
- class [StructureIntegratedDynBody](#)  
*Extends `DynBody` to integrate an object's structural reference frame as opposed to its center of mass.*
- class [Torque](#)

A [Torque](#) represents a Newtonian torque that acts on a [DynBody](#).

- class [CollectTorque](#)

A [CollectTorque](#) represents a collected torque that acts on a vehicle.

- class [CInterfaceTorque](#)

This class is deprecated.

- class [VehicleNonGravState](#)

Encapsulates various aspects of a vehicle's state with respect to inertial.

- class [VehicleProperties](#)

Captures pointers to various vehicle properties that are commonly used in the constraint concept.

- class [Wrench](#)

A wrench comprises a torque and a force applied at a point on a [DynBody](#).

## Functions

- template<class CollectType >  
void [release\\_vector](#) (CollectType &vec)  
*Release JEOD-allocated memory in the collect vector.*
- template<typename CollectType , typename value\_type >  
void [collect\\_insert](#) (CollectType &collect\_in, value\_type &elem)
- template<typename CollectType , typename value\_type >  
void [collect\\_push\\_back](#) (CollectType &collect\_in, value\_type &elem)
- static void [accumulate\\_forces](#) (const JeodPointerVector< [CollectForce](#) >::type &vec, double \*cumulation)  
*Accumulate forces acting on a vehicle.*
- static void [accumulate\\_torques](#) (const JeodPointerVector< [CollectTorque](#) >::type &vec, double \*cumulation)  
*Accumulate torques acting on a vehicle.*
- static void [check\\_frame\\_ownership](#) (const [BodyRefFrame](#) &frame, const [DynBody](#) \*dyn\_body, const char \*file, unsigned int line)  
*Check that the dyn\_body 'owns' the subject frame.*
- static void [accumulate\\_forces](#) (const JeodPointerVector< [CollectForce](#) >::type &vec, double \*cumulation)  
*Accumulate forces acting on a vehicle.*
- static void [accumulate\\_torques](#) (const JeodPointerVector< [CollectTorque](#) >::type &vec, double \*cumulation)  
*Accumulate torques acting on a vehicle.*

### 7.1.1 Detailed Description

Namespace jeod.

### 7.1.2 Function Documentation

- 7.1.2.1 static void jeod::accumulate\_forces ( const JeodPointerVector< [CollectForce](#) >::type & vec, double \* cumulation )  
[inline], [static]

Accumulate forces acting on a vehicle.

Parameters

in	vec	Forces
out	cumulation	Accumulated force

Definition at line 40 of file structure\_integrated\_dyn\_body\_collect.cc.

7.1.2.2 `static void jeod::accumulate_forces ( const JeodPointerVector< CollectForce >::type & vec, double * cumulation )`  
`[inline], [static]`

Accumulate forces acting on a vehicle.

## Parameters

in	<i>vec</i>	Forces
out	<i>cumulation</i>	Accumulated force

Definition at line 59 of file dyn\_body\_collect.cc.

Referenced by jeod::DynBody::collect\_forces\_and\_torques(), and jeod::StructureIntegratedDynBody::collect\_local\_forces\_and\_torques().

**7.1.2.3** static void jeod::accumulate\_torques ( const JeodPointerVector< CollectTorque >::type & *vec*, double \* *cumulation* )  
[inline], [static]

Accumulate torques acting on a vehicle.

## Parameters

in	<i>vec</i>	Torques
out	<i>cumulation</i>	Accumulated torque

Definition at line 61 of file structure\_integrated\_dyn\_body\_collect.cc.

**7.1.2.4** static void jeod::accumulate\_torques ( const JeodPointerVector< CollectTorque >::type & *vec*, double \* *cumulation* )  
[inline], [static]

Accumulate torques acting on a vehicle.

## Parameters

in	<i>vec</i>	Torques
out	<i>cumulation</i>	Accumulated torque

Definition at line 81 of file dyn\_body\_collect.cc.

Referenced by jeod::DynBody::collect\_forces\_and\_torques(), and jeod::StructureIntegratedDynBody::collect\_local\_forces\_and\_torques().

**7.1.2.5** static void jeod::check\_frame\_ownership ( const BodyRefFrame & *frame*, const DynBody \* *dyn\_body*, const char \* *file*, unsigned int *line* ) [inline], [static]

Check that the dyn\_body 'owns' the subject frame.

## Parameters

in	<i>frame</i>	Frame to test
in	<i>dyn_body</i>	Typically this
in	<i>file</i>	Typically <b>FILE</b>
in	<i>line</i>	Typically <b>LINE</b>

Definition at line 62 of file dyn\_body\_set\_state.cc.

References jeod::DynBodyMessages::invalid\_frame, and jeod::DynBody::name.

Referenced by jeod::DynBody::set\_attitude\_left\_quaternion(), jeod::DynBody::set\_attitude\_matrix(), jeod::DynBody::set\_attitude\_rate(), jeod::DynBody::set\_attitude\_right\_quaternion(), jeod::DynBody::set\_position(), jeod::DynBody::set\_state(), and jeod::DynBody::set\_velocity().

**7.1.2.6** template<typename CollectType , typename value\_type > void jeod::collect\_insert ( CollectType & *collect\_in*, value\_type & *elem* )

Definition at line 99 of file body\_force\_collect.hh.

7.1.2.7 `template<typename CollectType , typename value_type > void jeod::collect_push_back ( CollectType & collect_in, value_type & elem )`

Definition at line 131 of file `body_force_collect.hh`.

7.1.2.8 `template<class CollectType > void jeod::release_vector ( CollectType & vec )`

Release JEOD-allocated memory in the collect vector.

Parameters

<code><i>in, out</i></code>	<code><i>vec</i></code>	Collected vectors
-----------------------------	-------------------------	-------------------

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

Referenced by `jeod::BodyForceCollect::~~BodyForceCollect()`.



## Chapter 8

# Data Structure Documentation

### 8.1 jeod::BodyForceCollect Class Reference

Serves as the collection point for forces and torques that act on a vehicle.

```
#include <body_force_collect.hh>
```

#### Public Member Functions

- [BodyForceCollect](#) ()  
*Default constructor.*
- [~BodyForceCollect](#) ()  
*Destructor.*

#### Data Fields

- double [effector\\_forc](#) [3]  
*Sum of effector forces, struct ref.*
- double [environ\\_forc](#) [3]  
*Sum of env forces, struct ref.*
- double [no\\_xmit\\_forc](#) [3]  
*Sum of local forces, struct ref.*
- double [extern\\_forc\\_struct](#) [3]  
*Sum of external forces, struct ref.*
- double [extern\\_forc\\_inrtl](#) [3]  
*Sum of external forces, inertial.*
- double [effector\\_torq](#) [3]  
*Sum of effector torques about body CoM, struct ref.*
- double [environ\\_torq](#) [3]  
*Sum of environment torqs about body CoM, struct ref.*
- double [no\\_xmit\\_torq](#) [3]  
*Sum of torqs not transmitted to a parent about body CoM, struct ref.*
- double [inertial\\_torq](#) [3]  
*Induced inertial torques from second order rotational dynamics, w x lw, body ref.*
- double [extern\\_torq\\_struct](#) [3]  
*Sum of external torques, struct ref.*
- double [extern\\_torq\\_body](#) [3]

- *Sum of external torques, body ref.*
- [JPVCollectForce collect\\_effector\\_forc](#)  
*Vector of effector forces, (struct)*
- [JPVCollectForce collect\\_environ\\_forc](#)  
*Vector of env forces, (struct)*
- [JPVCollectForce collect\\_no\\_xmit\\_forc](#)  
*Vector of local forces, (struct)*
- [JPVCollectTorque collect\\_effector\\_torq](#)  
*Vector of effector torques, (struct)*
- [JPVCollectTorque collect\\_environ\\_torq](#)  
*Vector of env torques, (struct)*
- [JPVCollectTorque collect\\_no\\_xmit\\_torq](#)  
*Vector of local torques, (struct)*

## Private Member Functions

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

### 8.1.1 Detailed Description

Serves as the collection point for forces and torques that act on a vehicle.

This class is a simple class that is tightly coupled with the [DynBody](#) class. The [DynBody](#) class contains (has-a) a [BodyForceCollect](#) member.

The Trick vcollect mechanism (or a similar mechanism in a non-Trick sim) pushes the individual forces and torques onto the various collect\_XXX members of a [BodyForceCollect](#). [DynBody](#) members cumulate these collected forces and torques to form the total forces and torques acting on the vehicle.

Definition at line 258 of file `body_force_collect.hh`.

### 8.1.2 Constructor & Destructor Documentation

8.1.2.1 `jeod::BodyForceCollect::BodyForceCollect ( BodyForceCollect & )` `[private]`

8.1.2.2 `jeod::BodyForceCollect::BodyForceCollect ( void )`

Default constructor.

Definition at line 43 of file `aux_classes.cc`.

References `collect_effector_forc`, `collect_effector_torq`, `collect_environ_forc`, `collect_environ_torq`, `collect_no_xmit_forc`, `collect_no_xmit_torq`, `effector_forc`, `effector_torq`, `environ_forc`, `environ_torq`, `extern_forc_inrtl`, `extern_forc_struct`, `extern_torq_body`, `extern_torq_struct`, `inertial_torq`, `no_xmit_forc`, and `no_xmit_torq`.

8.1.2.3 `jeod::BodyForceCollect::~~BodyForceCollect ( void )`

Destructor.

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

References `collect_effector_forc`, `collect_effector_torq`, `collect_environ_forc`, `collect_environ_torq`, `collect_no_xmit_forc`, `collect_no_xmit_torq`, and `jeod::release_vector()`.



### 8.1.3 Member Function Documentation

8.1.3.1 **BodyForceCollect& jeod::BodyForceCollect::operator= ( const BodyForceCollect & )** [private]

### 8.1.4 Field Documentation

8.1.4.1 **JPVCollectForce jeod::BodyForceCollect::collect\_effector\_forc**

Vector of effector forces, (struct)

trick\_io(\*\*)

Definition at line 337 of file body\_force\_collect.hh.

Referenced by BodyForceCollect(), jeod::DynBody::collect\_forces\_and\_torques(), jeod::StructureIntegratedDynBody::collect\_local\_forces\_and\_torques(), and ~BodyForceCollect().

8.1.4.2 **JPVCollectTorque jeod::BodyForceCollect::collect\_effector\_torq**

Vector of effector torques, (struct)

trick\_io(\*\*)

Definition at line 352 of file body\_force\_collect.hh.

Referenced by BodyForceCollect(), jeod::DynBody::collect\_forces\_and\_torques(), jeod::StructureIntegratedDynBody::collect\_local\_forces\_and\_torques(), and ~BodyForceCollect().

8.1.4.3 **JPVCollectForce jeod::BodyForceCollect::collect\_envirion\_forc**

Vector of env forces, (struct)

trick\_io(\*\*)

Definition at line 342 of file body\_force\_collect.hh.

Referenced by BodyForceCollect(), jeod::DynBody::collect\_forces\_and\_torques(), jeod::StructureIntegratedDynBody::collect\_local\_forces\_and\_torques(), and ~BodyForceCollect().

8.1.4.4 **JPVCollectTorque jeod::BodyForceCollect::collect\_envirion\_torq**

Vector of env torques, (struct)

trick\_io(\*\*)

Definition at line 357 of file body\_force\_collect.hh.

Referenced by BodyForceCollect(), jeod::DynBody::collect\_forces\_and\_torques(), jeod::StructureIntegratedDynBody::collect\_local\_forces\_and\_torques(), and ~BodyForceCollect().

8.1.4.5 **JPVCollectForce jeod::BodyForceCollect::collect\_no\_xmit\_forc**

Vector of local forces, (struct)

trick\_io(\*\*)

Definition at line 347 of file body\_force\_collect.hh.

Referenced by BodyForceCollect(), jeod::DynBody::collect\_forces\_and\_torques(), jeod::StructureIntegratedDynBody::collect\_local\_forces\_and\_torques(), and ~BodyForceCollect().

**8.1.4.6 JPVCollectTorque jeod::BodyForceCollect::collect\_no\_xmit\_torq**

Vector of local torques, (struct)

trick\_io(\*\*)

Definition at line 362 of file body\_force\_collect.hh.

Referenced by BodyForceCollect(), jeod::DynBody::collect\_forces\_and\_torques(), jeod::StructureIntegratedDynBody::collect\_local\_forces\_and\_torques(), and ~BodyForceCollect().

**8.1.4.7 double jeod::BodyForceCollect::effector\_forc[3]**

Sum of effector forces, struct ref.

trick\_units(N)

Definition at line 281 of file body\_force\_collect.hh.

Referenced by BodyForceCollect(), jeod::StructureIntegratedDynBody::collect\_forces\_and\_torques(), jeod::DynBody::collect\_forces\_and\_torques(), jeod::StructureIntegratedDynBody::collect\_local\_forces\_and\_torques(), and jeod::StructureIntegratedDynBody::PropagateForcesAndTorques().

**8.1.4.8 double jeod::BodyForceCollect::effector\_torq[3]**

Sum of effector torques about body CoM, struct ref.

trick\_units(N\*m)

Definition at line 306 of file body\_force\_collect.hh.

Referenced by BodyForceCollect(), jeod::StructureIntegratedDynBody::collect\_forces\_and\_torques(), jeod::DynBody::collect\_forces\_and\_torques(), jeod::StructureIntegratedDynBody::collect\_local\_forces\_and\_torques(), and jeod::StructureIntegratedDynBody::PropagateForcesAndTorques().

**8.1.4.9 double jeod::BodyForceCollect::environ\_forc[3]**

Sum of env forces, struct ref.

trick\_units(N)

Definition at line 286 of file body\_force\_collect.hh.

Referenced by BodyForceCollect(), jeod::StructureIntegratedDynBody::collect\_forces\_and\_torques(), jeod::DynBody::collect\_forces\_and\_torques(), jeod::StructureIntegratedDynBody::collect\_local\_forces\_and\_torques(), and jeod::StructureIntegratedDynBody::PropagateForcesAndTorques().

**8.1.4.10 double jeod::BodyForceCollect::environ\_torq[3]**

Sum of environment torqs about body CoM, struct ref.

trick\_units(N\*m)

Definition at line 311 of file body\_force\_collect.hh.

Referenced by BodyForceCollect(), jeod::StructureIntegratedDynBody::collect\_forces\_and\_torques(), jeod::DynBody::collect\_forces\_and\_torques(), jeod::StructureIntegratedDynBody::collect\_local\_forces\_and\_torques(), and jeod::StructureIntegratedDynBody::PropagateForcesAndTorques().

**8.1.4.11 double jeod::BodyForceCollect::extern\_forc\_inrtl[3]**

Sum of external forces, inertial.

trick\_units(N)

Definition at line 301 of file body\_force\_collect.hh.

Referenced by BodyForceCollect(), jeod::StructureIntegratedDynBody::collect\_forces\_and\_torques(), jeod::DynBody::collect\_forces\_and\_torques(), and jeod::StructureIntegratedDynBody::compute\_translational\_acceleration().

#### 8.1.4.12 double jeod::BodyForceCollect::extern\_forc\_struct[3]

Sum of external forces, struct ref.

trick\_units(N)

Definition at line 296 of file body\_force\_collect.hh.

Referenced by BodyForceCollect(), jeod::StructureIntegratedDynBody::collect\_forces\_and\_torques(), jeod::DynBody::collect\_forces\_and\_torques(), jeod::StructureIntegratedDynBody::compute\_translational\_acceleration(), and jeod::StructureIntegratedDynBody::solve\_constraints().

#### 8.1.4.13 double jeod::BodyForceCollect::extern\_torq\_body[3]

Sum of external torques, body ref.

trick\_units(N\*m)

Definition at line 332 of file body\_force\_collect.hh.

Referenced by BodyForceCollect(), jeod::StructureIntegratedDynBody::collect\_forces\_and\_torques(), jeod::DynBody::collect\_forces\_and\_torques(), and jeod::StructureIntegratedDynBody::compute\_rotational\_acceleration().

#### 8.1.4.14 double jeod::BodyForceCollect::extern\_torq\_struct[3]

Sum of external torques, struct ref.

trick\_units(N\*m)

Definition at line 327 of file body\_force\_collect.hh.

Referenced by BodyForceCollect(), jeod::StructureIntegratedDynBody::collect\_forces\_and\_torques(), jeod::DynBody::collect\_forces\_and\_torques(), and jeod::StructureIntegratedDynBody::compute\_rotational\_acceleration().

#### 8.1.4.15 double jeod::BodyForceCollect::inertial\_torq[3]

Induced inertial torques from second order rotational dynamics,  $w \times lw$ , body ref.

trick\_units(N\*m)

Definition at line 322 of file body\_force\_collect.hh.

Referenced by BodyForceCollect(), jeod::StructureIntegratedDynBody::collect\_forces\_and\_torques(), jeod::DynBody::collect\_forces\_and\_torques(), jeod::StructureIntegratedDynBody::compute\_inertial\_torque(), jeod::StructureIntegratedDynBody::compute\_rotational\_acceleration(), and jeod::StructureIntegratedDynBody::solve\_constraints().

#### 8.1.4.16 double jeod::BodyForceCollect::no\_xmit\_forc[3]

Sum of local forces, struct ref.

trick\_units(N)

Definition at line 291 of file body\_force\_collect.hh.

Referenced by BodyForceCollect(), jeod::StructureIntegratedDynBody::collect\_forces\_and\_torques(), jeod::DynBody::collect\_forces\_and\_torques(), and jeod::StructureIntegratedDynBody::collect\_local\_forces\_and\_torques().

#### 8.1.4.17 double jeod::BodyForceCollect::no\_xmit\_torq[3]

Sum of torqs not transmitted to a parent about body CoM, struct ref.

trick\_units(N\*m)

Definition at line 316 of file body\_force\_collect.hh.

Referenced by BodyForceCollect(), jeod::StructureIntegratedDynBody::collect\_forces\_and\_torques(), jeod::DynBody::collect\_forces\_and\_torques(), and jeod::StructureIntegratedDynBody::collect\_local\_forces\_and\_torques().

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

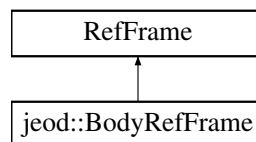
- [body\\_force\\_collect.hh](#)
- [aux\\_classes.cc](#)

## 8.2 jeod::BodyRefFrame Class Reference

Extend RefFrame to add coupling between the reference frame tree and the mass tree and to keep track of which state items have been set.

```
#include <body_ref_frame.hh>
```

Inheritance diagram for jeod::BodyRefFrame:



### Public Member Functions

- [BodyRefFrame](#) (void)  
*Default constructor.*
- [~BodyRefFrame](#) (void) override  
*Destructor.*

### Data Fields

- RefFrameItems [initialized\\_items](#)  
*Specifies which state elements (position, velocity, attitude, and rate) have been initialized.*
- MassPoint \* [mass\\_point](#)  
*Pointer to the mass point that defines the origin and orientation of this frame, but with respect to the mass tree rather than with respect to the reference frame tree.*

### Private Member Functions

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

### Friends

- class [InputProcessor](#)
- void [init\\_attrjeod\\_\\_BodyRefFrame](#) ()

### 8.2.1 Detailed Description

Extend RefFrame to add coupling between the reference frame tree and the mass tree and to keep track of which state items have been set.

Definition at line 79 of file body\_ref\_frame.hh.

### 8.2.2 Constructor & Destructor Documentation

8.2.2.1 `jeod::BodyRefFrame::BodyRefFrame ( const BodyRefFrame & ) [private]`

8.2.2.2 `jeod::BodyRefFrame::BodyRefFrame ( void ) [inline]`

Default constructor.

Definition at line 126 of file body\_ref\_frame.hh.

8.2.2.3 `jeod::BodyRefFrame::~~BodyRefFrame ( void ) [inline], [override]`

Destructor.

Definition at line 140 of file body\_ref\_frame.hh.

### 8.2.3 Member Function Documentation

8.2.3.1 `BodyRefFrame& jeod::BodyRefFrame::operator= ( const BodyRefFrame & ) [private]`

### 8.2.4 Friends And Related Function Documentation

8.2.4.1 `void init_attrjeod__BodyRefFrame ( ) [friend]`

8.2.4.2 `friend class InputProcessor [friend]`

Definition at line 81 of file body\_ref\_frame.hh.

### 8.2.5 Field Documentation

8.2.5.1 `RefFrameItems jeod::BodyRefFrame::initialized_items`

Specifies which state elements (position, velocity, attitude, and rate) have been initialized.

trick\_units(-)

Definition at line 92 of file body\_ref\_frame.hh.

Referenced by `jeod::DynBody::compute_derived_state_forward()`, `jeod::DynBody::compute_derived_state_reverse()`, `jeod::DynBody::compute_state_elements_forward()`, `jeod::DynBody::compute_state_elements_reverse()`, `jeod::DynBody::propagate_state_from_composite()`, `jeod::DynBody::propagate_state_from_structure()`, `jeod::DynBody::set_state_source_internal()`, and `jeod::DynBody::update_integrated_state()`.

8.2.5.2 `MassPoint* jeod::BodyRefFrame::mass_point`

Pointer to the mass point that defines the origin and orientation of this frame, but with respect to the mass tree rather than with respect to the reference frame tree.

trick\_units(-)

Definition at line 99 of file `body_ref_frame.hh`.

Referenced by `jeod::DynBody::add_mass_body()`, `jeod::DynBody::add_mass_body_frames()`, `jeod::DynBody::add_mass_point()`, `jeod::DynBody::attach_child()`, `jeod::DynBody::attach_to_frame()`, `jeod::DynBody::compute_ref_point_transform()`, `jeod::StructureIntegratedDynBody::compute_vehicle_point_derivatives()`, `jeod::DynBody::compute_vehicle_point_derivatives()`, `jeod::DynBody::compute_vehicle_point_states()`, and `jeod::DynBody::DynBody()`.

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

- [body\\_ref\\_frame.hh](#)

## 8.3 jeod::BodyWrenchCollect Class Reference

Serves as the collection point for wrenches that act on a vehicle.

```
#include <body_wrench_collect.hh>
```

### Public Member Functions

- [BodyWrenchCollect](#) ()  
*Default constructor.*
- [~BodyWrenchCollect](#) ()  
*Destructor.*
- [BodyWrenchCollect](#) (const [BodyWrenchCollect](#) &)=delete
- [BodyWrenchCollect](#) & operator= (const [BodyWrenchCollect](#) &)=delete
- [Wrench](#) & accumulate ([Wrench](#) &sum) const  
*Accumulate the collected wrenches.*
- [Wrench](#) & accumulate (const double point[3], [Wrench](#) &sum) const  
*Accumulate the collected wrenches.*

### Data Fields

- `JeodPointerVector< Wrench >::type collect_wrench`  
*Vector of effector wrenches.*

#### 8.3.1 Detailed Description

Serves as the collection point for wrenches that act on a vehicle.

This is a simple class that is tightly coupled with the [StructureIntegratedDynBody](#) class. This latter class contains (has-a) a [BodyWrenchCollect](#) data member.

The Trick vcollect mechanism (or a similar mechanism in a non-Trick sim) pushes pointers to the individual wrenches onto the various collection member of a [BodyWrenchCollect](#). [StructureIntegratedDynBody](#) members cumulate these collected wrenches to form the total wrench acting on the vehicle.

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

#### 8.3.2 Constructor & Destructor Documentation

##### 8.3.2.1 jeod::BodyWrenchCollect::BodyWrenchCollect ( )

Default constructor.

Definition at line 26 of file body\_wrench\_collect.cc.

References collect\_wrench.

#### 8.3.2.2 jeod::BodyWrenchCollect::~BodyWrenchCollect ( )

Destructor.

Definition at line 35 of file body\_wrench\_collect.cc.

References collect\_wrench.

#### 8.3.2.3 jeod::BodyWrenchCollect::BodyWrenchCollect ( const BodyWrenchCollect & ) [delete]

### 8.3.3 Member Function Documentation

#### 8.3.3.1 Wrench& jeod::BodyWrenchCollect::accumulate ( Wrench & sum ) const [inline]

Accumulate the collected wrenches.

Parameters

<i>sum</i>	<a href="#">Wrench</a> into which the accumulated sum is to be placed. The summation is about sum.point.
------------	--

Returns

Reference to the input wrench.

Definition at line 131 of file body\_wrench\_collect.hh.

References jeod::Wrench::accumulate(), and collect\_wrench.

Referenced by accumulate(), and jeod::StructureIntegratedDynBody::collect\_local\_forces\_and\_torques().

#### 8.3.3.2 Wrench& jeod::BodyWrenchCollect::accumulate ( const double point[3], Wrench & sum ) const [inline]

Accumulate the collected wrenches.

Parameters

<i>point</i>	Point about which summation is to be performed.
<i>sum</i>	<a href="#">Wrench</a> into which the accumulated sum is to be placed.

Returns

Reference to the input wrench.

Definition at line 143 of file body\_wrench\_collect.hh.

References accumulate(), and jeod::Wrench::set\_point().

#### 8.3.3.3 BodyWrenchCollect& jeod::BodyWrenchCollect::operator= ( const BodyWrenchCollect & ) [delete]

### 8.3.4 Field Documentation

#### 8.3.4.1 JeodPointerVector<Wrench>::type jeod::BodyWrenchCollect::collect\_wrench

Vector of effector wrenches.

The effector wrenches are collected into the vector at the `S_define` level via `& vcollect` containing `body.effector_wrench_collection.collect_wrench { pointer_to_wrench1, ... pointer_to_wrench_n }`;

The vector of collected wrenches are processed by the containing body's `collect_forces_and_torques` member function `trick_io(**)`

Definition at line 100 of file `body_wrench_collect.hh`.

Referenced by `accumulate()`, `BodyWrenchCollect()`, and `~BodyWrenchCollect()`.

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

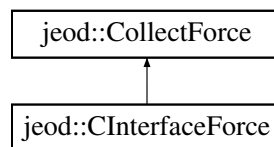
- [body\\_wrench\\_collect.hh](#)
- [body\\_wrench\\_collect.cc](#)

## 8.4 jeod::CInterfaceForce Class Reference

This class is deprecated.

```
#include <force.hh>
```

Inheritance diagram for `jeod::CInterfaceForce`:



### Public Member Functions

- [CInterfaceForce](#) ()  
*CInterfaceForce* default constructor.
- [CInterfaceForce](#) (double \*vec)  
*CInterfaceForce* constructor for use with C force array.
- [~CInterfaceForce](#) () override  
*CInterfaceForce* destructor; frees 'active' but not the force.

### Private Member Functions

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

### Additional Inherited Members

#### 8.4.1 Detailed Description

This class is deprecated.

Definition at line 227 of file `force.hh`.



## 8.4.2 Constructor & Destructor Documentation

### 8.4.2.1 jeod::CInterfaceForce::CInterfaceForce ( void )

[CInterfaceForce](#) default constructor.

Note that this has changed from JEOD 2.1. In JEOD 2.2 the default constructor of a JEOD-allocable class must not allocate any resources.

Definition at line 140 of file force.cc.

### 8.4.2.2 jeod::CInterfaceForce::CInterfaceForce ( double \* force\_3vec ) [explicit]

[CInterfaceForce](#) constructor for use with C force array.

Note that the new [CInterfaceForce](#)'s force is the force\_3vec.

Parameters

in, out	force_3vec	<a href="#">Force</a> vector to encapsulate Units: N
---------	------------	---

Definition at line 154 of file force.cc.

References [jeod::CollectForce::active](#), and [jeod::CollectForce::force](#).

### 8.4.2.3 jeod::CInterfaceForce::~~CInterfaceForce ( void ) [override]

[CInterfaceForce](#) destructor; frees 'active' but not the force.

Definition at line 167 of file force.cc.

References [jeod::CollectForce::active](#).

### 8.4.2.4 jeod::CInterfaceForce::CInterfaceForce ( const CInterfaceForce & ) [private]

Not implemented.

## 8.4.3 Member Function Documentation

### 8.4.3.1 CInterfaceForce& jeod::CInterfaceForce::operator= ( const CInterfaceForce & ) [private]

Not implemented.

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

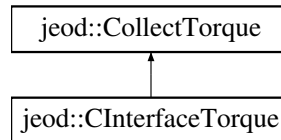
- [force.hh](#)
- [force.cc](#)

## 8.5 jeod::CInterfaceTorque Class Reference

This class is deprecated.

```
#include <torque.hh>
```

Inheritance diagram for [jeod::CInterfaceTorque](#):



## Public Member Functions

- [CInterfaceTorque](#) ()  
*CInterfaceTorque* default constructor.
- [CInterfaceTorque](#) (double \*vec)  
*CInterfaceTorque* constructor for use with C torque array.
- [~CInterfaceTorque](#) () override  
*CInterfaceTorque* destructor; frees 'active' but not the torque.

## Private Member Functions

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

## Additional Inherited Members

### 8.5.1 Detailed Description

This class is deprecated.

Definition at line 222 of file torque.hh.

### 8.5.2 Constructor & Destructor Documentation

#### 8.5.2.1 jeod::CInterfaceTorque::CInterfaceTorque ( void )

[CInterfaceTorque](#) default constructor.

Note that this has changed from JEOD 2.1: In JEOD 2.2 the default constructor of a JEOD-allocable class must not allocate any resources.

Definition at line 140 of file torque.cc.

#### 8.5.2.2 jeod::CInterfaceTorque::CInterfaceTorque ( double \* torque\_3vec ) [explicit]

[CInterfaceTorque](#) constructor for use with C torque array.

Note that the new [CInterfaceTorque](#)'s torque is the torque\_3vec.

#### Parameters

<code>in, out</code>	<code>torque_3vec</code>	<a href="#">Torque</a> vector to encapsulate Units: NM
----------------------	--------------------------	---

Definition at line 154 of file torque.cc.

References [jeod::CollectTorque::active](#), and [jeod::CollectTorque::torque](#).

8.5.2.3 `jeod::CInterfaceTorque::~~CInterfaceTorque ( void ) [override]`

[CInterfaceTorque](#) destructor; frees 'active' but not the torque.

Definition at line 167 of file torque.cc.

References [jeod::CollectTorque::active](#).

8.5.2.4 `jeod::CInterfaceTorque::CInterfaceTorque ( const CInterfaceTorque & ) [private]`

Not implemented.

### 8.5.3 Member Function Documentation

8.5.3.1 `CInterfaceTorque& jeod::CInterfaceTorque::operator= ( const CInterfaceTorque & ) [private]`

Not implemented.

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

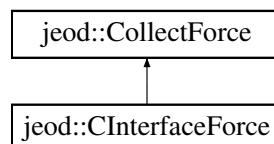
- [torque.hh](#)
- [torque.cc](#)

## 8.6 jeod::CollectForce Class Reference

A [CollectForce](#) represents a collected force that acts on a vehicle.

```
#include <force.hh>
```

Inheritance diagram for `jeod::CollectForce`:



### Public Member Functions

- [CollectForce \(\)](#)  
*CollectForce* default constructor.
- [CollectForce \(double vec\[3\]\)](#)  
*CollectForce* constructor that encapsulates a C-style 3-vector.
- [CollectForce \(Force &\)](#)  
*CollectForce* constructor that encapsulates a [Force](#).
- [CollectForce \(CollectForce &\)](#)  
*CollectForce* constructor that encapsulates another [CollectForce](#).
- virtual [~CollectForce \(\)](#)  
*CollectForce* destructor.
- bool [is\\_active \(\)](#) const  
A force is active if it has a non-null force vector and the active pointer is null or the pointed-to boolean is true.
- double & [operator\[\]](#) (const unsigned int index)  
Access a force element, non-const version.

- double `operator[]` (const unsigned int index) const  
*Access a force element, const version.*
- bool `operator==` (const `CollectForce` &other)

### Static Public Member Functions

- static `CollectForce * create` (double \*vec)  
*Create a `CollectForce` whose force is the specified array.*
- static `CollectForce * create` (`Force` &force)  
*Create a shallow copy of a `Force`.*
- static `CollectForce * create` (`CollectForce` &force)  
*Create a shallow copy of a `CollectForce`.*
- static `CollectForce * create` (`Force` \*force)  
*Create a shallow copy of a `Force`.*
- static `CollectForce * create` (`CollectForce` \*force)  
*Create a shallow copy of a `CollectForce`.*

### Data Fields

- bool \* `active`  
*Is this force active?*
- double \* `force`  
*`Force` vector.*

### Private Member Functions

- `CollectForce` (const `CollectForce` &)  
*Not implemented.*
- `CollectForce & operator=` (const `CollectForce` &)  
*Not implemented.*

## 8.6.1 Detailed Description

A `CollectForce` represents a collected force that acts on a vehicle.

The `BodyForceCollect` class contains STL vectors that in turn contain `CollectForce` pointers. These vectors are populated via the Trick vcollect mechanism. A Trick simulation issues vcollect statements such as

```
vcollect vehicle.body.collect.collect_XXX_forc CollectForce::create {
    vehicle.force_model1.force,
    vehicle.force_model2.force
};
```

This invokes the appropriate `CollectForce` create method on each listed element.

CollectForces should not be used in model code to represent forces. Use the `Force` class instead.

Definition at line 149 of file force.hh.

## 8.6.2 Constructor & Destructor Documentation

### 8.6.2.1 jeod::CollectForce::CollectForce ( void )

`CollectForce` default constructor.

Definition at line 67 of file force.cc.

**8.6.2.2** `jeod::CollectForce::CollectForce ( double force_3vec[3] ) [explicit]`

[CollectForce](#) constructor that encapsulates a C-style 3-vector.

Note that the new [CollectForce](#)'s force *is* the *force\_3vec*.

**Parameters**

<i>in, out</i>	<i>force_3vec</i>	<a href="#">Force</a> vector to encapsulate Units: N
----------------	-------------------	---

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

**8.6.2.3** `jeod::CollectForce::CollectForce ( Force & source_force ) [explicit]`

[CollectForce](#) constructor that encapsulates a [Force](#).

Note that this performs a shallow copy by intent.

**Parameters**

<i>in, out</i>	<i>source_force</i>	<a href="#">Force</a> to encapsulate
----------------	---------------------	--------------------------------------

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

**8.6.2.4** `jeod::CollectForce::CollectForce ( CollectForce & source_force ) [explicit]`

[CollectForce](#) constructor that encapsulates another [CollectForce](#).

Note that this performs a shallow copy by intent.

**Parameters**

<i>in, out</i>	<i>source_force</i>	<a href="#">Force</a> to encapsulate
----------------	---------------------	--------------------------------------

Definition at line 112 of file `force.cc`.

**8.6.2.5** `jeod::CollectForce::~~CollectForce ( void ) [virtual]`

[CollectForce](#) destructor.

Note that this does not free any element memory.

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

**8.6.2.6** `jeod::CollectForce::CollectForce ( const CollectForce & ) [private]`

Not implemented.

**8.6.3 Member Function Documentation****8.6.3.1** `CollectForce * jeod::CollectForce::create ( double * force_3vec ) [static]`

Create a [CollectForce](#) whose force is the specified array.

Note that the created instance is actually a [CInterfaceForce](#).

**Returns**

Constructed [CollectForce](#)

## Parameters

<i>in, out</i>	<i>force_3vec</i>	<a href="#">Force</a> vector to encapsulate Units: N
----------------	-------------------	---

Definition at line 214 of file force.cc.

Referenced by `create()`.

#### 8.6.3.2 `CollectForce * jeod::CollectForce::create ( Force & source_force ) [static]`

Create a shallow copy of a [Force](#).

Note that the new [CollectForce](#) refers to the [Force](#)'s active flag and force array.

## Returns

Constructed [CollectForce](#)

## Parameters

<i>in, out</i>	<i>source_force</i>	<a href="#">Force</a> object to encapsulate
----------------	---------------------	---

Definition at line 185 of file force.cc.

#### 8.6.3.3 `CollectForce * jeod::CollectForce::create ( CollectForce & source_force ) [static]`

Create a shallow copy of a [CollectForce](#).

Note that both the source and new [CollectForces](#) refer to the same active flag and force array.

## Returns

Constructed [CollectForce](#)

## Parameters

<i>in, out</i>	<i>source_force</i>	<a href="#">Force</a> to copy
----------------	---------------------	-------------------------------

Definition at line 229 of file force.cc.

#### 8.6.3.4 `CollectForce * jeod::CollectForce::create ( Force * source_force ) [static]`

Create a shallow copy of a [Force](#).

Note that the new [CollectForce](#) refers to the [Force](#)'s active flag and force array.

## Returns

Constructed [CollectForce](#)

## Parameters

<i>in, out</i>	<i>source_force</i>	<a href="#">Force</a> object to encapsulate
----------------	---------------------	---

Definition at line 200 of file force.cc.

References `create()`.

#### 8.6.3.5 `CollectForce * jeod::CollectForce::create ( CollectForce * source_force ) [static]`

Create a shallow copy of a [CollectForce](#).

Note that both the source and new [CollectForces](#) refer to the same active flag and force array.

**Returns**

Constructed [CollectForce](#)

**Parameters**

<i>in, out</i>	<i>source_force</i>	<a href="#">Force</a> to copy
----------------	---------------------	-------------------------------

Definition at line 244 of file force.cc.

References [create\(\)](#).

**8.6.3.6 bool jeod::CollectForce::is\_active ( void ) const [inline]**

A force is active if it has a non-null force vector and the active pointer is null or the pointed-to boolean is true.

**Returns**

Is the force active?

Definition at line 104 of file force\_inline.hh.

References [active](#), and [force](#).

**8.6.3.7 CollectForce& jeod::CollectForce::operator= ( const CollectForce & ) [private]**

Not implemented.

**8.6.3.8 bool jeod::CollectForce::operator== ( const CollectForce & other ) [inline]**

Definition at line 185 of file force.hh.

References [force](#).

**8.6.3.9 double & jeod::CollectForce::operator[] ( const unsigned int index ) [inline]**

Access a force element, non-const version.

**Returns**

[Force](#) component at specified index  
Units: N

**Parameters**

<i>in</i>	<i>index</i>	Index number
-----------	--------------	--------------

Definition at line 118 of file force\_inline.hh.

References [force](#).

**8.6.3.10 double jeod::CollectForce::operator[] ( const unsigned int index ) const [inline]**

Access a force element, const version.

**Returns**

[Force](#) component at specified index  
Units: N

## Parameters

<code>in</code>	<code>index</code>	Index number
-----------------	--------------------	--------------

Definition at line 131 of file `force_inline.hh`.

References `force`.

## 8.6.4 Field Documentation

### 8.6.4.1 `bool* jeod::CollectForce::active`

Is this force active?

`trick_units(-)`

Definition at line 197 of file `force.hh`.

Referenced by `jeod::CInterfaceForce::CInterfaceForce()`, `is_active()`, and `jeod::CInterfaceForce::~~CInterfaceForce()`.

### 8.6.4.2 `double* jeod::CollectForce::force`

[Force](#) vector.

`trick_units(N)`

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

Referenced by `jeod::CInterfaceForce::CInterfaceForce()`, `is_active()`, `operator==()`, and `operator[]()`.

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

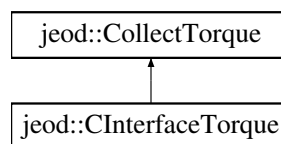
- [force.hh](#)
- [force\\_inline.hh](#)
- [force.cc](#)

## 8.7 `jeod::CollectTorque` Class Reference

A [CollectTorque](#) represents a collected torque that acts on a vehicle.

```
#include <torque.hh>
```

Inheritance diagram for `jeod::CollectTorque`:



### Public Member Functions

- [CollectTorque](#) ()  
*CollectTorque default constructor.*
- [CollectTorque](#) (double vec[3])  
*CollectTorque constructor that encapsulates a C-style 3-vector.*
- [CollectTorque](#) ([Torque](#) &)



- *CollectTorque* constructor that encapsulates a *Torque*.
- *CollectTorque* (*CollectTorque* &)
- *CollectTorque* constructor that encapsulates another *CollectTorque*.
- virtual *~CollectTorque* ()
- *CollectTorque* destructor.
- bool *is\_active* () const
- *A torque is active if it has a non-null torque vector and the active pointer is null or the pointed-to boolean is true.*
- double & *operator[]* (const unsigned int index)
- *Access a torque element, non-const version.*
- double *operator[]* (const unsigned int index) const
- *Access a torque element, const version.*
- bool *operator==* (const *CollectTorque* &other)

### Static Public Member Functions

- static *CollectTorque* \* *create* (double \*vec)
- *Create a CollectTorque whose torque is the specified array.*
- static *CollectTorque* \* *create* (*Torque* &torque)
- *Create a shallow copy of a Torque.*
- static *CollectTorque* \* *create* (*CollectTorque* &torque)
- *Create a shallow copy of a CollectTorque.*
- static *CollectTorque* \* *create* (*Torque* \*torque)
- *Create a shallow copy of a Torque.*
- static *CollectTorque* \* *create* (*CollectTorque* \*torque)
- *Create a shallow copy of a CollectTorque.*

### Data Fields

- bool \* *active*
- *Is this torque active?*
- double \* *torque*
- *Torque vector.*

### Private Member Functions

- *CollectTorque* (const *CollectTorque* &)
- *Not implemented.*
- *CollectTorque* & *operator=* (const *CollectTorque* &)
- *Not implemented.*

#### 8.7.1 Detailed Description

A *CollectTorque* represents a collected torque that acts on a vehicle.

The *BodyTorqueCollect* class contains STL vectors that in turn contain *CollectTorque* pointers. These vectors are populated via the Trick *vcollect* mechanism. A Trick simulation issues *vcollect* statements such as

```
vcollect vehicle.body.collect.collect_XXX_forc CollectTorque::create {
    vehicle.torque_model1.torque,
    vehicle.torque_model2.torque
};
```

This invokes the appropriate [CollectTorque](#) create method on each listed element.

CollectTorques should not be used in model code to represent torques. Use the [Torque](#) class instead.

Definition at line 147 of file torque.hh.

## 8.7.2 Constructor & Destructor Documentation

### 8.7.2.1 `jeod::CollectTorque::CollectTorque ( void )`

[CollectTorque](#) default constructor.

Definition at line 67 of file torque.cc.

### 8.7.2.2 `jeod::CollectTorque::CollectTorque ( double torque_3vec[3] ) [explicit]`

[CollectTorque](#) constructor that encapsulates a C-style 3-vector.

Note that the new [CollectTorque](#)'s torque *is* the *torque\_3vec*.

Parameters

<i>in, out</i>	<i>torque_3vec</i>	<a href="#">Torque</a> vector to encapsulate Units: NM
----------------	--------------------	---

Definition at line 97 of file torque.cc.

### 8.7.2.3 `jeod::CollectTorque::CollectTorque ( Torque & source_torque ) [explicit]`

[CollectTorque](#) constructor that encapsulates a [Torque](#).

Note that this performs a shallow copy by intent.

Parameters

<i>in, out</i>	<i>source_torque</i>	<a href="#">Torque</a> to encapsulate
----------------	----------------------	---------------------------------------

Definition at line 82 of file torque.cc.

### 8.7.2.4 `jeod::CollectTorque::CollectTorque ( CollectTorque & source_torque ) [explicit]`

[CollectTorque](#) constructor that encapsulates another [CollectTorque](#).

Note that this performs a shallow copy by intent.

Parameters

<i>in, out</i>	<i>source_torque</i>	<a href="#">Torque</a> to encapsulate
----------------	----------------------	---------------------------------------

Definition at line 112 of file torque.cc.

### 8.7.2.5 `jeod::CollectTorque::~~CollectTorque ( void ) [virtual]`

[CollectTorque](#) destructor.

Note that this does not free any element memory.

Definition at line 126 of file torque.cc.

## 8.7.2.6 jeod::CollectTorque::CollectTorque ( const CollectTorque &amp; ) [private]

Not implemented.

## 8.7.3 Member Function Documentation

## 8.7.3.1 CollectTorque \* jeod::CollectTorque::create ( double \* torque\_3vec ) [static]

Create a [CollectTorque](#) whose torque is the specified array.

Note that the created instance is actually a [CInterfaceTorque](#).

## Returns

Constructed [CollectTorque](#)

## Parameters

<i>in, out</i>	<i>torque_3vec</i>	<a href="#">Torque</a> vector to encapsulate Units: NM
----------------	--------------------	---

Definition at line 214 of file torque.cc.

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

## 8.7.3.2 CollectTorque \* jeod::CollectTorque::create ( Torque &amp; source\_torque ) [static]

Create a shallow copy of a [Torque](#).

Note that the new [CollectTorque](#) refers to the [Torque](#)'s active flag and torque array.

## Returns

Constructed [CollectTorque](#)

## Parameters

<i>in, out</i>	<i>source_torque</i>	<a href="#">Torque</a> object to encapsulate
----------------	----------------------	--

Definition at line 185 of file torque.cc.

## 8.7.3.3 CollectTorque \* jeod::CollectTorque::create ( CollectTorque &amp; source\_torque ) [static]

Create a shallow copy of a [CollectTorque](#).

Note that both the source and new [CollectTorques](#) refer to the same active flag and torque array.

## Returns

Constructed [CollectTorque](#)

## Parameters

<i>in, out</i>	<i>source_torque</i>	<a href="#">Torque</a> to copy
----------------	----------------------	--------------------------------

Definition at line 229 of file torque.cc.

## 8.7.3.4 CollectTorque \* jeod::CollectTorque::create ( Torque \* source\_torque ) [static]

Create a shallow copy of a [Torque](#).

Note that the new [CollectTorque](#) refers to the [Torque](#)'s active flag and torque array.

**Returns**

Constructed [CollectTorque](#)

**Parameters**

<a href="#">in</a> , <a href="#">out</a>	<i>source_torque</i>	<a href="#">Torque</a> object to encapsulate
--	----------------------	--

Definition at line 200 of file torque.cc.

References [create\(\)](#).

**8.7.3.5** `CollectTorque * jeod::CollectTorque::create ( CollectTorque * source_torque ) [static]`

Create a shallow copy of a [CollectTorque](#).

Note that both the source and new CollectTorques refer to the same active flag and torque array.

**Returns**

Constructed [CollectTorque](#)

**Parameters**

<a href="#">in</a> , <a href="#">out</a>	<i>source_torque</i>	<a href="#">Torque</a> to copy
--	----------------------	--------------------------------

Definition at line 244 of file torque.cc.

References [create\(\)](#).

**8.7.3.6** `bool jeod::CollectTorque::is_active ( void ) const [inline]`

A torque is active if it has a non-null torque vector and the active pointer is null or the pointed-to boolean is true.

**Returns**

Is the torque active?

Definition at line 104 of file torque\_inline.hh.

References [active](#), and [torque](#).

**8.7.3.7** `CollectTorque& jeod::CollectTorque::operator= ( const CollectTorque & ) [private]`

Not implemented.

**8.7.3.8** `bool jeod::CollectTorque::operator== ( const CollectTorque & other ) [inline]`

Definition at line 180 of file torque.hh.

References [torque](#).

**8.7.3.9** `double & jeod::CollectTorque::operator[] ( const unsigned int index ) [inline]`

Access a torque element, non-const version.

**Returns**

[Torque](#) component at specified index  
Units: N

## Parameters

<i>in</i>	<i>index</i>	Index number
-----------	--------------	--------------

Definition at line 118 of file torque\_inline.hh.

References torque.

**8.7.3.10** `double jeod::CollectTorque::operator[] ( const unsigned int index ) const` `[inline]`

Access a torque element, const version.

## Returns

[Torque](#) component at specified index

Units: N

## Parameters

<i>in</i>	<i>index</i>	Index number
-----------	--------------	--------------

Definition at line 131 of file torque\_inline.hh.

References torque.

## 8.7.4 Field Documentation

**8.7.4.1** `bool* jeod::CollectTorque::active`

Is this torque active?

trick\_units(-)

Definition at line 192 of file torque.hh.

Referenced by jeod::CInterfaceTorque::CInterfaceTorque(), is\_active(), and jeod::CInterfaceTorque::~~CInterfaceTorque().

**8.7.4.2** `double* jeod::CollectTorque::torque`

[Torque](#) vector.

trick\_units(N\*m)

Definition at line 197 of file torque.hh.

Referenced by jeod::CInterfaceTorque::CInterfaceTorque(), is\_active(), operator==(), and operator[]().

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

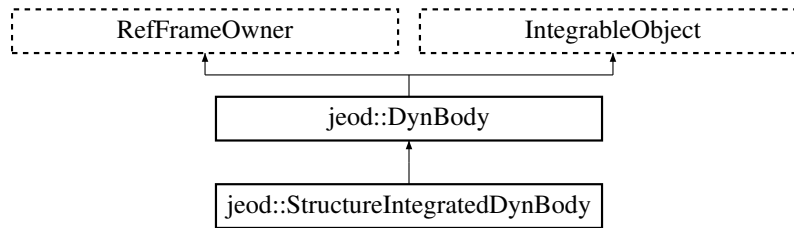
- [torque.hh](#)
- [torque\\_inline.hh](#)
- [torque.cc](#)

## 8.8 jeod::DynBody Class Reference

Class [DynBody](#) is the base class for all dynamic bodies.

```
#include <dyn_body.hh>
```

Inheritance diagram for jeod::DynBody:



## Public Member Functions

- [DynBody](#) ()  
*DynBody* default constructor.
- [~DynBody](#) () override  
*DynBody* destructor.
- virtual void [initialize\\_model](#) (BaseDynManager &dyn\_manager\_in)  
*Initialize internal and external interrelations, including registration / with the dynamics manager.*
- void [activate](#) ()  
*Activate a *DynBody* object.*
- void [deactivate](#) ()  
*Deactivate a *DynBody* object.*
- void [set\\_name](#) (const std::string &name\_in)  
*Set the name of the vehicle.*
- virtual void [add\\_control](#) (GravityControls \*control)  
*Add a new GravityControls to the list in grav\_interaction.*
- virtual void [initialize\\_controls](#) (GravityManager &grav\_manager)  
*Initialize the gravity controls of this *DynBody*.*
- virtual void [reset\\_controls](#) ()  
*Make the frame subscriptions for each control consistent with the requirements for that control.*
- virtual void [sort\\_controls](#) ()  
*Sort the gravity controls in ascending acceleration magnitude order.*
- virtual void [collect\\_forces\\_and\\_torques](#) ()  
*Collect forces and torques acting on the vehicle.*
- virtual void [create\\_body\\_integrators](#) (const er7\_utils::IntegratorConstructor &generator, er7\_utils::IntegrationControls &controls, const JeodIntegrationTime &time\_mgr)  
*Create the integrator (integrators) needed to propagate the translational and rotational state of a *DynBody*.*
- er7\_utils::IntegratorResult [integrate](#) (double dyn\_dt, unsigned int target\_stage) override  
*Integrate state by the specified dynamic time interval.*
- virtual void [switch\\_integration\\_frames](#) (EphemerisRefFrame &new\_integ\_frame)  
*Switch the integration frame for this body and all its child bodies to the indicated frame.*
- virtual void [switch\\_integration\\_frames](#) (const char \*new\_integ\_frame\_name)  
*Switch the integration frame for this body and all its child bodies to the frame indicated by the provided name.*
- void [create\\_integrators](#) (const er7\_utils::IntegratorConstructor &generator, er7\_utils::IntegrationControls &controls, const er7\_utils::TimeInterface &time\_if) override  
*This interface is required by er7\_utils::IntegrableObject.*
- void [destroy\\_integrators](#) (void) override  
*Destroy the integrators.*
- void [reset\\_integrators](#) (void) override  
*Reset the translational and rotational integrators.*
- virtual [BodyRefFrame](#) \* [find\\_body\\_frame](#) (const char \*frame\_id) const  
*Find the *BodyRefFrame* named by the provided identifier.*
- DynamicsIntegrationGroup \* [get\\_dynamics\\_integration\\_group](#) ()

- Get the *DynamicsIntegrationGroup* that integrates this *DynBody* object.
- JeodPointerVector
  - < er7\_utils::IntegrableObject >
  - ::type [get\\_integrable\\_objects](#) ()
  - Get the *IntegrableObjects* associated with this *DynBody*.
- void [clear\\_integrable\\_objects](#) ()
  - Remove all *IntegrableObjects* associated with this *DynBody*.
- void [migrate\\_integrable\\_objects](#) ()
  - Call this method before switching this dyn body to a new group if you want the associated integrable objects to follow.
- void [add\\_integrable\\_object](#) (er7\_utils::IntegrableObject &associated\_integrable\_object)
  - Add an *IntegrableObject* to be integrated with this *DynBody*.
- void [remove\\_integrable\\_object](#) (er7\_utils::IntegrableObject &associated\_integrable\_object)
  - Remove an *IntegrableObject* from association with this *DynBody*.
- void [set\\_position](#) (const double position[3], *BodyRefFrame* &subject\_frame)
  - Set the position of the vehicle.
- void [set\\_velocity](#) (const double velocity[3], *BodyRefFrame* &subject\_frame)
  - Set the velocity of the vehicle.
- void [set\\_attitude\\_left\\_quaternion](#) (const Quaternion &left\_quat, *BodyRefFrame* &subject\_frame)
  - Set the attitude of the vehicle.
- void [set\\_attitude\\_right\\_quaternion](#) (const Quaternion &right\_quat, *BodyRefFrame* &subject\_frame)
  - Set the attitude of the vehicle.
- void [set\\_attitude\\_matrix](#) (const double matrix[3][3], *BodyRefFrame* &subject\_frame)
  - Set the attitude of the vehicle.
- void [set\\_attitude\\_rate](#) (const double attitude\_rate[3], *BodyRefFrame* &subject\_frame)
  - Set the attitude rate of the vehicle.
- void [set\\_state](#) (RefFrameItems::Items set\_items, const RefFrameState &state, *BodyRefFrame* &subject\_frame)
  - Set the parts of the specified reference frame as indicated by the *set\_items* parameter from the supplied state and propagate these items to all dynamic bodies attached to this body.
- void [set\\_state\\_source](#) (RefFrameItems::Items items, *BodyRefFrame* &frame)
  - Set the source of aspects of the state.
- virtual void [propagate\\_state](#) ()
  - Propagate state from the integrated state to attached bodies.
- virtual void [update\\_integrated\\_state](#) ()
  - Propagate state from state owners to the integrated state.
- virtual void [compute\\_vehicle\\_point\\_states](#) (RefFrameItems::Items set\_items)
  - Propagate structure frame state to vehicle points.
- bool [is\\_root\\_body](#) ()
  - Indicates whether this *DynBody* object is a root body.
- virtual const *DynBody* \* [get\\_parent\\_body](#) () const
  - Returns this *DynBody* object's parent body.
- virtual const *DynBody* \* [get\\_root\\_body](#) () const
  - Finds this *DynBody* object's root body.
- virtual void [add\\_mass\\_point](#) (const MassPointInit &mass\_point\_init)
  - Add a mass point to the dyn body's list of such and make a vehicle point that corresponds to the added mass point.
- const *BodyRefFrame* \* [find\\_vehicle\\_point](#) (const char \*pt\_name) const
  - Find the vehicle point with the given name.
- virtual void [compute\\_vehicle\\_point\\_derivatives](#) (const *BodyRefFrame* &frame, *FrameDerivs* &derivs)
  - Compute the state derivatives at a vehicle point.
- const RefFrameItems & [get\\_initialized\\_states](#) () const
  - Indicate which state elements have been initialized.

- bool [initialized\\_states\\_contains](#) (RefFrameItems::Items test\_items) const  
*Indicate whether the specified state elements have been initialized.*
- virtual bool [add\\_mass\\_body](#) (const char \*this\_point\_name, const char \*child\_point\_name, MassBody &child)
- virtual bool [add\\_mass\\_body](#) (const double offset[3], const double T\_pstr\_cstr[3][3], MassBody &child)
- virtual bool [attach\\_to](#) (const char \*this\_point\_name, const char \*parent\_point\_name, [DynBody](#) &parent)  
*Attach this dyn body's root body as a child of the specified dyn body such that the specified mass points on the two bodies are coincident and the frames associated with those mass points are related by a 180 degree yaw.*
- virtual bool [attach\\_to](#) (const double offset\_pstr\_cstr\_pstr[3], const double T\_pstr\_cstr[3][3], [DynBody](#) &parent)  
*Attach this dyn body's root body as a child of the specified dyn body such that this body's structural origin is offset from the parent body's structural origin and this body's structural axes are oriented with respect to the parent body's structural axes as specified.*
- virtual bool [attach\\_child](#) (const char \*this\_point\_name, const char \*child\_point\_name, [DynBody](#) &child)  
*Attach a child [DynBody](#) by point specification.*
- virtual bool [attach\\_child](#) (const double offset\_pstr\_cstr\_pstr[3], const double T\_pstr\_cstr[3][3], [DynBody](#) &child)  
*Attach a child [DynBody](#) by location specification.*
- virtual bool [attach\\_to\\_frame](#) (const char \*parent\_ref\_frame\_name)
- virtual bool [attach\\_to\\_frame](#) (RefFrame &parent)
- virtual bool [attach\\_to\\_frame](#) (const char \*this\_point\_name, const char \*parent\_ref\_frame\_name, const double offset\_pframe\_cpt\_pframe[3], const double T\_pframe\_cpt[3][3])
- virtual bool [attach\\_to\\_frame](#) (const double offset\_pframe\_cstr\_pframe[3], const double T\_pframe\_cstr[3][3], RefFrame &parent)
- virtual bool [detach](#) ([DynBody](#) &other\_body)  
*Detach parent and child [DynBodies](#), 'this' and the argument body, such that the detachment happens at the parent body level.*
- virtual bool [detach](#) (void)  
*Detach this [DynBody](#) from its parent RefFrame or [DynBody](#) parent.*
- virtual bool [remove\\_mass\\_body](#) (MassBody &child)  
*Remove connectivity between this (parent) [DynBody](#) and the argument (child) MassBody mass subbody.*

## Data Fields

- MassBody [mass](#)  
*Mass properties of the vehicle, defined about the structure reference frame.*
- NamedItem & [name](#)  
*Body name, reference linked to mass.name.*
- char \* [integ\\_frame\\_name](#)  
*The name of the reference frame with respect to which the body's reference frames (core, composite, structure, plus vehicle point frames) are to be represented and propagated.*
- EphemerisRefFrame \* [integ\\_frame](#)  
*The current integration frame.*
- [BodyRefFrame](#) [core\\_body](#)  
*Vehicle core body reference frame.*
- [BodyRefFrame](#) [composite\\_body](#)  
*Vehicle composite body reference frame.*
- [BodyRefFrame](#) [structure](#)  
*Vehicle structural reference frame.*
- bool [translational\\_dynamics](#)  
*Is translational dynamics enabled? The body's translational state is integrated only if this member is true.*
- bool [rotational\\_dynamics](#)  
*Is rotational dynamics enabled? The body's rotational state is integrated only if this member is true.*
- bool [compute\\_point\\_derivative](#)



*Should the point derivatives for the body be computed? A child body's translational and rotational derivatives are only computed if this is true.*

- bool [three\\_dof](#)

*Is this a three degrees of freedom (translation only) body? This data member has effect only when set prior to the creation of the body's integrators.*

- GeneralizedSecondOrderODETechnique::TechniqueType [rotation\\_integration](#)

*Specifies the preferred mechanism for integrating rotational state.*

- bool [autoupdate\\_vehicle\\_points](#)

*Are vehicle points automatically updated? The vehicle points are automatically calculated at initialization time but are only automatically updated at runtime if this member is true.*

- GravityInteraction [grav\\_interaction](#)

*Gravitational interactions.*

- [FrameDerivs](#) [derivs](#)

*Translational/rotational accelerations.*

- [BodyForceCollect](#) [collect](#)

*Force/Torque collection mechanism.*

## Protected Member Functions

- virtual void [set\\_integ\\_frame](#) (EphemerisRefFrame &new\_integ\_frame)

*Set the integration frame for this body and all its child bodies to the provided frame.*

- virtual void [set\\_integ\\_frame](#) (const char \*new\_integ\_frame\_name)

*Set the integration frame for this body and all its child bodies to the frame indicated by the provided name.*

- virtual er7\_utils::IntegratorResult [trans\\_integ](#) (double dyn\_dt, unsigned int target\_stage)

*Integrate the vehicle's translational state.*

- virtual er7\_utils::IntegratorResult [rot\\_integ](#) (double dyn\_dt, unsigned int target\_stage)

*Integrate the vehicle's rotational state.*

- void [set\\_state\\_source\\_internal](#) (RefFrameItems::Items items, [BodyRefFrame](#) &frame)

*Set the source of aspects of the state.*

- virtual [DynBody](#) \* [get\\_parent\\_body\\_internal](#) ()

*Returns this [DynBody](#) object's parent body.*

- virtual [DynBody](#) \* [get\\_root\\_body\\_internal](#) ()

*Finds this [DynBody](#) object's root body.*

- virtual bool [attach\\_validate\\_parent](#) (const [DynBody](#) &parent, bool generate\_message) const

*Validate whether the pending attachment is legal from a connectivity point of view.*

- virtual bool [attach\\_validate\\_child](#) (const [DynBody](#) &child, bool generate\_message) const

*Validate whether the pending attachment is legal from a physical point of view.*

- virtual bool [add\\_mass\\_body\\_validate](#) (const [MassBody](#) &child, bool generate\_message) const

*Validate whether the pending sub body is legal from a mass tree point of view.*

- virtual void [add\\_mass\\_body\\_frames](#) ([MassBody](#) &subbody)

*For a newly attached mass sub-body, create body frames for the root sub-body and all child sub-bodies via recursion.*

- virtual void [detach\\_mass\\_body\\_frames](#) ([MassBody](#) &subbody)

*For a newly detached mass sub-body, remove body frames for the root sub-body and all child sub-bodies via recursion.*

- virtual void [attach\\_establish\\_links](#) ([DynBody](#) &parent)

*Establish the logical connectivity between parent and child.*

- virtual void [attach\\_update\\_properties](#) (const double offset\_pstr\_cstr\_pstr[3], const double T\_pstr\_cstr[3][3], [DynBody](#) &child)

*Set the relation between parent and child and update the mass properties.*

- virtual void [process\\_dynamic\\_attachment](#) (const double offset\_pstr\_cstr\_pstr[3], const double T\_pstr\_cstr[3][3], [DynBody](#) &root\_body, [DynBody](#) &child\_body)

*Process the attachment event of one body from another.*

- virtual void [detach\\_mass\\_internal](#) (MassBody &child)  
*Update parent and child properties to reflect that they are detached.*
- virtual void [propagate\\_state\\_from\\_structure](#) ()  
*Propagate state to attached bodies starting from this body's structural frame.*
- virtual void [propagate\\_state\\_from\\_composite](#) ()  
*Propagate state to attached bodies starting from this body's composite frame.*
- void [compute\\_ref\\_point\\_transform](#) (const [BodyRefFrame](#) &source\_frame, const MassPoint \*\*const ref\_point, MassPointState &rel\_state)  
*Compute the relative state between the integrated frame's mass point and the source frame's mass point.*
- void [compute\\_derived\\_state\\_forward](#) (const [BodyRefFrame](#) &source\_frame, const MassPoint &rel\_state, [BodyRefFrame](#) &derived\_frame) const  
*Compute a derived state given the source state and the position/ attitude transformation from the source to the derived state.*
- void [compute\\_state\\_elements\\_forward](#) (const [BodyRefFrame](#) &source\_frame, const MassPoint &rel\_state, const RefFrameItems &state\_items, [BodyRefFrame](#) &derived\_frame) const  
*Compute selected aspects of the derived state given the source state and the position/ attitude transformation from the source to the derived state.*
- void [compute\\_derived\\_state\\_reverse](#) (const [BodyRefFrame](#) &source\_frame, const MassPoint &rel\_state, [BodyRefFrame](#) &derived\_frame) const  
*Compute a derived state given the source state and the position/ attitude transformation from the derived to the source state.*
- void [compute\\_state\\_elements\\_reverse](#) (const [BodyRefFrame](#) &source\_frame, const MassPoint &rel\_state, const RefFrameItems &state\_items, [BodyRefFrame](#) &derived\_frame) const  
*Compute selected aspects of the derived state given the source state and the position/ attitude transformation from the derived to the source state.*

## Protected Attributes

- BaseDynManager \*& [dyn\\_manager](#)  
*The dynamics manager for the simulation.*
- const JeodIntegrationTime \* [time\\_manager](#)  
*The time manager to be used to obtain timestamp information.*
- [DynBody](#) \* [dyn\\_parent](#)  
*The [DynBody](#) to which this body is attached.*
- [DynBodyGenericFrameAttachment](#) [frame\\_attach](#)  
*The [RefFrame](#) this body is attached to.*
- std::list< [DynBody](#) \* > [dyn\\_children](#)  
*The subset of the dynamic bodies attached to this dynamic body.*
- std::list< [MassBody](#) \* > [mass\\_children](#)  
*The subset of the mass bodies attached to this dynamic body that are themselves not dynamic bodies.*
- std::list< [BodyRefFrame](#) \* > [vehicle\\_points](#)  
*An array of vehicle points associated with this dynamic body.*
- RefFrameItems [initialized\\_states](#)  
*Enum value indicating which of position, velocity, attitude, and rate have been initialized.*
- [BodyRefFrame](#) \* [position\\_source](#)  
*The reference frame that contains the user-set position.*
- [BodyRefFrame](#) \* [velocity\\_source](#)  
*The reference frame that contains the user-set velocity.*
- [BodyRefFrame](#) \* [attitude\\_source](#)  
*The reference frame that contains the user-set attitude.*
- [BodyRefFrame](#) \* [rate\\_source](#)  
*The reference frame that contains the user-set attitude rate.*

- [BodyRefFrame](#) \* [integrated\\_frame](#)  
*The reference frame whose state is updated via the state integrator.*
- `std::vector`  
< `er7_utils::IntegrableObject` \* > [associated\\_integrable\\_objects](#)  
*List of integrable objects to be integrated with this [DynBody](#).*
- `er7_utils::IntegratorResultMergerContainer` [integ\\_results\\_merger](#)  
*The object that merges integration results.*
- `RestartableT3SecondOrderODEIntegrator` [trans\\_integrator](#)  
*Translational state checkpointable/restartable integrator generator.*
- `RestartableSO3SecondOrderODEIntegrator` [rot\\_integrator](#)  
*Rotational state checkpointable/restartable integrator generator.*

## Private Member Functions

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

## Friends

- class [InputProcessor](#)
- void [init\\_attrjeod\\_\\_DynBody](#) ()

### 8.8.1 Detailed Description

Class [DynBody](#) is the base class for all dynamic bodies.

A [DynBody](#) is a `MassBody` that is connected to the outside world. These connections are in the form of three reference frames tied to the body – the structural, core body, and composite body frames.

For a non-root body, the states for each of these frames is calculated based on the parent body's state and on the body attachment.

For a root body, one of these three frames must be integrated. The details of how that integration is performed is the subject of classes that derive from [DynBody](#).

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

### 8.8.2 Constructor & Destructor Documentation

#### 8.8.2.1 `jeod::DynBody::DynBody ( )`

[DynBody](#) default constructor.

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

References `composite_body`, `core_body`, `initialized_states`, `integrated_frame`, `mass`, `jeod::BodyRefFrame::mass_point`, `rot_integrator`, `structure`, and `trans_integrator`.

#### 8.8.2.2 `jeod::DynBody::~~DynBody ( )` `[override]`

[DynBody](#) destructor.

Definition at line 113 of file `dyn_body.cc`.

References `composite_body`, `core_body`, `detach()`, `dyn_children`, `dyn_manager`, `dyn_parent`, `mass_children`, `remove_mass_body()`, `rot_integrator`, `structure`, `trans_integrator`, and `vehicle_points`.

#### 8.8.2.3 `jeod::DynBody::DynBody ( const DynBody & )` [private]

Not implemented.

### 8.8.3 Member Function Documentation

#### 8.8.3.1 `void jeod::DynBody::activate ( )` [inline]

Activate a [DynBody](#) object.

The current implementation does nothing. [DynBody](#) objects are always active.

Definition at line 151 of file `dyn_body.hh`.

#### 8.8.3.2 `void jeod::DynBody::add_control ( GravityControls * control )` [virtual]

Add a new GravityControls to the list in `grav_interaction`.

Parameters

<code>in</code>	<code>control</code>	Control to be added
-----------------	----------------------	---------------------

Definition at line 222 of file `dyn_body.cc`.

References `grav_interaction`.

#### 8.8.3.3 `void jeod::DynBody::add_integrable_object ( er7_utils::IntegrableObject & associated_integrable_object )`

Add an IntegrableObject to be integrated with this [DynBody](#).

Note that the associated IntegrableObject may or may not follow this [DynBody](#) if it is moved to a new integration group/loop.

Parameters

<code>in</code>	<code>associated_integrable_object</code>	The IntegrableObject to be associated with this <a href="#">DynBody</a> .
-----------------	---	---

Definition at line 290 of file `dyn_body.cc`.

References `associated_integrable_objects`.

#### 8.8.3.4 `bool jeod::DynBody::add_mass_body ( const char * this_point_name, const char * child_point_name, MassBody & child )` [virtual]

Definition at line 592 of file `dyn_body_attach.cc`.

References `find_vehicle_point()`, `jeod::DynBodyMessages::invalid_attachment`, `mass`, and `jeod::BodyRefFrame::mass_point`.

#### 8.8.3.5 `bool jeod::DynBody::add_mass_body ( const double offset[3], const double T_pstr_cstr[3][3], MassBody & child )` [virtual]

Definition at line 716 of file `dyn_body_attach.cc`.

References `add_mass_body_frames()`, `add_mass_body_validate()`, `mass`, `mass_children`, and `name`.

**8.8.3.6** void jeod::DynBody::add\_mass\_body\_frames ( MassBody & *subbody* ) [protected], [virtual]

For a newly attached mass sub-body, create body frames for the root sub-body and all child sub-bodies via recursion.

#### Returns

Validity indicator

#### Parameters

in	<i>subbody</i>	the root of the newly attached sub-bodies
----	----------------	---

Definition at line 803 of file dyn\_body\_attach.cc.

References dyn\_manager, integ\_frame, jeod::BodyRefFrame::mass\_point, name, and vehicle\_points.

Referenced by add\_mass\_body().

**8.8.3.7** bool jeod::DynBody::add\_mass\_body\_validate ( const MassBody & *child*, bool *generate\_message* ) const [protected], [virtual]

Validate whether the pending sub body is legal from a mass tree point of view.

#### Note

##### Assumptions and Limitations

- The subject mass, child, must not belong to a child body.

#### Returns

Validity indicator

#### Parameters

in	<i>child</i>	The child body; the body to be attached to this body.
in	<i>generate_message</i>	Generate message if invalid?

Definition at line 185 of file dyn\_body\_attach.cc.

References dyn\_manager, and name.

Referenced by add\_mass\_body().

**8.8.3.8** void jeod::DynBody::add\_mass\_point ( const MassPointInit & *mass\_point\_init* ) [virtual]

Add a mass point to the dyn body's list of such and make a vehicle point that corresponds to the added mass point.

#### Parameters

in	<i>mass_point_init</i>	Mass point specification
----	------------------------	--------------------------

Definition at line 53 of file dyn\_body\_vehicle\_point.cc.

References dyn\_manager, integ\_frame, jeod::DynBodyMessages::invalid\_body, mass, jeod::BodyRefFrame::mass\_point, name, and vehicle\_points.

**8.8.3.9** bool jeod::DynBody::attach\_child ( const char \* *this\_point\_name*, const char \* *child\_point\_name*, DynBody & *child* ) [virtual]

Attach a child [DynBody](#) by point specification.

See corresponding [DynBody::attach\\_to\(\)](#) method for more information.

Definition at line 373 of file `dyn_body_attach.cc`.

References `find_vehicle_point()`, `jeod::DynBodyMessages::invalid_attachment`, `mass`, and `jeod::BodyRefFrame::mass_point`.

Referenced by `attach_to()`.

**8.8.3.10** `bool jeod::DynBody::attach_child ( const double xyz_cstr_wrt_pstr[3], const double T_pstr_to_cstr[3][3], DynBody & child ) [virtual]`

Attach a child [DynBody](#) by location specification.

See corresponding [DynBody::attach\\_to\(\)](#) method for more information. Note that the offset and transformation are specified w.r.t. the parent in both [attach\\_to\(\)](#) and [attach\\_child\(\)](#)

Definition at line 511 of file `dyn_body_attach.cc`.

References `attach_establish_links()`, `attach_update_properties()`, `attach_validate_child()`, `attach_validate_parent()`, `get_root_body_internal()`, `mass`, and `name`.

**8.8.3.11** `void jeod::DynBody::attach_establish_links ( DynBody & parent ) [protected],[virtual]`

Establish the logical connectivity between parent and child.

Extensibility comments –

- This method is invoked before the computing the physical relation between parent and child.
- The generic purpose of this method is to establish the logical connectivity between parent and child in terms of the child class.
- Any class that overrides this method must either invoke this method or perform the actions performed herein.

#### Note

##### Assumptions and Limitations

- The attachment is valid; not checked.

#### Parameters

<code>in, out</code>	<code>parent</code>	The new parent body; the body to which this body is to be attached.
----------------------	---------------------	---

Definition at line 845 of file `dyn_body_attach.cc`.

References `dyn_children`, `dyn_parent`, `integ_frame`, `mass`, and `set_integ_frame()`.

Referenced by `attach_child()`.

**8.8.3.12** `bool jeod::DynBody::attach_to ( const char * this_point_name, const char * parent_point_name, DynBody & parent ) [virtual]`

Attach this dyn body's root body as a child of the specified dyn body such that the specified mass points on the two bodies are coincident and the frames associated with those mass points are related by a 180 degree yaw.

#### Returns

Success indicator: `true`=success, `false`=attachment not performed.

## Parameters

in	<i>this_point_name</i>	The name of a mass point contained in this dyn body's list of mass points.
in	<i>parent_point_name</i>	The name of a mass point contained in the parent body's list of mass points.
in, out	<i>parent</i>	The parent body; the body to which this body's root body is to be attached.

Definition at line 238 of file dyn\_body\_attach.cc.

References `attach_child()`.

**8.8.3.13** `bool jeod::DynBody::attach_to ( const double offset_pstr_cstr_pstr[3], const double T_pstr_cstr[3][3], DynBody & parent ) [virtual]`

Attach this dyn body's root body as a child of the specified dyn body such that this body's structural origin is offset from the parent body's structural origin and this body's structural axes are oriented with respect to the parent body's structural axes as specified.

## Returns

Success indicator: true=success, false=attachment not performed.

## Parameters

in	<i>offset_pstr_cstr_pstr</i>	Location of this body's structural origin with respect to the new parent body's structural origin, specified in structural coordinates of the parent body. Units: M
in	<i>T_pstr_cstr</i>	Transformation matrix from the parent body's structural frame to this body's structural frame.
in, out	<i>parent</i>	The parent body; the body to which this body's root body is to be attached.

Definition at line 257 of file dyn\_body\_attach.cc.

References `attach_child()`.

**8.8.3.14** `bool jeod::DynBody::attach_to_frame ( const char * parent_ref_frame_name ) [virtual]`

Definition at line 267 of file dyn\_body\_attach.cc.

References `dyn_manager`, `jeod::DynBodyMessages::invalid_attachment`, and `mass`.

**8.8.3.15** `bool jeod::DynBody::attach_to_frame ( RefFrame & parent ) [virtual]`

Definition at line 287 of file dyn\_body\_attach.cc.

References `frame_attach`, `get_root_body_internal()`, `jeod::DynBodyGenericFrameAttachment::initialize_attachment()`, and `structure`.

**8.8.3.16** `bool jeod::DynBody::attach_to_frame ( const char * this_point_name, const char * parent_ref_frame_name, const double offset_pframe_cpt_pframe[3], const double T_pframe_cpt[3][3] ) [virtual]`

Definition at line 296 of file dyn\_body\_attach.cc.

References `dyn_manager`, `find_vehicle_point()`, `frame_attach`, `get_root_body_internal()`, `jeod::DynBodyGenericFrameAttachment::initialize_attachment()`, `jeod::DynBodyMessages::invalid_attachment`, `mass`, `jeod::BodyRefFrame::mass_point`, and `structure`.

**8.8.3.17** `bool jeod::DynBody::attach_to_frame ( const double offset_pframe_cstr_pframe[3], const double T_pframe_cstr[3][3], RefFrame & parent ) [virtual]`

Definition at line 355 of file `dyn_body_attach.cc`.

References `frame_attach`, `get_root_body_internal()`, and `jeod::DynBodyGenericFrameAttachment::initialize_attachment()`.

**8.8.3.18** `void jeod::DynBody::attach_update_properties ( const double offset_pstr_cstr_pstr[3], const double T_pstr_cstr[3][3], DynBody & child ) [protected],[virtual]`

Set the relation between parent and child and update the mass properties.

Extensibility comments –

- This method is sent to the parent body of the attachment after the child body has established the logical connectivity between the parent body and child body.
- The generic purpose of this method is to establish the physical relation between parent and child and to update any physical properties that change as a result of the attachment.
- Any class that overrides this method must either invoke this method or perform the actions performed herein.

#### Note

##### Assumptions and Limitations

- The attachment is valid
- Logical connectivity has been established.

Neither assumption is checked.

#### Parameters

in	<i>offset_pstr_cstr_pstr</i>	Location of this body's structural origin with respect to the new parent body's structural origin, specified in structural coordinates of the new parent body. Units: m
in	<i>T_pstr_cstr</i>	Transformation matrix from the new parent body's structural frame to this body's structural frame.
in, out	<i>child</i>	The child body; the body newly attached to this body.

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

Definition at line 871 of file `dyn_body_attach.cc`.

References `get_dynamics_integration_group()`, `get_root_body_internal()`, `initialized_states`, `mass`, `process_dynamic_attachment()`, `propagate_state()`, `set_state_source_internal()`, and `structure`.

Referenced by `attach_child()`, and `jeod::StructureIntegratedDynBody::attach_update_properties()`.

**8.8.3.19** `bool jeod::DynBody::attach_validate_child ( const DynBody & child, bool generate_message ) const [protected],[virtual]`

Validate whether the pending attachment is legal from a physical point of view.

Extensibility comments –

- This method determines whether invoking `attach_update_properties` makes sense.



**Note**

## Assumptions and Limitations

- The subject body, child, must be a root body. This is not checked.

**Returns**

Validity indicator

**Parameters**

<i>in</i>	<i>child</i>	The child body; the body to be attached to this body.
<i>in</i>	<i>generate_ - message</i>	Generate message if invalid?

Definition at line 110 of file dyn\_body\_attach.cc.

References `get_root_body()`, `initialized_states`, `jeod::DynBodyMessages::invalid_attachment`, and `name`.

Referenced by `attach_child()`.

**8.8.3.20** `bool jeod::DynBody::attach_validate_parent ( const DynBody & parent, bool generate_message ) const`  
`[protected], [virtual]`

Validate whether the pending attachment is legal from a connectivity point of view.

Extensibility comments –

- This method determines whether invoking `attach_establish_links` makes sense.
- Any class that overrides this method must either invoke this method or perform the actions performed herein.

**Note**

## Assumptions and Limitations:

- The subject body, this, must be a root body. This is not checked.

**Returns**

Validity indicator

**Parameters**

<i>in</i>	<i>parent</i>	The new parent body; the body to which this body is to be attached.
<i>in</i>	<i>generate_ - message</i>	Generate message if invalid?

Definition at line 58 of file dyn\_body\_attach.cc.

References `dyn_manager`, `get_root_body()`, `jeod::DynBodyMessages::invalid_attachment`, `jeod::DynBodyMessages::invalid_body`, `name`, and `jeod::DynBodyMessages::not_dyn_body`.

Referenced by `attach_child()`.

**8.8.3.21** `void jeod::DynBody::clear_integrable_objects ( )`

Remove all IntegrableObjects associated with this [DynBody](#).

You might do this if you want to switch the [DynBody](#) to a new group without switching the associated Integrable-Objects.

Definition at line 330 of file dyn\_body.cc.

References `associated_integrable_objects`.

### 8.8.3.22 void jeod::DynBody::collect\_forces\_and\_torques ( ) [virtual]

Collect forces and torques acting on the vehicle.

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

Definition at line 98 of file `dyn_body_collect.cc`.

References `jeod::accumulate_forces()`, `jeod::accumulate_torques()`, `collect`, `jeod::BodyForceCollect::collect_effector_forc`, `jeod::BodyForceCollect::collect_effector_torq`, `jeod::BodyForceCollect::collect_environtorc`, `jeod::BodyForceCollect::collect_environtorq`, `collect_forces_and_torques()`, `jeod::BodyForceCollect::collect_no_xmit_forc`, `jeod::BodyForceCollect::collect_no_xmit_torq`, `composite_body`, `compute_point_derivative`, `compute_vehicle_point_derivatives()`, `derivs`, `dyn_children`, `dyn_parent`, `jeod::BodyForceCollect::effector_forc`, `jeod::BodyForceCollect::effector_torq`, `jeod::BodyForceCollect::environtorc`, `jeod::BodyForceCollect::environtorq`, `jeod::BodyForceCollect::extern_forc_inrtl`, `jeod::BodyForceCollect::extern_forc_struct`, `jeod::BodyForceCollect::extern_torq_body`, `jeod::BodyForceCollect::extern_torq_struct`, `grav_interaction`, `jeod::BodyForceCollect::inertial_torq`, `mass`, `jeod::BodyForceCollect::no_xmit_forc`, `jeod::BodyForceCollect::no_xmit_torq`, `jeod::FrameDerivs::non_grav_accel`, `jeod::FrameDerivs::rot_accel`, `rotational_dynamics`, `structure`, `jeod::FrameDerivs::trans_accel`, and `translational_dynamics`.

Referenced by `collect_forces_and_torques()`.

### 8.8.3.23 void jeod::DynBody::compute\_derived\_state\_forward ( const BodyRefFrame & source\_frame, const MassPoint & rel\_state, BodyRefFrame & derived\_frame ) const [protected]

Compute a derived state given the source state and the position/ attitude transformation from the source to the derived state.

Parameters

in	<i>source_frame</i>	Source state
in	<i>rel_state</i>	Relative state
out	<i>derived_frame</i>	Derived state

Definition at line 160 of file `dyn_body_propagate_state.cc`.

References `jeod::BodyRefFrame::initialized_items`.

Referenced by `compute_vehicle_point_states()`, `propagate_state_from_composite()`, and `propagate_state_from_structure()`.

### 8.8.3.24 void jeod::DynBody::compute\_derived\_state\_reverse ( const BodyRefFrame & source\_frame, const MassPoint & rel\_state, BodyRefFrame & derived\_frame ) const [protected]

Compute a derived state given the source state and the position/ attitude transformation from the derived to the source state.

Parameters

in	<i>source_frame</i>	Source state
in	<i>rel_state</i>	Relative state
out	<i>derived_frame</i>	Derived state

Definition at line 278 of file `dyn_body_propagate_state.cc`.

References `jeod::BodyRefFrame::initialized_items`.

Referenced by `propagate_state_from_composite()`.

### 8.8.3.25 void jeod::DynBody::compute\_ref\_point\_transform ( const BodyRefFrame & source\_frame, const MassPoint &&const ref\_point, MassPointState & rel\_state ) [protected]

Compute the relative state between the integrated frame's mass point and the source frame's mass point.

## Note

## Assumptions and Limitations

- This method is only called to be called for a root body. This assumption is not enforced.

## Parameters

in	<i>source_frame</i>	The frame that contains the relevant state data.
in, out	<i>ref_point</i>	The mass point corresponding to the previous call to this function. This is an efficiency hack used to avoid duplicative computations.
in, out	<i>rel_state</i>	The relative state between the integration frame mass point and the source frame mass point.

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

References `composite_body`, `core_body`, `integrated_frame`, `jeod::DynBodyMessages::invalid_frame`, `mass`, `jeod::BodyRefFrame::mass_point`, `name`, and `structure`.

Referenced by `update_integrated_state()`.

**8.8.3.26** `void jeod::DynBody::compute_state_elements_forward ( const BodyRefFrame & source_frame, const MassPoint & rel_state, const RefFrameItems & state_items, BodyRefFrame & derived_frame ) const` [protected]

Compute selected aspects of the derived state given the source state and the position/ attitude transformation from the source to the derived state.

## Parameters

in	<i>source_frame</i>	Source state
in	<i>rel_state</i>	Relative state
in	<i>state_items</i>	States to compute
out	<i>derived_frame</i>	Derived state

Definition at line 215 of file `dyn_body_propagate_state.cc`.

References `jeod::BodyRefFrame::initialized_items`.

Referenced by `compute_vehicle_point_states()`, `propagate_state_from_composite()`, and `propagate_state_from_structure()`.

**8.8.3.27** `void jeod::DynBody::compute_state_elements_reverse ( const BodyRefFrame & source_frame, const MassPoint & rel_state, const RefFrameItems & state_items, BodyRefFrame & derived_frame ) const` [protected]

Compute selected aspects of the derived state given the source state and the position/ attitude transformation from the derived to the source state.

## Parameters

in	<i>source_frame</i>	Source state
in	<i>rel_state</i>	Relative state
in	<i>state_items</i>	States to compute
out	<i>derived_frame</i>	Derived state

Definition at line 333 of file `dyn_body_propagate_state.cc`.

References `jeod::BodyRefFrame::initialized_items`.

Referenced by `propagate_state_from_composite()`.

**8.8.3.28** `void jeod::DynBody::compute_vehicle_point_derivatives ( const BodyRefFrame & vehicle_pt, FrameDerivs & pt_derivs )` [virtual]

Compute the state derivatives at a vehicle point.

## Parameters

in	<i>vehicle_pt</i>	Vehicle point reference frame
out	<i>pt_derivs</i>	Computed derivatives

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

Definition at line 129 of file `dyn_body_vehicle_point.cc`.

References `composite_body`, `derivs`, `get_root_body()`, `grav_interaction`, `jeod::DynBodyMessages::invalid_frame`, `mass`, `jeod::BodyRefFrame::mass_point`, `name`, `jeod::FrameDerivs::non_grav_accel`, `jeod::FrameDerivs::Qdot_parent_this`, `jeod::FrameDerivs::rot_accel`, and `jeod::FrameDerivs::trans_accel`.

Referenced by `collect_forces_and_torques()`.

**8.8.3.29** `void jeod::DynBody::compute_vehicle_point_states ( RefFrameItems::Items set_items ) [virtual]`

Propagate structure frame state to vehicle points.

## Parameters

in	<i>set_items</i>	States truly propagated
----	------------------	-------------------------

Definition at line 789 of file `dyn_body_propagate_state.cc`.

References `compute_derived_state_forward()`, `compute_state_elements_forward()`, `jeod::BodyRefFrame::mass_point`, `structure`, and `vehicle_points`.

Referenced by `propagate_state_from_composite()`, and `propagate_state_from_structure()`.

**8.8.3.30** `void jeod::DynBody::create_body_integrators ( const er7_utils::IntegratorConstructor & generator, er7_utils::IntegrationControls & controls, const JeodIntegrationTime & time_mgr ) [virtual]`

Create the integrator (integrators) needed to propagate the translational and rotational state of a [DynBody](#).

Create the translational and rotational integrators for a [DynBody](#).

## Parameters

in	<i>generator</i>	Integrator constructor to be used to create state integrators.
in	<i>controls</i>	The integration ontrls created the integrator constructor's <code>create_integration_controls</code> method.
in	<i>time_mgr</i>	The JEOD time manager object.

A [DynBody](#) integrates forces and torques in the body frame and forces induced by changes in mass properties.

## Parameters

in	<i>generator</i>	Integrator constructor to be used to create state integrators.
in	<i>controls</i>	The integration ontrls created the integrator constructor's <code>create_integration_controls</code> method.
in	<i>time_mgr</i>	The JEOD time manager object.

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

References `integ_results_merger`, `name`, `rot_integrator`, `rotation_integration`, `three_dof`, `time_manager`, and `trans_integrator`.

Referenced by `create_integrators()`.

**8.8.3.31** `void jeod::DynBody::create_integrators ( const er7_utils::IntegratorConstructor & generator, er7_utils::IntegrationControls & controls, const er7_utils::TimeInterface & time_if ) [override]`

This interface is required by `er7_utils::IntegrableObject`.

It should not be used. Use [DynBody::create\\_body\\_integrators](#) instead.

**Parameters**

in	<i>generator</i>	Unused.
in	<i>controls</i>	Unused.
in	<i>time_if</i>	Unused.

Definition at line 259 of file dyn\_body\_integration.cc.

References create\_body\_integrators(), and jeod::DynBodyMessages::internal\_error.

**8.8.3.32 void jeod::DynBody::deactivate ( ) [inline]**

Deactivate a [DynBody](#) object.

The current implementation does nothing. [DynBody](#) objects are always active.

Definition at line 159 of file dyn\_body.hh.

**8.8.3.33 void jeod::DynBody::destroy\_integrators ( void ) [override]**

Destroy the integrators.

Does nothing, but must be implemented to complete abstract function from the inherited IntegrableObject

Definition at line 285 of file dyn\_body\_integration.cc.

**8.8.3.34 bool jeod::DynBody::detach ( DynBody & other\_body ) [virtual]**

Detach parent and child DynBodies, 'this' and the argument body, such that the detachment happens at the parent body level.

Returns true if successfully detached the bodies. Returns false if unable to detach. Will fail if, for example, the bodies are not in the same mass tree.

**Assumptions and Limitations**

- The detach point between non-immediate attachments (i.e. not parent/child attachments) takes place at whichever body is a progenitor. For example, a call to A.detach(D) in an A->B->C->D attachment is interpreted as a call desiring A // B->C->D. A call to D.detach(B) is interpreted as a call to A->B // C->D.

**Returns**

Success flag

**Parameters**

in	<i>other_body</i>	The other body at which the detach will occur
----	-------------------	---

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

Definition at line 50 of file dyn\_body\_detach.cc.

References detach\_mass\_internal(), dyn\_children, dyn\_parent, jeod::DynBodyMessages::invalid\_attachment, mass, and name.

Referenced by ~DynBody().

**8.8.3.35 bool jeod::DynBody::detach ( void ) [virtual]**

Detach this [DynBody](#) from its parent RefFrame or [DynBody](#) parent.

If detaching from a [DynBody](#), evoking this method is the equivalent to the above function via detach(\*dyn\_parent)

## Assumptions and Limitations

- Will inform and return false if the body has no parent.

## Returns

Success flag

Definition at line 138 of file dyn\_body\_detach.cc.

References jeod::DynBodyGenericFrameAttachment::clear\_attachment(), dyn\_parent, frame\_attach, jeod::DynBodyMessages::invalid\_technique, jeod::DynBodyGenericFrameAttachment::isAttached(), and name.

Referenced by jeod::StructureIntegratedDynBody::detach(), remove\_mass\_body(), and ~DynBody().

**8.8.3.36** void jeod::DynBody::detach\_mass\_body\_frames ( MassBody & *subbody* ) [protected],[virtual]

For a newly detached mass sub-body, remove body frames for the root sub-body and all child sub-bodies via recursion.

## Returns

Validity indicator

## Parameters

in	<i>subbody</i>	the root of the newly attached sub-bodies
----	----------------	---

Definition at line 239 of file dyn\_body\_detach.cc.

References dyn\_manager, find\_body\_frame(), and vehicle\_points.

Referenced by remove\_mass\_body().

**8.8.3.37** void jeod::DynBody::detach\_mass\_internal ( MassBody & *child* ) [protected],[virtual]

Update parent and child properties to reflect that they are detached.

Extensibility comments –

- This method is sent to the parent body of the detachment after the child body has severed the logical connectivity between the parent body and child body.
- The generic purpose of this method is to update any physical properties that change as a result of the detachment.
- Any class that overrides this method must either invoke this method or perform the actions performed herein.

## Note

## Assumptions and Limitations

- The detachment is valid and logical connectivity has been severed. Neither assumption is checked.

## Parameters

in, out	<i>child</i>	The child body; the body newly detached from this body.
---------	--------------	---

Definition at line 285 of file dyn\_body\_detach.cc.

References core\_body, get\_root\_body\_internal(), mass, propagate\_state(), and set\_state\_source\_internal().

Referenced by detach(), and remove\_mass\_body().

### 8.8.3.38 **BodyRefFrame** \* jeod::DynBody::find\_body\_frame ( const char \* *frame\_id* ) const [virtual]

Find the [BodyRefFrame](#) named by the provided identifier.

The name of a [BodyRefFrame](#) must be prefixed by the body name. The provided identifier can include or exclude this prefix. The body name is used as the prefix if the the provided name does not start with the body name.

#### Note

##### Assumptions and Limitations

- Limitation: Provided identifier must be non-NULL and non-empty. Failure to comply is a fatal error.
- Limitation: The found frame must be a [BodyRefFrame](#). Finding a non-BodyRefFrame that matches the name is a fatal error.
- Assumption: Failure to find a frame is not an error. The method returns NULL if this is the case.

#### Returns

Found frame

#### Parameters

in	<i>frame_id</i>	Frame ID suffix
----	-----------------	-----------------

Definition at line 50 of file dyn\_body\_find\_body\_frame.cc.

References [dyn\\_manager](#), [jeod::DynBodyMessages::invalid\\_name](#), and [name](#).

Referenced by [detach\\_mass\\_body\\_frames\(\)](#).

### 8.8.3.39 const **BodyRefFrame** \* jeod::DynBody::find\_vehicle\_point ( const char \* *pt\_name* ) const

Find the vehicle point with the given name.

#### Returns

Vehicle point

#### Parameters

in	<i>pt_name</i>	Vehicle point name
----	----------------	--------------------

Definition at line 101 of file dyn\_body\_vehicle\_point.cc.

References [name](#), and [vehicle\\_points](#).

Referenced by [add\\_mass\\_body\(\)](#), [attach\\_child\(\)](#), and [attach\\_to\\_frame\(\)](#).

### 8.8.3.40 **DynamicsIntegrationGroup** \* jeod::DynBody::get\_dynamics\_integration\_group ( )

Get the [DynamicsIntegrationGroup](#) that integrates this [DynBody](#) object.

#### Returns

Pointer to the [DynamicsIntegrationGroup](#) of this [DynBody](#).

Definition at line 261 of file dyn\_body.cc.

References [jeod::DynBodyMessages::internal\\_error](#).

Referenced by [attach\\_update\\_properties\(\)](#), and [set\\_integ\\_frame\(\)](#).



**8.8.3.41** `const RefFrameItems& jeod::DynBody::get_initialized_states ( ) const [inline]`

Indicate which state elements have been initialized.

#### Returns

Initialized states indicator.

Definition at line 528 of file `dyn_body.hh`.

References `initialized_states`.

**8.8.3.42** `JeodPointerVector<er7_utils::IntegrableObject>::type jeod::DynBody::get_integrable_objects ( ) [inline]`

Get the IntegrableObjects associated with this [DynBody](#).

#### Returns

A pointer to a JeodPointerVector containing the associated integrable objects.

Definition at line 308 of file `dyn_body.hh`.

References `associated_integrable_objects`.

**8.8.3.43** `const DynBody * jeod::DynBody::get_parent_body ( ) const [virtual]`

Returns this [DynBody](#) object's parent body.

#### Returns

Const pointer to the parent body.

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

References `dyn_parent`.

Referenced by `jeod::StructureIntegratedDynBody::detach()`.

**8.8.3.44** `DynBody * jeod::DynBody::get_parent_body_internal ( ) [protected],[virtual]`

Returns this [DynBody](#) object's parent body.

#### Returns

Pointer to parent body.

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

References `dyn_parent`.

**8.8.3.45** `const DynBody * jeod::DynBody::get_root_body ( ) const [virtual]`

Finds this [DynBody](#) object's root body.

#### Returns

Const pointer to the root body.

Definition at line 194 of file `dyn_body.cc`.

Referenced by `attach_validate_child()`, `attach_validate_parent()`, `jeod::StructureIntegratedDynBody::compute_vehicle_point_derivatives()`, `compute_vehicle_point_derivatives()`, and `set_state_source()`.

#### 8.8.3.46 `DynBody * jeod::DynBody::get_root_body_internal ( ) [protected],[virtual]`

Finds this [DynBody](#) object's root body.

##### Returns

Pointer to the root body.

Definition at line 205 of file `dyn_body.cc`.

References `dyn_parent`.

Referenced by `attach_child()`, `attach_to_frame()`, `attach_update_properties()`, `detach_mass_internal()`, `set_attitude_left_quaternion()`, `set_attitude_matrix()`, `set_attitude_rate()`, `set_attitude_right_quaternion()`, `set_position()`, `set_state()`, `set_state_source()`, `set_velocity()`, and `update_integrated_state()`.

#### 8.8.3.47 `void jeod::DynBody::initialize_controls ( GravityManager & grav_manager ) [virtual]`

Initialize the gravity controls of this [DynBody](#).

##### Note

Initialization phasing:

The following must have been called prior to calling this method:

- `GravityManager::initialize_model` to register the `GravityManager` object with the dynamics manager.
- `GravityManager::add_grav_source` to register the pertinent `GravitySource` objects with the `Gravity Manager`.
- `Planet::register_model` to associate the planet with a `GravitySource`.

##### Parameters

in	<i>grav_manager</i>	Reference to Gravity Manager
----	---------------------	------------------------------

Definition at line 232 of file `dyn_body.cc`.

References `dyn_manager`, and `grav_interaction`.

#### 8.8.3.48 `void jeod::DynBody::initialize_model ( BaseDynManager & dyn_manager_in ) [virtual]`

Initialize internal and external interrelations, including registration / with the dynamics manager.

##### Parameters

in, out	<i>dyn_manager_in</i>	Dynamics manager
---------	-----------------------	------------------

Definition at line 45 of file `dyn_body_initialize_model.cc`.

References `composite_body`, `core_body`, `dyn_manager`, `initialized_states`, `integ_frame`, `integ_frame_name`, `jeod::DynBodyMessages::invalid_frame`, `jeod::DynBodyMessages::invalid_name`, `mass`, `name`, `set_integ_frame()`, and `structure`.

#### 8.8.3.49 `bool jeod::DynBody::initialized_states_contains ( RefFrameItems::Items test_items ) const [inline]`

Indicate whether the specified state elements have been initialized.

## Parameters

<i>test_items</i>	States to test.
-------------------	-----------------

## Returns

True if all test items have been initialized, false otherwise.

Definition at line 538 of file dyn\_body.hh.

References initialized\_states.

### 8.8.3.50 er7\_utils::IntegratorResult jeod::DynBody::integrate ( double *dyn\_dt*, unsigned int *target\_stage* ) [override]

Integrate state by the specified dynamic time interval.

Integrate the translational and rotational state and propagate the integrated state to derived states.

## Parameters

in	<i>dyn_dt</i>	Dynamic time step, in dynamic time seconds.
in	<i>target_stage</i>	The stage of the integration process that the integrator should try to attain.

## Returns

The status (time advance, pass/fail status) of the integration.

Definition at line 316 of file dyn\_body\_integration.cc.

References frame\_attach, jeod::DynBodyGenericFrameAttachment::get\_attach\_offset(), jeod::DynBodyGenericFrameAttachment::get\_parent\_frame(), initialized\_states, integ\_frame, integ\_results\_merger, jeod::DynBodyGenericFrameAttachment::isAttached(), propagate\_state(), rot\_integ(), rotational\_dynamics, set\_state(), structure, trans\_integ(), and translational\_dynamics.

### 8.8.3.51 bool jeod::DynBody::is\_root\_body ( )

Indicates whether this [DynBody](#) object is a root body.

## Returns

Is this a root body?

Definition at line 169 of file dyn\_body.cc.

References dyn\_parent.

### 8.8.3.52 void jeod::DynBody::migrate\_integrable\_objects ( void )

Call this method before switching this dyn body to a new group if you want the associated integrable objects to follow.

Definition at line 337 of file dyn\_body.cc.

References associated\_integrable\_objects, jeod::DynBodyMessages::invalid\_group, and name.

### 8.8.3.53 DynBody& jeod::DynBody::operator= ( const DynBody & ) [private]

Not implemented.

**8.8.3.54** `void jeod::DynBody::process_dynamic_attachment ( const double offset_pstr_cstr_pstr[3], const double T_pstr_cstr[3][3], DynBody & root_body, DynBody & child_body )` `[protected]`, `[virtual]`

Process the attachment event of one body from another.

This method is called by the attach method after the links have established or severed and is invoked twice:

- On the parent, in which case the parent argument is null and the child argument is the child that attached from the parent, and
- On the detaching child, in which case the child argument is null and the parent argument is the body from which the child was detached.

#### Note

Assumptions and Limitations:

- Instances of more derived classes, with presumably more involved dynamics, are situated higher in the mass tree than are more basic instances. For example, a simple MassBody can be a child of a [DynBody](#), but not the other way around.
- The attachment in the mass tree between the immediate child and the superior body is assumed to reflect a real physical attachment.

#### Parameters

in	<i>offset_pstr_cstr_pstr</i>	Location of this body's structural origin with respect to the new parent body's structural origin, specified in structural coordinates of the new parent body. Units: m
in	<i>T_pstr_cstr</i>	Transformation matrix from the new parent body's structural frame to this body's structural frame.
in, out	<i>root_body</i>	Body at the root of the mass tree
in, out	<i>child_body</i>	Body that is being attached to this body.

Definition at line 953 of file `dyn_body_attach.cc`.

References `composite_body`, `core_body`, `mass`, `propagate_state()`, `set_state_source_internal()`, and `structure`.

Referenced by `attach_update_properties()`.

**8.8.3.55** `void jeod::DynBody::propagate_state ( )` `[virtual]`

Propagate state from the integrated state to attached bodies.

Definition at line 574 of file `dyn_body_propagate_state.cc`.

References `composite_body`, `dyn_parent`, `initialized_states`, `integrated_frame`, `jeod::DynBodyMessages::invalid_frame`, `name`, `propagate_state()`, `propagate_state_from_composite()`, `propagate_state_from_structure()`, `structure`, and `update_integrated_state()`.

Referenced by `attach_update_properties()`, `detach_mass_internal()`, `integrate()`, `process_dynamic_attachment()`, `propagate_state()`, and `switch_integration_frames()`.

**8.8.3.56** `void jeod::DynBody::propagate_state_from_composite ( )` `[protected]`, `[virtual]`

Propagate state to attached bodies starting from this body's composite frame.

#### Note

Assumptions and Limitations

- At least some states are set.

Definition at line 700 of file dyn\_body\_propagate\_state.cc.

References `autoupdate_vehicle_points`, `composite_body`, `compute_derived_state_forward()`, `compute_derived_state_reverse()`, `compute_state_elements_forward()`, `compute_state_elements_reverse()`, `compute_vehicle_point_states()`, `core_body`, `dyn_children`, `jeod::BodyRefFrame::initialized_items`, `initialized_states`, `mass`, `propagate_state_from_composite()`, `propagate_state_from_structure()`, and `structure`.

Referenced by `propagate_state()`, and `propagate_state_from_composite()`.

#### 8.8.3.57 void jeod::DynBody::propagate\_state\_from\_structure ( ) [protected],[virtual]

Propagate state to attached bodies starting from this body's structural frame.

##### Note

##### Assumptions and Limitations

- At least some states are set.

Definition at line 608 of file dyn\_body\_propagate\_state.cc.

References `autoupdate_vehicle_points`, `composite_body`, `compute_derived_state_forward()`, `compute_state_elements_forward()`, `compute_vehicle_point_states()`, `core_body`, `dyn_children`, `jeod::BodyRefFrame::initialized_items`, `initialized_states`, `mass`, `propagate_state_from_structure()`, and `structure`.

Referenced by `propagate_state()`, `propagate_state_from_composite()`, and `propagate_state_from_structure()`.

#### 8.8.3.58 void jeod::DynBody::remove\_integrable\_object ( er7\_utils::IntegrableObject & associated\_integrable\_object )

Remove an IntegrableObject from association with this [DynBody](#).

##### Parameters

in	<i>associated_integrable_object</i>	The IntegrableObject to be associated with this <a href="#">DynBody</a> .
----	-------------------------------------	---

Definition at line 309 of file dyn\_body.cc.

References `associated_integrable_objects`.

#### 8.8.3.59 bool jeod::DynBody::remove\_mass\_body ( MassBody & child ) [virtual]

Remove connectivity between this (parent) [DynBody](#) and the argument (child) `MassBody` mass subbody.

The `MassBody` and associated body frames are removed, such that the `MassBody` effectively "jettisons" from dynamics operations.

Extensibility comments –

- This method is invoked before the updating the parent/child states.
- The generic purpose of this method is to sever all connectivity links between parent and child, most importantly mass properties.
- Any class that overrides this method must either invoke this method or perform the actions performed herein.

##### Note

##### Assumptions and Limitations

- The detachment must be valid or it is not performed. The `MassBody` must not belong to a `DynBody`-derived dynamic body.

**Parameters**

<i>in, out</i>	<i>child</i>	The child mass subbody; the body to be detached
----------------	--------------	---

Definition at line 166 of file `dyn_body_detach.cc`.

References `detach()`, `detach_mass_body_frames()`, `detach_mass_internal()`, `jeod::DynBodyMessages::invalid_technique`, `mass`, `mass_children`, and `name`.

Referenced by `~DynBody()`.

#### 8.8.3.60 `void jeod::DynBody::reset_controls ( ) [virtual]`

Make the frame subscriptions for each control consistent with the requirements for that control.

Definition at line 243 of file `dyn_body.cc`.

References `dyn_manager`, and `grav_interaction`.

#### 8.8.3.61 `void jeod::DynBody::reset_integrators ( void ) [override]`

Reset the translational and rotational integrators.

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

References `rot_integrator`, `rotational_dynamics`, `trans_integrator`, and `translational_dynamics`.

#### 8.8.3.62 `er7_utils::IntegratorResult jeod::DynBody::rot_integ ( double dyn_dt, unsigned int target_stage ) [protected], [virtual]`

Integrate the vehicle's rotational state.

Integrate the rotational state of a [DynBody](#).

**Parameters**

<i>in</i>	<i>target_stage</i>	The stage of the integration process that the integrator should try to attain.
-----------	---------------------	--

**Returns**

The status (time advance, pass/fail status) of the integration.

**Parameters**

<i>in</i>	<i>dyn_dt</i>	Dynamic time step, in dynamic time seconds.
<i>in</i>	<i>target_stage</i>	The stage of the integration process that the integrator should try to attain.

**Returns**

The status (time advance, pass/fail status) of the integration.

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

Definition at line 388 of file `dyn_body_integration.cc`.

References `composite_body`, `derivs`, `jeod::FrameDerivs::Qdot_parent_this`, `jeod::FrameDerivs::rot_accel`, and `rot_integrator`.

Referenced by `integrate()`.

#### 8.8.3.63 `void jeod::DynBody::set_attitude_left_quaternion ( const Quaternion & left_quat, BodyRefFrame & subject_frame )`

Set the attitude of the vehicle.

**Note**

## Assumptions and Limitations

- Provided quaternion is a unit quaternion.

**Parameters**

in	<i>left_quat</i>	Attitude wrt integ frame
out	<i>subject_frame</i>	Frame to update

Definition at line 218 of file `dyn_body_set_state.cc`.

References `jeod::check_frame_ownership()`, `get_root_body_internal()`, and `set_state_source_internal()`.

**8.8.3.64** `void jeod::DynBody::set_attitude_matrix ( const double matrix[3][3], BodyRefFrame & subject_frame )`

Set the attitude of the vehicle.

**Note**

## Assumptions and Limitations

- Provided matrix is orthogonal.

**Parameters**

in	<i>matrix</i>	Attitude wrt integ frame
out	<i>subject_frame</i>	Frame to update

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

References `jeod::check_frame_ownership()`, `get_root_body_internal()`, and `set_state_source_internal()`.

**8.8.3.65** `void jeod::DynBody::set_attitude_rate ( const double attitude_rate[3], BodyRefFrame & subject_frame )`

Set the attitude rate of the vehicle.

**Note**

## Assumptions and Limitations

- Provided vector is expressed in body frame coordinates.

**Parameters**

in	<i>attitude_rate</i>	Attitude wrt integ frame Units: r/s
out	<i>subject_frame</i>	Frame to update

Definition at line 275 of file `dyn_body_set_state.cc`.

References `jeod::check_frame_ownership()`, `get_root_body_internal()`, and `set_state_source_internal()`.

**8.8.3.66** `void jeod::DynBody::set_attitude_right_quaternion ( const Quaternion & right_quat, BodyRefFrame & subject_frame )`

Set the attitude of the vehicle.

**Note**

## Assumptions and Limitations

- Provided quaternion is a unit quaternion.

**Parameters**

in	<i>right_quat</i>	Attitude wrt integ frame
out	<i>subject_frame</i>	Frame to update

Definition at line 237 of file `dyn_body_set_state.cc`.

References `jeod::check_frame_ownership()`, `get_root_body_internal()`, and `set_state_source_internal()`.

**8.8.3.67** `void jeod::DynBody::set_integ_frame ( EphemerisRefFrame & new_integ_frame )` `[protected]`, `[virtual]`

Set the integration frame for this body and all its child bodies to the provided frame.

**Note****Assumptions and Limitations**

- Provided frame is a valid integration frame.

**Parameters**

in	<i>new_integ - frame</i>	New integration frame
----	--------------------------	-----------------------

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

References `composite_body`, `core_body`, `dyn_children`, `dyn_manager`, `get_dynamics_integration_group()`, `grav_interaction`, `integ_frame`, `set_integ_frame()`, `structure`, and `vehicle_points`.

Referenced by `attach_establish_links()`, `initialize_model()`, `set_integ_frame()`, and `switch_integration_frames()`.

**8.8.3.68** `void jeod::DynBody::set_integ_frame ( const char * new_integ_frame_name )` `[protected]`, `[virtual]`

Set the integration frame for this body and all its child bodies to the frame indicated by the provided name.

**Note****Assumptions and Limitations**

- Assumption: Provided string is a non-NULL, non-empty string.
- Assumption: State is not to be updated.
- Limitation: Associated frame must be a valid integration frame.

**Parameters**

in	<i>new_integ - frame_name</i>	New integration frame
----	-------------------------------	-----------------------

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

References `dyn_manager`, `jeod::DynBodyMessages::invalid_name`, `name`, and `set_integ_frame()`.

**8.8.3.69** `void jeod::DynBody::set_name ( const std::string & name_in )`

Set the name of the vehicle.

**Parameters**



in	<i>name_in</i>	Name of this body
----	----------------	-------------------

Definition at line 161 of file dyn\_body.cc.

References mass.

#### 8.8.3.70 void jeod::DynBody::set\_position ( const double *position*[3], BodyRefFrame & *subject\_frame* )

Set the position of the vehicle.

##### Parameters

in	<i>position</i>	Position wrt integ frame Units: M
out	<i>subject_frame</i>	Frame to update

Definition at line 184 of file dyn\_body\_set\_state.cc.

References jeod::check\_frame\_ownership(), get\_root\_body\_internal(), and set\_state\_source\_internal().

#### 8.8.3.71 void jeod::DynBody::set\_state ( RefFrameItems::Items *set\_items*, const RefFrameState & *state*, BodyRefFrame & *subject\_frame* )

Set the parts of the specified reference frame as indicated by the *set\_items* parameter from the supplied state and propagate these items to all dynamic bodies attached to this body.

This method forms an integral part of the state initialization process and can also be used by a simulation that receives state overrides from some other simulation.

##### Note

##### Assumptions and Limitations

- The subject reference frame is owned by this dynamic body. This limitation is enforced.

##### Parameters

in	<i>set_items</i>	Items to set
in	<i>state</i>	State to be copied
out	<i>subject_frame</i>	Frame to be set

Definition at line 79 of file dyn\_body\_set\_state.cc.

References jeod::check\_frame\_ownership(), get\_root\_body\_internal(), and set\_state\_source\_internal().

Referenced by integrate().

#### 8.8.3.72 void jeod::DynBody::set\_state\_source ( RefFrameItems::Items *items*, BodyRefFrame & *frame* )

Set the source of aspects of the state.

The setting is applied to the root of the [DynBody](#) tree.

##### Note

##### Assumptions and Limitations

- The supplied frame must either be owned directly by this body or this body must be a root body and the owner of the supplied frame must be a child body of this body.

**Parameters**

in	<i>items</i>	Items to propagate
in	<i>frame</i>	Frame containing state

Definition at line 132 of file dyn\_body\_set\_state.cc.

References dyn\_parent, get\_root\_body(), get\_root\_body\_internal(), jeod::DynBodyMessages::invalid\_frame, name, and set\_state\_source\_internal().

**8.8.3.73** void jeod::DynBody::set\_state\_source\_internal ( RefFrameItems::Items *items*, BodyRefFrame & *frame* )  
[protected]

Set the source of aspects of the state.

**Note****Assumptions and Limitations**

- Assumptions, neither of which is checked:
  - This is a root body.
  - The supplied frame is owned by a body that is a child of this body.

**Parameters**

in	<i>items</i>	Items to propagate
in	<i>frame</i>	Frame containing state

Definition at line 293 of file dyn\_body\_set\_state.cc.

References attitude\_source, jeod::BodyRefFrame::initialized\_items, initialized\_states, position\_source, rate\_source, and velocity\_source.

Referenced by attach\_update\_properties(), detach\_mass\_internal(), process\_dynamic\_attachment(), set\_attitude\_left\_quaternion(), set\_attitude\_matrix(), set\_attitude\_rate(), set\_attitude\_right\_quaternion(), set\_position(), set\_state(), set\_state\_source(), and set\_velocity().

**8.8.3.74** void jeod::DynBody::set\_velocity ( const double *velocity*[3], BodyRefFrame & *subject\_frame* )

Set the velocity of the vehicle.

**Parameters**

in	<i>velocity</i>	Velocity wrt integ frame Units: M/s
out	<i>subject_frame</i>	Frame to update

Definition at line 201 of file dyn\_body\_set\_state.cc.

References jeod::check\_frame\_ownership(), get\_root\_body\_internal(), and set\_state\_source\_internal().

**8.8.3.75** void jeod::DynBody::sort\_controls ( ) [virtual]

Sort the gravity controls in ascending acceleration magnitude order.

Definition at line 252 of file dyn\_body.cc.

References grav\_interaction.

**8.8.3.76** void jeod::DynBody::switch\_integration\_frames ( EphemerisRefFrame & *new\_integ\_frame* ) [virtual]

Switch the integration frame for this body and all its child bodies to the indicated frame.

## Note

## Assumptions and Limitations

- Limitation: Associated frame must be a valid integration frame.

## Parameters

<i>in</i>	<i>new_integ_ - frame</i>	New integration frame
-----------	-------------------------------	-----------------------

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

References `dyn_manager`, `dyn_parent`, `integrated_frame`, `jeod::DynBodyMessages::invalid_frame`, `name`, `propagate_state()`, `set_integ_frame()`, `switch_integration_frames()`, and `update_integrated_state()`.

Referenced by `switch_integration_frames()`.

**8.8.3.77** `void jeod::DynBody::switch_integration_frames ( const char * new_integ_frame_name ) [virtual]`

Switch the integration frame for this body and all its child bodies to the frame indicated by the provided name.

## Note

## Assumptions and Limitations

- Assumption: Provided string is a non-NULL, non-empty string.
- Limitation: Associated frame must be a valid integration frame.

## Parameters

<i>in</i>	<i>new_integ_ - frame_name</i>	New integration frame
-----------	------------------------------------	-----------------------

Definition at line 190 of file `dyn_body_integration.cc`.

References `dyn_manager`, `jeod::DynBodyMessages::invalid_name`, `name`, and `switch_integration_frames()`.

**8.8.3.78** `er7_utils::IntegratorResult jeod::DynBody::trans_integ ( double dyn_dt, unsigned int target_stage ) [protected], [virtual]`

Integrate the vehicle's translational state.

Integrate the translational state of a [DynBody](#).

## Parameters

<i>in</i>	<i>target_stage</i>	The stage of the integration process that the integrator should try to attain.
-----------	---------------------	--

## Returns

The status (time advance, pass/fail status) of the integration.

## Parameters

<i>in</i>	<i>dyn_dt</i>	Dynamic time step, in dynamic time seconds.
<i>in</i>	<i>target_stage</i>	The stage of the integration process that the integrator should try to attain.

**Returns**

The status (time advance, pass/fail status) of the integration.

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

Definition at line 368 of file `dyn_body_integration.cc`.

References `composite_body`, `derivs`, `jeod::FrameDerivs::trans_accel`, and `trans_integrator`.

Referenced by `integrate()`.

**8.8.3.79 void jeod::DynBody::update\_integrated\_state ( ) [virtual]**

Propagate state from state owners to the integrated state.

Definition at line 396 of file `dyn_body_propagate_state.cc`.

References `attitude_source`, `compute_ref_point_transform()`, `dyn_parent`, `get_root_body_internal()`, `jeod::BodyRefFrame::initialized_items`, `initialized_states`, `integrated_frame`, `position_source`, `rate_source`, `time_manager`, `update_integrated_state()`, and `velocity_source`.

Referenced by `propagate_state()`, `switch_integration_frames()`, and `update_integrated_state()`.

**8.8.4 Friends And Related Function Documentation****8.8.4.1 void init\_attrjeod\_\_DynBody ( ) [friend]****8.8.4.2 friend class InputProcessor [friend]**

Definition at line 116 of file `dyn_body.hh`.

**8.8.5 Field Documentation****8.8.5.1 std::vector<er7\_utils::IntegrableObject\*> jeod::DynBody::associated\_integrable\_objects [protected]**

List of integrable objects to be integrated with this [DynBody](#).

`trick_io(**)`

Definition at line 1225 of file `dyn_body.hh`.

Referenced by `add_integrable_object()`, `clear_integrable_objects()`, `get_integrable_objects()`, `migrate_integrable_objects()`, and `remove_integrable_object()`.

**8.8.5.2 BodyRefFrame\* jeod::DynBody::attitude\_source [protected]**

The reference frame that contains the user-set attitude.

`trick_units(-)`

Definition at line 1207 of file `dyn_body.hh`.

Referenced by `set_state_source_internal()`, and `update_integrated_state()`.

**8.8.5.3 bool jeod::DynBody::autoupdate\_vehicle\_points**

Are vehicle points automatically updated? The vehicle points are automatically calculated at initialization time but are only automatically updated at runtime if this member is true.

Setting this member to false indicates the responsibility for updating vehicle point states is performed elsewhere, such as in a scheduled call to `compute_vehicle_point_states.trick_units(-)`

Definition at line 762 of file dyn\_body.hh.

Referenced by propagate\_state\_from\_composite(), and propagate\_state\_from\_structure().

#### 8.8.5.4 BodyForceCollect jeod::DynBody::collect

Force/Torque collection mechanism.

trick\_units(–)

Definition at line 781 of file dyn\_body.hh.

Referenced by jeod::StructureIntegratedDynBody::collect\_forces\_and\_torques(), collect\_forces\_and\_torques(), jeod::StructureIntegratedDynBody::collect\_local\_forces\_and\_torques(), jeod::StructureIntegratedDynBody::compute\_inertial\_torque(), jeod::StructureIntegratedDynBody::compute\_rotational\_acceleration(), jeod::StructureIntegratedDynBody::compute\_translational\_acceleration(), jeod::StructureIntegratedDynBody::PropagateForcesAndTorques(), and jeod::StructureIntegratedDynBody::solve\_constraints().

#### 8.8.5.5 BodyRefFrame jeod::DynBody::composite\_body

Vehicle composite body reference frame.

The reference frame origin is at the composite body center of mass, and the reference frame axes are the body frame axes as defined in the composite mass properties.trick\_units(–)

Definition at line 697 of file dyn\_body.hh.

Referenced by collect\_forces\_and\_torques(), compute\_ref\_point\_transform(), jeod::StructureIntegratedDynBody::compute\_vehicle\_point\_derivatives(), compute\_vehicle\_point\_derivatives(), DynBody(), initialize\_model(), process\_dynamic\_attachment(), propagate\_state(), propagate\_state\_from\_composite(), propagate\_state\_from\_structure(), jeod::StructureIntegratedDynBody::PropagateForcesAndTorques(), rot\_integ(), set\_integ\_frame(), jeod::StructureIntegratedDynBody::solve\_constraints(), trans\_integ(), and ~DynBody().

#### 8.8.5.6 bool jeod::DynBody::compute\_point\_derivative

Should the point derivatives for the body be computed? A child body's translational and rotational derivatives are only computed if this is true.

If this is false, they will be 0.trick\_units(–)

Definition at line 731 of file dyn\_body.hh.

Referenced by collect\_forces\_and\_torques().

#### 8.8.5.7 BodyRefFrame jeod::DynBody::core\_body

Vehicle core body reference frame.

The reference frame origin is at the core body center of mass, and the reference frame axes are the body frame axes as defined in the core mass properties.trick\_units(–)

Definition at line 689 of file dyn\_body.hh.

Referenced by compute\_ref\_point\_transform(), detach\_mass\_internal(), DynBody(), initialize\_model(), process\_dynamic\_attachment(), propagate\_state\_from\_composite(), propagate\_state\_from\_structure(), set\_integ\_frame(), and ~DynBody().

#### 8.8.5.8 FrameDerivs jeod::DynBody::derivs

Translational/rotational accelerations.

trick\_units(–)

Definition at line 775 of file dyn\_body.hh.

Referenced by `jeod::StructureIntegratedDynBody::collect_forces_and_torques()`, `collect_forces_and_torques()`, `jeod::StructureIntegratedDynBody::complete_translational_acceleration()`, `jeod::StructureIntegratedDynBody::compute_rotational_acceleration()`, `jeod::StructureIntegratedDynBody::compute_translational_acceleration()`, `compute_vehicle_point_derivatives()`, `jeod::StructureIntegratedDynBody::rot_integ()`, `rot_integ()`, `jeod::StructureIntegratedDynBody::solve_constraints()`, and `trans_integ()`.

#### 8.8.5.9 `std::list<DynBody*> jeod::DynBody::dyn_children` [protected]

The subset of the dynamic bodies attached to this dynamic body.

Definition at line 1175 of file dyn\_body.hh.

Referenced by `attach_establish_links()`, `jeod::StructureIntegratedDynBody::collect_forces_and_torques()`, `collect_forces_and_torques()`, `detach()`, `propagate_state_from_composite()`, `propagate_state_from_structure()`, `set_integ_frame()`, and `~DynBody()`.

#### 8.8.5.10 `BaseDynManager*& jeod::DynBody::dyn_manager` [protected]

The dynamics manager for the simulation.

`trick_units(-)`

Definition at line 1149 of file dyn\_body.hh.

Referenced by `add_mass_body_frames()`, `add_mass_body_validate()`, `add_mass_point()`, `attach_to_frame()`, `attach_validate_parent()`, `detach_mass_body_frames()`, `find_body_frame()`, `initialize_controls()`, `initialize_model()`, `reset_controls()`, `set_integ_frame()`, `switch_integration_frames()`, and `~DynBody()`.

#### 8.8.5.11 `DynBody* jeod::DynBody::dyn_parent` [protected]

The [DynBody](#) to which this body is attached.

This points to exactly the same object as does the `links.parent` member. While a mass body can be attached to any kind of mass body, a dynamic body can only be attached to another dynamic body.  
`trick_units(-)`

Definition at line 1162 of file dyn\_body.hh.

Referenced by `attach_establish_links()`, `jeod::StructureIntegratedDynBody::collect_forces_and_torques()`, `collect_forces_and_torques()`, `detach()`, `get_parent_body()`, `get_parent_body_internal()`, `get_root_body_internal()`, `is_root_body()`, `propagate_state()`, `jeod::StructureIntegratedDynBody::PropagateForcesAndTorques()`, `set_state_source()`, `jeod::StructureIntegratedDynBody::solve_constraints()`, `switch_integration_frames()`, `update_integrated_state()`, and `~DynBody()`.

#### 8.8.5.12 `DynBodyGenericFrameAttachment jeod::DynBody::frame_attach` [protected]

The `RefFrame` this body is attached to.

Once attached, the [DynBody](#) will no longer numerically integrate rotational or dynamic states and is considered fixed wrt the `RefFrame`. The [DynBody](#)'s integration frame will continue to be used to populate the `composite_body`, `structure`, `core_body` and `mass point` dynamic states.

Definition at line 1170 of file dyn\_body.hh.

Referenced by `attach_to_frame()`, `detach()`, and `integrate()`.

#### 8.8.5.13 `GravityInteraction jeod::DynBody::grav_interaction`

Gravitational interactions.

This data member specifies how the vehicle interacts gravitationally with various planetary bodies in the simulation and contains the computed acceleration toward those planetary bodies. `trick_units(-)`

Definition at line 770 of file `dyn_body.hh`.

Referenced by `add_control()`, `collect_forces_and_torques()`, `jeod::StructureIntegratedDynBody::complete_translational_acceleration()`, `jeod::StructureIntegratedDynBody::compute_vehicle_point_derivatives()`, `compute_vehicle_point_derivatives()`, `initialize_controls()`, `reset_controls()`, `set_integ_frame()`, and `sort_controls()`.

#### 8.8.5.14 `RefFrameItems jeod::DynBody::initialized_states` `[protected]`

Enum value indicating which of position, velocity, attitude, and rate have been initialized.

`trick_units(-)`

Definition at line 1192 of file `dyn_body.hh`.

Referenced by `attach_update_properties()`, `attach_validate_child()`, `DynBody()`, `get_initialized_states()`, `initialize_model()`, `initialized_states_contains()`, `integrate()`, `propagate_state()`, `propagate_state_from_composite()`, `propagate_state_from_structure()`, `set_state_source_internal()`, and `update_integrated_state()`.

#### 8.8.5.15 `EphemerisRefFrame* jeod::DynBody::integ_frame`

The current integration frame.

`trick_units(-)`

Definition at line 681 of file `dyn_body.hh`.

Referenced by `add_mass_body_frames()`, `add_mass_point()`, `attach_establish_links()`, `initialize_model()`, `integrate()`, and `set_integ_frame()`.

#### 8.8.5.16 `char* jeod::DynBody::integ_frame_name`

The name of the reference frame with respect to which the body's reference frames (core, composite, structure, plus vehicle point frames) are to be represented and propagated.

The value must identify a valid integration frame, i.e., a non-rotating, ephemeris based reference frame.

This member is used at initialization time only. To change the integration frame post-initialization use the function [DynBody::switch\\_integration\\_frames](#). This can be invoked directly, or indirectly via a `FrameSwitch` body action. `trick_units(-)`

Definition at line 676 of file `dyn_body.hh`.

Referenced by `initialize_model()`.

#### 8.8.5.17 `er7_utils::IntegratorResultMergerContainer jeod::DynBody::integ_results_merger` `[protected]`

The object that merges integration results.

`trick_units(-)`

Definition at line 1231 of file `dyn_body.hh`.

Referenced by `create_body_integrators()`, and `integrate()`.

#### 8.8.5.18 `BodyRefFrame* jeod::DynBody::integrated_frame` `[protected]`

The reference frame whose state is updated via the state integrator.

All other reference frames are calculated from this frame. `trick_units(-)`

Definition at line 1218 of file dyn\_body.hh.

Referenced by compute\_ref\_point\_transform(), DynBody(), propagate\_state(), jeod::StructureIntegratedDynBody::StructureIntegratedDynBody(), switch\_integration\_frames(), and update\_integrated\_state().

#### 8.8.5.19 MassBody jeod::DynBody::mass

Mass properties of the vehicle, defined about the structure reference frame.

Definition at line 658 of file dyn\_body.hh.

Referenced by add\_mass\_body(), add\_mass\_point(), attach\_child(), attach\_establish\_links(), attach\_to\_frame(), attach\_update\_properties(), jeod::StructureIntegratedDynBody::collect\_forces\_and\_torques(), collect\_forces\_and\_torques(), jeod::StructureIntegratedDynBody::complete\_translational\_acceleration(), jeod::StructureIntegratedDynBody::compute\_inertial\_torque(), compute\_ref\_point\_transform(), jeod::StructureIntegratedDynBody::compute\_rotational\_acceleration(), jeod::StructureIntegratedDynBody::compute\_translational\_acceleration(), jeod::StructureIntegratedDynBody::compute\_vehicle\_point\_derivatives(), compute\_vehicle\_point\_derivatives(), detach(), detach\_mass\_internal(), DynBody(), initialize\_model(), process\_dynamic\_attachment(), propagate\_state\_from\_composite(), propagate\_state\_from\_structure(), jeod::StructureIntegratedDynBody::PropagateForcesAndTorques(), remove\_mass\_body(), set\_name(), and jeod::StructureIntegratedDynBody::solve\_constraints().

#### 8.8.5.20 std::list<MassBody\*> jeod::DynBody::mass\_children [protected]

The subset of the mass bodies attached to this dynamic body that are themselves not dynamic bodies.

Definition at line 1181 of file dyn\_body.hh.

Referenced by add\_mass\_body(), remove\_mass\_body(), and ~DynBody().

#### 8.8.5.21 NamedItem& jeod::DynBody::name

Body name, reference linked to mass.name.

trick\_units(-)

Definition at line 663 of file dyn\_body.hh.

Referenced by add\_mass\_body(), add\_mass\_body\_frames(), add\_mass\_body\_validate(), add\_mass\_point(), attach\_child(), jeod::StructureIntegratedDynBody::attach\_update\_properties(), attach\_validate\_child(), attach\_validate\_parent(), jeod::check\_frame\_ownership(), compute\_ref\_point\_transform(), jeod::StructureIntegratedDynBody::compute\_vehicle\_point\_derivatives(), compute\_vehicle\_point\_derivatives(), create\_body\_integrators(), jeod::StructureIntegratedDynBody::detach(), detach(), find\_body\_frame(), find\_vehicle\_point(), initialize\_model(), migrate\_integrable\_objects(), propagate\_state(), remove\_mass\_body(), set\_integ\_frame(), jeod::StructureIntegratedDynBody::set\_solver(), set\_state\_source(), and switch\_integration\_frames().

#### 8.8.5.22 BodyRefFrame\* jeod::DynBody::position\_source [protected]

The reference frame that contains the user-set position.

trick\_units(-)

Definition at line 1197 of file dyn\_body.hh.

Referenced by set\_state\_source\_internal(), and update\_integrated\_state().

#### 8.8.5.23 BodyRefFrame\* jeod::DynBody::rate\_source [protected]

The reference frame that contains the user-set attitude rate.

trick\_units(-)



Definition at line 1212 of file dyn\_body.hh.

Referenced by set\_state\_source\_internal(), and update\_integrated\_state().

#### 8.8.5.24 RestartableSO3SecondOrderODEIntegrator jeod::DynBody::rot\_integrator [protected]

Rotational state checkpointable/restartable integrator generator.

Rotational state is much harder to integrate. The canonical position is the attitude quaternion, canonical velocity is angular velocity, and the time derivative of the attitude quaternion is a function of the orientation and the angular velocity.trick\_units(-)

Definition at line 1248 of file dyn\_body.hh.

Referenced by create\_body\_integrators(), DynBody(), reset\_integrators(), jeod::StructureIntegratedDynBody::rot\_integ(), rot\_integ(), and ~DynBody().

#### 8.8.5.25 GeneralizedSecondOrderODETechnique::TechniqueType jeod::DynBody::rotation\_integration

Specifies the preferred mechanism for integrating rotational state.

This data member has effect only when set prior to the creation of the body's integrators. The body's rotational integrator will be created based on the value of this data member.trick\_units(-)

Definition at line 752 of file dyn\_body.hh.

Referenced by create\_body\_integrators().

#### 8.8.5.26 bool jeod::DynBody::rotational\_dynamics

Is rotational dynamics enabled? The body's rotational state is integrated only if this member is true.

Setting this member to false indicates the responsibility for updating the rotational state is performed elsewhere, such as by a user-defined forced rotation model.trick\_units(-)

Definition at line 723 of file dyn\_body.hh.

Referenced by jeod::StructureIntegratedDynBody::collect\_forces\_and\_torques(), collect\_forces\_and\_torques(), jeod::StructureIntegratedDynBody::collect\_local\_forces\_and\_torques(), integrate(), jeod::StructureIntegratedDynBody::PropagateForcesAndTorques(), reset\_integrators(), and jeod::StructureIntegratedDynBody::solve\_constraints().

#### 8.8.5.27 BodyRefFrame jeod::DynBody::structure

Vehicle structural reference frame.

The reference frame origin is at the structural origin, and the reference frame axes are the structure frame axes as defined in the composite mass properties.trick\_units(-)

Definition at line 705 of file dyn\_body.hh.

Referenced by attach\_to\_frame(), attach\_update\_properties(), collect\_forces\_and\_torques(), jeod::StructureIntegratedDynBody::complete\_translational\_acceleration(), jeod::StructureIntegratedDynBody::compute\_inertial\_torque(), compute\_ref\_point\_transform(), jeod::StructureIntegratedDynBody::compute\_translational\_acceleration(), jeod::StructureIntegratedDynBody::compute\_vehicle\_point\_derivatives(), compute\_vehicle\_point\_states(), DynBody(), initialize\_model(), integrate(), process\_dynamic\_attachment(), propagate\_state(), propagate\_state\_from\_composite(), propagate\_state\_from\_structure(), jeod::StructureIntegratedDynBody::PropagateForcesAndTorques(), jeod::StructureIntegratedDynBody::rot\_integ(), set\_integ\_frame(), jeod::StructureIntegratedDynBody::solve\_constraints(), jeod::StructureIntegratedDynBody::StructureIntegratedDynBody(), jeod::StructureIntegratedDynBody::trans\_integ(), and ~DynBody().

**8.8.5.28** `bool jeod::DynBody::three_dof`

Is this a three degrees of freedom (translation only) body? This data member has effect only when set prior to the creation of the body's integrators.

The body's rotational integrator is not created and `rotational_dynamics` is set to false if this member's value is true.

Note that very bad mojo (a core dump) will result if this member is set to true at initialization time and `rotational_dynamics` is later enabled during run time. `trick_units(-)`

Definition at line 743 of file `dyn_body.hh`.

Referenced by `create_body_integrators()`.

**8.8.5.29** `const JeodIntegrationTime* jeod::DynBody::time_manager` `[protected]`

The time manager to be used to obtain timestamp information.

`trick_units(-)`

Definition at line 1154 of file `dyn_body.hh`.

Referenced by `create_body_integrators()`, and `update_integrated_state()`.

**8.8.5.30** `RestartableT3SecondOrderODEIntegrator jeod::DynBody::trans_integrator` `[protected]`

Translational state checkpointable/restartable integrator generator.

Translational state is comparatively easy to integrate. The canonical position is just position, canonical velocity is just velocity, and the time derivative of position is velocity. `trick_units(-)`

Definition at line 1239 of file `dyn_body.hh`.

Referenced by `create_body_integrators()`, `DynBody()`, `reset_integrators()`, `jeod::StructureIntegratedDynBody::trans_integ()`, `trans_integ()`, and `~DynBody()`.

**8.8.5.31** `bool jeod::DynBody::translational_dynamics`

Is translational dynamics enabled? The body's translational state is integrated only if this member is true.

Setting this member to false indicates the responsibility for updating the translational state is performed elsewhere, such as by a user-defined forced translation model. `trick_units(-)`

Definition at line 714 of file `dyn_body.hh`.

Referenced by `jeod::StructureIntegratedDynBody::collect_forces_and_torques()`, `collect_forces_and_torques()`, `jeod::StructureIntegratedDynBody::collect_local_forces_and_torques()`, `integrate()`, `jeod::StructureIntegratedDynBody::PropagateForcesAndTorques()`, `reset_integrators()`, and `jeod::StructureIntegratedDynBody::solve_constraints()`.

**8.8.5.32** `std::list<BodyRefFrame*> jeod::DynBody::vehicle_points` `[protected]`

An array of vehicle points associated with this dynamic body.

Definition at line 1186 of file `dyn_body.hh`.

Referenced by `add_mass_body_frames()`, `add_mass_point()`, `compute_vehicle_point_states()`, `detach_mass_body_frames()`, `find_vehicle_point()`, `set_integ_frame()`, and `~DynBody()`.

**8.8.5.33** `BodyRefFrame* jeod::DynBody::velocity_source` `[protected]`

The reference frame that contains the user-set velocity.

trick\_units(-)

Definition at line 1202 of file dyn\_body.hh.

Referenced by set\_state\_source\_internal(), and update\_integrated\_state().

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

- [dyn\\_body.hh](#)
- [dyn\\_body.cc](#)
- [dyn\\_body\\_attach.cc](#)
- [dyn\\_body\\_collect.cc](#)
- [dyn\\_body\\_detach.cc](#)
- [dyn\\_body\\_find\\_body\\_frame.cc](#)
- [dyn\\_body\\_initialize\\_model.cc](#)
- [dyn\\_body\\_integration.cc](#)
- [dyn\\_body\\_propagate\\_state.cc](#)
- [dyn\\_body\\_set\\_state.cc](#)
- [dyn\\_body\\_vehicle\\_point.cc](#)

## 8.9 jeod::DynBodyGenericFrameAttachment Class Reference

A wrench comprises a torque and a force applied at a point on a [DynBody](#).

```
#include <dyn_body_generic_rigid_attach.hh>
```

### Public Member Functions

- [DynBodyGenericFrameAttachment](#) ()  
*Default constructor.*
- void [initialize\\_attachment](#) (RefFrame &parent\_frame, const RefFrameState &attach\_state)
- void [clear\\_attachment](#) ()
- bool [isAttached](#) () const
- RefFrame \* [get\\_parent\\_frame](#) () const
- const RefFrameState & [get\\_attach\\_offset](#) () const

### Private Attributes

- bool [active](#)  
*trick\_units(-)*
- RefFrame \* [rigid\\_attach\\_parent](#)  
*trick\_units(-)*
- RefFrameState [rigid\\_attach\\_state](#)  
*trick\_units(-)*

### Friends

- class [InputProcessor](#)
- void [init\\_attrjeod\\_\\_DynBodyGenericFrameAttachment](#) ()

### 8.9.1 Detailed Description

A wrench comprises a torque and a force applied at a point on a [DynBody](#).

The torque should not include the torque due to the application of the force.

A Trick simulation issues vcollect statements such as

```
vcollect vehicle.dyn_body.collect_wrench.collection
{
    wrench_model1.wrench,
    wrench_model2.wrench
};
```

Definition at line 78 of file dyn\_body\_generic\_rigid\_attach.hh.

### 8.9.2 Constructor & Destructor Documentation

#### 8.9.2.1 jeod::DynBodyGenericFrameAttachment::DynBodyGenericFrameAttachment ( ) [inline]

Default constructor.

Definition at line 88 of file dyn\_body\_generic\_rigid\_attach.hh.

### 8.9.3 Member Function Documentation

#### 8.9.3.1 void jeod::DynBodyGenericFrameAttachment::clear\_attachment ( ) [inline]

Definition at line 107 of file dyn\_body\_generic\_rigid\_attach.hh.

References active.

Referenced by jeod::DynBody::detach().

#### 8.9.3.2 const RefFrameState& jeod::DynBodyGenericFrameAttachment::get\_attach\_offset ( ) const [inline]

Definition at line 122 of file dyn\_body\_generic\_rigid\_attach.hh.

References rigid\_attach\_state.

Referenced by jeod::DynBody::integrate().

#### 8.9.3.3 RefFrame\* jeod::DynBodyGenericFrameAttachment::get\_parent\_frame ( ) const [inline]

Definition at line 117 of file dyn\_body\_generic\_rigid\_attach.hh.

References rigid\_attach\_parent.

Referenced by jeod::DynBody::integrate().

#### 8.9.3.4 void jeod::DynBodyGenericFrameAttachment::initialize\_attachment ( RefFrame & parent\_frame, const RefFrameState & attach\_state ) [inline]

Definition at line 98 of file dyn\_body\_generic\_rigid\_attach.hh.

References active, rigid\_attach\_parent, and rigid\_attach\_state.

Referenced by jeod::DynBody::attach\_to\_frame().

8.9.3.5 `bool jeod::DynBodyGenericFrameAttachment::isAttached ( ) const [inline]`

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

References active.

Referenced by `jeod::DynBody::detach()`, and `jeod::DynBody::integrate()`.

## 8.9.4 Friends And Related Function Documentation

8.9.4.1 `void init_attrjeod__DynBodyGenericFrameAttachment ( ) [friend]`

8.9.4.2 `friend class InputProcessor [friend]`

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

## 8.9.5 Field Documentation

8.9.5.1 `bool jeod::DynBodyGenericFrameAttachment::active [private]`

`trick_units(-)`

Definition at line 132 of file `dyn_body_generic_rigid_attach.hh`.

Referenced by `clear_attachment()`, `initialize_attachment()`, and `isAttached()`.

8.9.5.2 `RefFrame* jeod::DynBodyGenericFrameAttachment::rigid_attach_parent [private]`

`trick_units(-)`

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

Referenced by `get_parent_frame()`, and `initialize_attachment()`.

8.9.5.3 `RefFrameState jeod::DynBodyGenericFrameAttachment::rigid_attach_state [private]`

`trick_units(-)`

Definition at line 136 of file `dyn_body_generic_rigid_attach.hh`.

Referenced by `get_attach_offset()`, and `initialize_attachment()`.

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

- [dyn\\_body\\_generic\\_rigid\\_attach.hh](#)

## 8.10 jeod::DynBodyMessages Class Reference

Specify the message IDs used in the [DynBody](#) model.

```
#include <dyn_body_messages.hh>
```

### Static Public Attributes

- static char const \* [invalid\\_body](#)  
*Issued when a body is invalid such as not being initialized.*
- static char const \* [invalid\\_group](#)

- Issued when a group is invalid such as not initialized or NULL.*

  - static char const \* [invalid\\_name](#)

*Issued when a name is invalid – NULL, empty, a duplicate, ...*
- static char const \* [invalid\\_frame](#)

*Issued when a frame is invalid – not an integ frame, ...*
- static char const \* [invalid\\_attachment](#)

*Issued when a attachment is invalid from a state point of view.*
- static char const \* [invalid\\_technique](#)

*Issued when an integration technique is invalid.*
- static char const \* [not\\_dyn\\_body](#)

*Issued when a MassBody is expected to be a [DynBody](#) but that is not the case.*
- static char const \* [internal\\_error](#)

*Error issued when some internal error occurred.*

## Private Member Functions

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

## Friends

- class [InputProcessor](#)
- void [init\\_attrjeod\\_\\_DynBodyMessages](#) ()

### 8.10.1 Detailed Description

Specify the message IDs used in the [DynBody](#) model.

#### Assumptions and Limitations

- This is a complete catalog of all the messages sent by the [DynBody](#) model.
- This is not an exhaustive list of all the things that can go awry.

Definition at line 81 of file [dyn\\_body\\_messages.hh](#).

### 8.10.2 Constructor & Destructor Documentation

8.10.2.1 [jeod::DynBodyMessages::DynBodyMessages](#) ( void ) [private]

8.10.2.2 [jeod::DynBodyMessages::DynBodyMessages](#) ( const [DynBodyMessages](#) & ) [private]

### 8.10.3 Member Function Documentation

8.10.3.1 [DynBodyMessages& jeod::DynBodyMessages::operator=](#) ( const [DynBodyMessages](#) & ) [private]

### 8.10.4 Friends And Related Function Documentation

8.10.4.1 void [init\\_attrjeod\\_\\_DynBodyMessages](#) ( ) [friend]

8.10.4.2 friend class [InputProcessor](#) [friend]

Definition at line 83 of file [dyn\\_body\\_messages.hh](#).

### 8.10.5 Field Documentation

#### 8.10.5.1 `char const * jeod::DynBodyMessages::internal_error` `[static]`

##### Initial value:

```
=
    "dynamics/dyn_body/" "internal_error"
```

Error issued when some internal error occurred.

These errors should never happen. `trick_units(-)`

Definition at line 130 of file `dyn_body_messages.hh`.

Referenced by `jeod::DynBody::create_integrators()`, and `jeod::DynBody::get_dynamics_integration_group()`.

#### 8.10.5.2 `char const * jeod::DynBodyMessages::invalid_attachment` `[static]`

##### Initial value:

```
=
    "dynamics/dyn_body/" "invalid_attachment"
```

Issued when a attachment is invalid from a state point of view.

`trick_units(-)`

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

Referenced by `jeod::DynBody::add_mass_body()`, `jeod::DynBody::attach_child()`, `jeod::DynBody::attach_to_-frame()`, `jeod::StructureIntegratedDynBody::attach_update_properties()`, `jeod::DynBody::attach_validate_child()`, `jeod::DynBody::attach_validate_parent()`, `jeod::StructureIntegratedDynBody::detach()`, and `jeod::DynBody::detach()`.

#### 8.10.5.3 `char const * jeod::DynBodyMessages::invalid_body` `[static]`

##### Initial value:

```
=
    "dynamics/dyn_body/" "invalid_body"
```

Issued when a body is invalid such as not being initialized.

`trick_units(-)`

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

Referenced by `jeod::StructureIntegratedDynBody::add_constraint()`, `jeod::DynBody::add_mass_point()`, `jeod::DynBody::attach_validate_parent()`, `jeod::StructureIntegratedDynBody::set_solver()`, and `jeod::StructureIntegratedDynBody::solve_constraints()`.

#### 8.10.5.4 `char const * jeod::DynBodyMessages::invalid_frame` `[static]`

##### Initial value:

```
=
    "dynamics/dyn_body/" "invalid_frame"
```

Issued when a frame is invalid – not an integ frame, ...

`trick_units(-)`

Definition at line 108 of file dyn\_body\_messages.hh.

Referenced by jeod::check\_frame\_ownership(), jeod::DynBody::compute\_ref\_point\_transform(), jeod::StructureIntegratedDynBody::compute\_vehicle\_point\_derivatives(), jeod::DynBody::compute\_vehicle\_point\_derivatives(), jeod::DynBody::initialize\_model(), jeod::DynBody::propagate\_state(), jeod::DynBody::set\_state\_source(), and jeod::DynBody::switch\_integration\_frames().

#### 8.10.5.5 char const \* jeod::DynBodyMessages::invalid\_group [static]

**Initial value:**

```
=
    "dynamics/dyn_body/" "invalid_group"
```

Issued when a group is invalid such as not initialized or NULL.

trick\_units(–)

Definition at line 98 of file dyn\_body\_messages.hh.

Referenced by jeod::DynBody::migrate\_integrable\_objects().

#### 8.10.5.6 char const \* jeod::DynBodyMessages::invalid\_name [static]

**Initial value:**

```
=
    "dynamics/dyn_body/" "invalid_name"
```

Issued when a name is invalid – NULL, empty, a duplicate, ...

trick\_units(–)

Definition at line 103 of file dyn\_body\_messages.hh.

Referenced by jeod::DynBody::find\_body\_frame(), jeod::DynBody::initialize\_model(), jeod::DynBody::set\_integ\_frame(), and jeod::DynBody::switch\_integration\_frames().

#### 8.10.5.7 char const \* jeod::DynBodyMessages::invalid\_technique [static]

**Initial value:**

```
=
    "dynamics/dyn_body/" "invalid_technique"
```

Issued when an integration technique is invalid.

trick\_units(–)

Definition at line 118 of file dyn\_body\_messages.hh.

Referenced by jeod::DynBody::detach(), and jeod::DynBody::remove\_mass\_body().

#### 8.10.5.8 char const \* jeod::DynBodyMessages::not\_dyn\_body [static]

**Initial value:**

```
=
    "dynamics/dyn_body/" "not_dyn_body"
```



Issued when a MassBody is expected to be a [DynBody](#) but that is not the case.

trick\_units(-)

Definition at line 124 of file dyn\_body\_messages.hh.

Referenced by jeod::DynBody::attach\_validate\_parent().

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

- [dyn\\_body\\_messages.hh](#)
- [dyn\\_body\\_messages.cc](#)

## 8.11 jeod::Force Class Reference

A [Force](#) represents a Newtonian force that acts on a [DynBody](#).

```
#include <force.hh>
```

### Public Member Functions

- [Force](#) ()  
*Force default constructor.*
- virtual [~Force](#) ()  
*Force destructor.*
- double & [operator\[\]](#) (const unsigned int index)  
*Access a force element, non-const version.*
- double [operator\[\]](#) (const unsigned int index) const  
*Access a force element, const version.*

### Data Fields

- bool [active](#)  
*Is this force active?*
- double [force](#) [3]  
*Force vector.*

### Private Member Functions

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

#### 8.11.1 Detailed Description

A [Force](#) represents a Newtonian force that acts on a [DynBody](#).

The class encapsulates an active flag and a 3-vector that contains the force components. Forces are collected in one of a [DynBody](#) object's force collection STL vectors. The force vector is expressed in the structural frame of that [DynBody](#) object.

The [Force](#) class is the recommended mechanism for representing forces in JEOD. While 3-vectors can also be collected into a collect STL vector, there is no way to turn off these collected 3-vectors. Even worse, there is no

way to tell whether a collected 3-vector does indeed represent a force – or even if it is a 3-vector. In comparison, [Force](#) objects can be turned on and off, and more importantly, they are type-safe.

Definition at line 82 of file force.hh.

## 8.11.2 Constructor & Destructor Documentation

### 8.11.2.1 `jeod::Force::Force ( void )`

[Force](#) default constructor.

Definition at line 44 of file force.cc.

References [force](#).

### 8.11.2.2 `jeod::Force::~~Force ( void ) [virtual]`

[Force](#) destructor.

Definition at line 56 of file force.cc.

### 8.11.2.3 `jeod::Force::Force ( const Force & ) [private]`

Not implemented.

## 8.11.3 Member Function Documentation

### 8.11.3.1 `Force& jeod::Force::operator= ( const Force & ) [private]`

Not implemented.

### 8.11.3.2 `double &jeod::Force::operator[] ( const unsigned int index ) [inline]`

Access a force element, non-const version.

#### Returns

[Force](#) component at specified index  
Units: N

#### Parameters

<code>in</code>	<code><i>index</i></code>	Index number
-----------------	---------------------------	--------------

Definition at line 76 of file force\_inline.hh.

References [force](#).

### 8.11.3.3 `double jeod::Force::operator[] ( const unsigned int index ) const [inline]`

Access a force element, const version.

#### Returns

[Force](#) component at specified index  
Units: N

## Parameters

<code>in</code>	<code>index</code>	Index number
-----------------	--------------------	--------------

Definition at line 89 of file `force_inline.hh`.

References `force`.

### 8.11.4 Field Documentation

#### 8.11.4.1 `bool jeod::Force::active`

Is this force active?

`trick_units(-)`

Definition at line 98 of file `force.hh`.

#### 8.11.4.2 `double jeod::Force::force[3]`

[Force](#) vector.

`trick_units(N)`

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

Referenced by `Force()`, and `operator[]()`.

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

- [force.hh](#)
- [force\\_inline.hh](#)
- [force.cc](#)

## 8.12 jeod::FrameDerivs Class Reference

Contains translational and rotational second derivatives.

```
#include <frame_derivs.hh>
```

### Public Member Functions

- [FrameDerivs](#) ()  
*Default constructor.*

### Data Fields

- double [non\\_grav\\_accel](#) [3]  
*Non-gravitational acceleration.*
- double [trans\\_accel](#) [3]  
*Total acceleration.*
- Quaternion [Qdot\\_parent\\_this](#)  
*Time derivative of  $Q_{parent\_this}$ .*
- double [rot\\_accel](#) [3]  
*Total rotational acceleration (expressed in body frame)*

### 8.12.1 Detailed Description

Contains translational and rotational second derivatives.

Definition at line 73 of file `frame_derivs.hh`.

### 8.12.2 Constructor & Destructor Documentation

#### 8.12.2.1 `jeod::FrameDerivs::FrameDerivs ( void )`

Default constructor.

Definition at line 107 of file `aux_classes.cc`.

References `non_grav_accel`, `rot_accel`, and `trans_accel`.

### 8.12.3 Field Documentation

#### 8.12.3.1 `double jeod::FrameDerivs::non_grav_accel[3]`

Non-gravitational acceleration.

trick\_units(m/s<sup>2</sup>)

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

Referenced by `jeod::StructureIntegratedDynBody::collect_forces_and_torques()`, `jeod::DynBody::collect_forces_and_torques()`, `jeod::StructureIntegratedDynBody::complete_translational_acceleration()`, `jeod::StructureIntegratedDynBody::compute_translational_acceleration()`, `jeod::StructureIntegratedDynBody::compute_vehicle_point_derivatives()`, `jeod::DynBody::compute_vehicle_point_derivatives()`, `FrameDerivs()`, and `jeod::StructureIntegratedDynBody::solve_constraints()`.

#### 8.12.3.2 Quaternion `jeod::FrameDerivs::Qdot_parent_this`

Time derivative of `Q_parent_this`.

trick\_units(1/s)

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

Referenced by `jeod::StructureIntegratedDynBody::compute_vehicle_point_derivatives()`, `jeod::DynBody::compute_vehicle_point_derivatives()`, `jeod::StructureIntegratedDynBody::rot_integ()`, and `jeod::DynBody::rot_integ()`.

#### 8.12.3.3 `double jeod::FrameDerivs::rot_accel[3]`

Total rotational acceleration (expressed in body frame)

trick\_units(rad/s<sup>2</sup>)

Definition at line 98 of file `frame_derivs.hh`.

Referenced by `jeod::StructureIntegratedDynBody::collect_forces_and_torques()`, `jeod::DynBody::collect_forces_and_torques()`, `jeod::StructureIntegratedDynBody::complete_translational_acceleration()`, `jeod::StructureIntegratedDynBody::compute_rotational_acceleration()`, `jeod::StructureIntegratedDynBody::compute_vehicle_point_derivatives()`, `jeod::DynBody::compute_vehicle_point_derivatives()`, `FrameDerivs()`, `jeod::StructureIntegratedDynBody::rot_integ()`, `jeod::DynBody::rot_integ()`, and `jeod::StructureIntegratedDynBody::solve_constraints()`.

#### 8.12.3.4 `double jeod::FrameDerivs::trans_accel[3]`

Total acceleration.

trick\_units(m/s2)

Definition at line 88 of file frame\_derivs.hh.

Referenced by jeod::StructureIntegratedDynBody::collect\_forces\_and\_torques(), jeod::DynBody::collect\_forces\_and\_torques(), jeod::StructureIntegratedDynBody::complete\_translational\_acceleration(), jeod::StructureIntegratedDynBody::compute\_vehicle\_point\_derivatives(), jeod::DynBody::compute\_vehicle\_point\_derivatives(), FrameDerivs(), jeod::StructureIntegratedDynBody::trans\_integ(), and jeod::DynBody::trans\_integ().

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

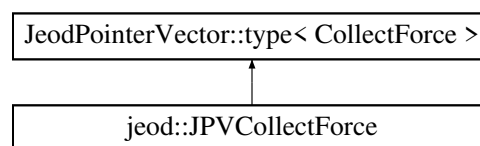
- [frame\\_derivs.hh](#)
- [aux\\_classes.cc](#)

## 8.13 jeod::JPVCollectForce Class Reference

This is a derived version of the template class JeodPointerVector<CollectForce>::type with an implementation of the method perform\_cleanup\_action which frees and clears stale data following a restore.

```
#include <body_force_collect.hh>
```

Inheritance diagram for jeod::JPVCollectForce:



### Public Member Functions

- void [perform\\_insert\\_action](#) (const std::string &value) override  
*Interpret the provided value and add it to the list.*
- void [push\\_back](#) (CollectForce \*const &elem)  
*Add an element to the end of the contents.*

### Friends

- template<typename CollectType , typename value\_type >  
void [collect\\_insert](#) (CollectType &collect\_in, value\_type &elem)
- template<typename CollectType , typename value\_type >  
void [collect\\_push\\_back](#) (CollectType &collect\_in, value\_type &elem)

### 8.13.1 Detailed Description

This is a derived version of the template class JeodPointerVector<CollectForce>::type with an implementation of the method perform\_cleanup\_action which frees and clears stale data following a restore.

Definition at line 168 of file body\_force\_collect.hh.

### 8.13.2 Member Function Documentation

8.13.2.1 void jeod::JPVCollectForce::perform\_insert\_action ( const std::string & value ) [inline],[override]

Interpret the provided value and add it to the list.

For a [JPVCollectForce](#), the value should specify (in string form) the address of a unique force vector pointer in active memory. If the entry already exists, check and delete the "restored" [CollectTorque](#)

Definition at line 185 of file `body_force_collect.hh`.

References `collect_insert`.

**8.13.2.2** `void jeod::JPVCollectForce::push_back ( CollectForce *const & elem ) [inline]`

Add an element to the end of the contents.

Parameters

<i>elem</i>	Element to be added.
-------------	----------------------

Definition at line 198 of file `body_force_collect.hh`.

References `collect_push_back`.

### 8.13.3 Friends And Related Function Documentation

**8.13.3.1** `template<typename CollectType , typename value_type > void collect_insert ( CollectType & collect_in, value_type & elem ) [friend]`

Definition at line 99 of file `body_force_collect.hh`.

Referenced by `perform_insert_action()`.

**8.13.3.2** `template<typename CollectType , typename value_type > void collect_push_back ( CollectType & collect_in, value_type & elem ) [friend]`

Definition at line 131 of file `body_force_collect.hh`.

Referenced by `push_back()`.

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

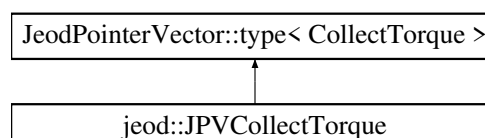
- [body\\_force\\_collect.hh](#)

## 8.14 jeod::JPVCollectTorque Class Reference

This is a derived version of the template class `JeodPointerVector<CollectTorque>::type` with an implementation of the method `perform_cleanup_action` which frees and clears stale data following a restore.

```
#include <body_force_collect.hh>
```

Inheritance diagram for `jeod::JPVCollectTorque`:



### Public Member Functions

- void [perform\\_insert\\_action](#) (const std::string &value) override  
*Interpret the provided value and add it to the list.*

- void [push\\_back](#) ([CollectTorque](#) \*const &elem)  
*Add an element to the end of the contents.*

## Friends

- template<typename CollectType , typename value\_type >  
void [collect\\_insert](#) (CollectType &collect\_in, value\_type &elem)
- template<typename CollectType , typename value\_type >  
void [collect\\_push\\_back](#) (CollectType &collect\_in, value\_type &elem)

### 8.14.1 Detailed Description

This is a derived version of the template class JeodPointerVector<CollectTorque>::type with an implementation of the method perform\_cleanup\_action which frees and clears stale data following a restore.

Definition at line 211 of file body\_force\_collect.hh.

### 8.14.2 Member Function Documentation

8.14.2.1 void jeod::JPVCollectTorque::perform\_insert\_action ( const std::string & value ) [inline], [override]

Interpret the provided value and add it to the list.

For a [JPVCollectTorque](#), the value should specify (in string form) the address of a unique torque vector pointer in active memory. If the entry already exists, check and delete the "restored" [CollectTorque](#)

Definition at line 226 of file body\_force\_collect.hh.

References [collect\\_insert](#).

8.14.2.2 void jeod::JPVCollectTorque::push\_back ( [CollectTorque](#) \*const & elem ) [inline]

Add an element to the end of the contents.

Parameters

<i>elem</i>	Element to be added.
-------------	----------------------

Definition at line 239 of file body\_force\_collect.hh.

References [collect\\_push\\_back](#).

### 8.14.3 Friends And Related Function Documentation

8.14.3.1 template<typename CollectType , typename value\_type > void [collect\\_insert](#) ( [CollectType](#) & *collect\_in*, value\_type & *elem* ) [friend]

Definition at line 99 of file body\_force\_collect.hh.

Referenced by [perform\\_insert\\_action\(\)](#).

8.14.3.2 template<typename CollectType , typename value\_type > void [collect\\_push\\_back](#) ( [CollectType](#) & *collect\_in*, value\_type & *elem* ) [friend]

Definition at line 131 of file body\_force\_collect.hh.

Referenced by [push\\_back\(\)](#).

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

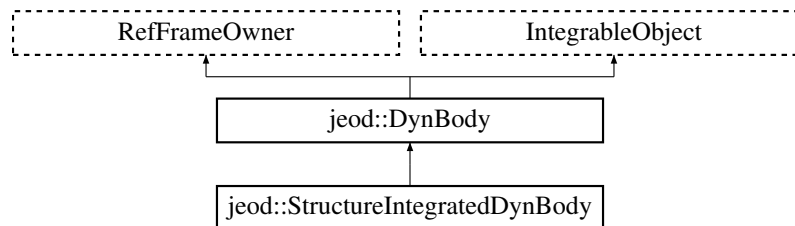
- [body\\_force\\_collect.hh](#)

## 8.15 jeod::StructureIntegratedDynBody Class Reference

Extends [DynBody](#) to integrate an object's structural reference frame as opposed to its center of mass.

```
#include <structure_integrated_dyn_body.hh>
```

Inheritance diagram for jeod::StructureIntegratedDynBody:



### Public Member Functions

- [StructureIntegratedDynBody](#) ()  
*Constructor.*
- [~StructureIntegratedDynBody](#) () override  
*Destructor.*
- void [collect\\_forces\\_and\\_torques](#) () override  
*Compute the rotational and translational accelerations that result from the collected forces and torques acting on the vehicle.*
- void [set\\_solver](#) (DynBodyConstraintsSolver &solver\_in)  
*Set the solver to be used to solve constraints.*
- void [add\\_constraint](#) (DynBodyConstraint \*constraint)  
*Add a constraint to the constraints solver.*
- virtual void [solve\\_constraints](#) ()  
*Solve for constraint forces and torques acting on the vehicle and apply them to the vehicle.*
- void [compute\\_vehicle\\_point\\_derivatives](#) (const BodyRefFrame &frame, FrameDerivs &derivs) override  
*Compute the state derivatives at a vehicle point.*
- bool [detach](#) (DynBody &other\_body) override  
*Break the logical connectivity between parent and child.*

### Data Fields

- [BodyWrenchCollect](#) effector\_wrench\_collection  
*Collection of effector wrenches.*

### Protected Member Functions

- void [attach\\_update\\_properties](#) (const double offset\_pstr\_cstr\_pstr[3], const double T\_pstr\_cstr[3][3], DynBody &child) override  
*Set the relation between parent and child and update the mass properties.*
- const [VehicleProperties](#) & [get\\_vehicle\\_properties](#) () const  
*Get the vehicle properties as a const reference.*
- er7\_utils::IntegratorResult [trans\\_integ](#) (double dyn\_dt, unsigned int target\_stage) override



- *Integrate the translational state of a [StructureIntegratedDynBody](#).*
- `er7_utils::IntegratorResult rot_integ` (double dyn\_dt, unsigned int target\_stage) override
- *Integrate the rotational state of a [StructureIntegratedDynBody](#).*
- `void collect_local_forces_and_torques ()`
- *Collect the local forces and torques that directly act on the vehicle.*
- `void PropagateForcesAndTorques ()`
- *Propagate forces and torques up the kinematic chain.*
- `void compute_inertial_torque ()`
- *Compute the inertial torque.*
- `void compute_rotational_acceleration ()`
- *Compute the body- and structure-referenced rotational acceleration.*
- `void compute_translational_acceleration ()`
- *Compute the inertial-referenced translational acceleration vector.*
- `void complete_translational_acceleration ()`
- *Finalize computation of the inertial-referenced translational acceleration vector.*

## Protected Attributes

- `DynBodyConstraintsSolver * constraints_solver`
- *The solver for constraint forces and torques, if there are any.*
- `Wrench effector_wrench`
- *Wrench into which the effector wrenches are accumulated.*
- `FrameDerivs struct_derivs`
- *Translational/rotational accelerations of the structural frame.*
- `VehicleProperties vehicle_properties`
- *Various properties of the vehicle, for the constraints solver.*
- `VehicleNonGravState non_grav_state`
- *Rotational and translational behaviors, for the constraints solver.*
- `double inertial_accel_struct_omega [3]`
- *Structure-referenced inertial acceleration at the structure frame origin due to vehicle angular velocity.*
- `double inertial_accel_struct_omega_dot [3]`
- *Structure-referenced inertial acceleration at the structure frame origin due to vehicle angular acceleration.*
- `double inertial_accel_struct [3]`
- *Structure-referenced inertial acceleration at the structure frame origin.*
- `double inertial_accel_inrtl [3]`
- *Inertial-referenced inertial acceleration at the structure frame origin.*

## Private Member Functions

- `StructureIntegratedDynBody` (const `StructureIntegratedDynBody` &)
- *Not implemented.*
- `StructureIntegratedDynBody & operator=` (const `StructureIntegratedDynBody` &)
- *Not implemented.*

## Friends

- class `InputProcessor`
- class `DynBodyConstraintsSolver`
- `void init_attrjeod__StructureIntegratedDynBody ()`

### 8.15.1 Detailed Description

Extends [DynBody](#) to integrate an object's structural reference frame as opposed to its center of mass.

In addition to structure integration, this class introduces two new concepts, wrenches and constrained objects. A wrench encapsulates a force applied at a point and a torque, with the torque induced by the force due to an off-centerline force direction automatically calculated by JEOD. A constrained object is an object that lies outside the [DynBody](#) system boundary that exchanges translational and/or rotational momentum with the [DynBody](#) and that is somehow constrained by the translation and/or rotational behavior of the [DynBody](#).

These new concepts might be migrated up the [DynBody](#) inheritance chain in subsequent releases of JEOD.

Definition at line 91 of file `structure_integrated_dyn_body.hh`.

### 8.15.2 Constructor & Destructor Documentation

#### 8.15.2.1 `jeod::StructureIntegratedDynBody::StructureIntegratedDynBody ( )`

Constructor.

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

References `jeod::DynBody::integrated_frame`, and `jeod::DynBody::structure`.

#### 8.15.2.2 `jeod::StructureIntegratedDynBody::~StructureIntegratedDynBody ( ) [override]`

Destructor.

Definition at line 59 of file `structure_integrated_dyn_body.cc`.

#### 8.15.2.3 `jeod::StructureIntegratedDynBody::StructureIntegratedDynBody ( const StructureIntegratedDynBody & ) [private]`

Not implemented.

### 8.15.3 Member Function Documentation

#### 8.15.3.1 `void jeod::StructureIntegratedDynBody::add_constraint ( DynBodyConstraint * constraint )`

Add a constraint to the constraints solver.

Note

Both the constraint and the solver must be non-null.

Parameters

<i>constraint</i>	The constraint to be added to the solver.
-------------------	---

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

References `constraints_solver`, and `jeod::DynBodyMessages::invalid_body`.

#### 8.15.3.2 `void jeod::StructureIntegratedDynBody::attach_update_properties ( const double offset_ptr_cstr_ptr[3], const double T_ptr_cstr[3][3], DynBody & child ) [override], [protected], [virtual]`

Set the relation between parent and child and update the mass properties.

## Parameters

in	<i>offset_pstr_cstr_pstr</i>	Location of the child body's structural origin with respect to the parent body's structural origin, specified in structural coordinates of the parent body.
in	<i>T_pstr_cstr</i>	Transformation matrix from the parent body's structural frame to the child body's structural frame.
in, out	<i>child</i>	The child body being attached to this body.

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

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

References [jeod::DynBody::attach\\_update\\_properties\(\)](#), [constraints\\_solver](#), [jeod::DynBodyMessages::invalid\\_attachment](#), [jeod::DynBody::name](#), and [vehicle\\_properties](#).

### 8.15.3.3 void jeod::StructureIntegratedDynBody::collect\_forces\_and\_torques ( ) [override],[virtual]

Compute the rotational and translational accelerations that result from the collected forces and torques acting on the vehicle.

This function should be called as a derivative class job, with a moderately high phase number. Functions that calculate the gravitational acceleration and the effector, environmental, and non-transmitted forces and torques should be called as scheduled jobs or as lower phase derivative class jobs.

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

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

References [jeod::DynBody::collect](#), [collect\\_local\\_forces\\_and\\_torques\(\)](#), [compute\\_inertial\\_torque\(\)](#), [compute\\_rotational\\_acceleration\(\)](#), [compute\\_translational\\_acceleration\(\)](#), [jeod::DynBody::derivs](#), [jeod::DynBody::dyn\\_children](#), [jeod::DynBody::dyn\\_parent](#), [jeod::BodyForceCollect::effector\\_forc](#), [jeod::BodyForceCollect::effector\\_torq](#), [effector\\_wrench](#), [jeod::BodyForceCollect::environ\\_forc](#), [jeod::BodyForceCollect::environ\\_torq](#), [jeod::BodyForceCollect::extern\\_forc\\_inrtl](#), [jeod::BodyForceCollect::extern\\_forc\\_struct](#), [jeod::BodyForceCollect::extern\\_torq\\_body](#), [jeod::BodyForceCollect::extern\\_torq\\_struct](#), [jeod::Wrench::get\\_force\(\)](#), [jeod::Wrench::get\\_torque\(\)](#), [jeod::BodyForceCollect::inertial\\_torq](#), [jeod::DynBody::mass](#), [jeod::BodyForceCollect::no\\_xmit\\_forc](#), [jeod::BodyForceCollect::no\\_xmit\\_torq](#), [jeod::FrameDerivs::non\\_grav\\_accel](#), [PropagateForcesAndTorques\(\)](#), [jeod::FrameDerivs::rot\\_accel](#), [jeod::DynBody::rotational\\_dynamics](#), [struct\\_derivs](#), [jeod::FrameDerivs::trans\\_accel](#), [jeod::Wrench::transform\\_to\\_point\(\)](#), and [jeod::DynBody::translational\\_dynamics](#).

### 8.15.3.4 void jeod::StructureIntegratedDynBody::collect\_local\_forces\_and\_torques ( ) [protected]

Collect the local forces and torques that directly act on the vehicle.

Definition at line 180 of file `structure_integrated_dyn_body_collect.cc`.

References [jeod::BodyWrenchCollect::accumulate\(\)](#), [jeod::accumulate\\_forces\(\)](#), [jeod::accumulate\\_torques\(\)](#), [jeod::DynBody::collect](#), [jeod::BodyForceCollect::collect\\_effector\\_forc](#), [jeod::BodyForceCollect::collect\\_effector\\_torq](#), [jeod::BodyForceCollect::collect\\_environ\\_forc](#), [jeod::BodyForceCollect::collect\\_environ\\_torq](#), [jeod::BodyForceCollect::collect\\_no\\_xmit\\_forc](#), [jeod::BodyForceCollect::collect\\_no\\_xmit\\_torq](#), [jeod::BodyForceCollect::effector\\_forc](#), [jeod::BodyForceCollect::effector\\_torq](#), [effector\\_wrench](#), [effector\\_wrench\\_collection](#), [jeod::BodyForceCollect::environ\\_forc](#), [jeod::BodyForceCollect::environ\\_torq](#), [jeod::BodyForceCollect::no\\_xmit\\_forc](#), [jeod::BodyForceCollect::no\\_xmit\\_torq](#), [jeod::Wrench::reset\\_force\\_and\\_torque\(\)](#), [jeod::DynBody::rotational\\_dynamics](#), and [jeod::DynBody::translational\\_dynamics](#).

Referenced by [collect\\_forces\\_and\\_torques\(\)](#).

### 8.15.3.5 void jeod::StructureIntegratedDynBody::complete\_translational\_acceleration ( ) [protected]

Finalize computation of the inertial-referenced translational acceleration vector.

Definition at line 417 of file `structure_integrated_dyn_body_collect.cc`.

References `jeod::DynBody::derivs`, `jeod::DynBody::grav_interaction`, `inertial_accel_inrtl`, `inertial_accel_struct`, `inertial_accel_struct_omega`, `inertial_accel_struct_omega_dot`, `jeod::DynBody::mass`, `jeod::FrameDerivs::non_grav_accel`, `jeod::FrameDerivs::rot_accel`, `struct_derivs`, `jeod::DynBody::structure`, and `jeod::FrameDerivs::trans_accel`.

Referenced by `compute_translational_acceleration()`, and `solve_constraints()`.

#### 8.15.3.6 `void jeod::StructureIntegratedDynBody::compute_inertial_torque ( )` `[protected]`

Compute the inertial torque.

Definition at line 330 of file `structure_integrated_dyn_body_collect.cc`.

References `jeod::DynBody::collect`, `jeod::BodyForceCollect::inertial_torq`, `jeod::DynBody::mass`, and `jeod::DynBody::structure`.

Referenced by `collect_forces_and_torques()`.

#### 8.15.3.7 `void jeod::StructureIntegratedDynBody::compute_rotational_acceleration ( )` `[protected]`

Compute the body- and structure-referenced rotational acceleration.

Definition at line 355 of file `structure_integrated_dyn_body_collect.cc`.

References `jeod::DynBody::collect`, `jeod::DynBody::derivs`, `jeod::BodyForceCollect::extern_torq_body`, `jeod::BodyForceCollect::extern_torq_struct`, `jeod::BodyForceCollect::inertial_torq`, `jeod::DynBody::mass`, `jeod::FrameDerivs::rot_accel`, and `struct_derivs`.

Referenced by `collect_forces_and_torques()`.

#### 8.15.3.8 `void jeod::StructureIntegratedDynBody::compute_translational_acceleration ( )` `[protected]`

Compute the inertial-referenced translational acceleration vector.

Definition at line 387 of file `structure_integrated_dyn_body_collect.cc`.

References `jeod::DynBody::collect`, `complete_translational_acceleration()`, `jeod::DynBody::derivs`, `jeod::BodyForceCollect::extern_forc_inrtl`, `jeod::BodyForceCollect::extern_forc_struct`, `inertial_accel_struct_omega`, `jeod::DynBody::mass`, `jeod::FrameDerivs::non_grav_accel`, and `jeod::DynBody::structure`.

Referenced by `collect_forces_and_torques()`.

#### 8.15.3.9 `void jeod::StructureIntegratedDynBody::compute_vehicle_point_derivatives ( const BodyRefFrame & frame, FrameDerivs & derivs )` `[override]`, `[virtual]`

Compute the state derivatives at a vehicle point.

Parameters

<i>frame</i>	The vehicle point, as a <a href="#">BodyRefFrame</a> , at which derivatives are to be calculated.
<i>derivs</i>	The calculated derivatives.

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

Definition at line 33 of file `structure_integrated_dyn_body_pt_accel.cc`.

References `jeod::DynBody::composite_body`, `jeod::DynBody::get_root_body()`, `jeod::DynBody::grav_interaction`, `jeod::DynBodyMessages::invalid_frame`, `jeod::DynBody::mass`, `jeod::BodyRefFrame::mass_point`, `jeod::DynBody::name`, `jeod::FrameDerivs::non_grav_accel`, `jeod::FrameDerivs::Qdot_parent_this`, `jeod::FrameDerivs::rot_accel`, `struct_derivs`, `jeod::DynBody::structure`, and `jeod::FrameDerivs::trans_accel`.

8.15.3.10 `bool jeod::StructureIntegratedDynBody::detach ( DynBody & other_body )` `[override]`, `[virtual]`

Break the logical connectivity between parent and child.

## Parameters

<i>in, out</i>	<i>other_body</i>	The other body to detach from
----------------	-------------------	-------------------------------

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

Definition at line 70 of file `structure_integrated_dyn_body_solve.cc`.

References `constraints_solver`, `detach()`, `jeod::DynBody::detach()`, `jeod::DynBody::get_parent_body()`, `jeod::DynBodyMessages::invalid_attachment`, `jeod::DynBody::name`, and `vehicle_properties`.

Referenced by `detach()`.

**8.15.3.11** `const VehicleProperties& jeod::StructureIntegratedDynBody::get_vehicle_properties ( ) const` `[inline]`, `[protected]`

Get the vehicle properties as a const reference.

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

References `vehicle_properties`.

**8.15.3.12** `StructureIntegratedDynBody& jeod::StructureIntegratedDynBody::operator= ( const StructureIntegratedDynBody & )` `[private]`

Not implemented.

**8.15.3.13** `void jeod::StructureIntegratedDynBody::PropagateForcesAndTorques ( )` `[protected]`

Propagate forces and torques up the kinematic chain.

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

References `jeod::DynBody::collect`, `jeod::DynBody::composite_body`, `jeod::DynBody::dyn_parent`, `jeod::BodyForceCollect::effector_forc`, `jeod::BodyForceCollect::effector_torq`, `effector_wrench`, `jeod::BodyForceCollect::environ_forc`, `jeod::BodyForceCollect::environ_torq`, `jeod::DynBody::mass`, `jeod::DynBody::rotational_dynamics`, `jeod::DynBody::structure`, `jeod::Wrench::transform_to_parent()`, and `jeod::DynBody::translational_dynamics`.

Referenced by `collect_forces_and_torques()`.

**8.15.3.14** `er7_utils::IntegratorResult jeod::StructureIntegratedDynBody::rot_integ ( double dyn_dt, unsigned int target_stage )` `[override]`, `[protected]`, `[virtual]`

Integrate the rotational state of a [StructureIntegratedDynBody](#).

## Parameters

<i>in</i>	<i>dyn_dt</i>	Dynamic time step, in dynamic time seconds.
<i>in</i>	<i>target_stage</i>	The stage of the integration process that the integrator should try to attain.

## Returns

The status (time advance, pass/fail status) of the integration.

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

Definition at line 52 of file `structure_integrated_dyn_body_integration.cc`.

References `jeod::DynBody::derivs`, `jeod::FrameDerivs::Qdot_parent_this`, `jeod::FrameDerivs::rot_accel`, `jeod::DynBody::rot_integrator`, `struct_derivs`, and `jeod::DynBody::structure`.

#### 8.15.3.15 void jeod::StructureIntegratedDynBody::set\_solver ( DynBodyConstraintsSolver & solver\_in )

Set the solver to be used to solve constraints.

Definition at line 108 of file structure\_integrated\_dyn\_body\_solve.cc.

References constraints\_solver, jeod::DynBodyMessages::invalid\_body, and jeod::DynBody::name.

#### 8.15.3.16 void jeod::StructureIntegratedDynBody::solve\_constraints ( ) [virtual]

Solve for constraint forces and torques acting on the vehicle and apply them to the vehicle.

This function should be called as a derivative class job, with a very high phase number. Functions that calculate the constraints should be called as derivative class jobs with a phase intermediate between that of collect\_forces\_and\_torques and of this function.

Definition at line 142 of file structure\_integrated\_dyn\_body\_solve.cc.

References jeod::VehicleNonGravState::accel\_struct, jeod::DynBody::collect, complete\_translational\_acceleration(), jeod::DynBody::composite\_body, constraints\_solver, jeod::DynBody::derivs, jeod::DynBody::dyn\_parent, jeod::BodyForceCollect::extern\_forc\_struct, jeod::BodyForceCollect::inertial\_torq, jeod::VehicleNonGravState::inertial\_torque\_struct, jeod::DynBodyMessages::invalid\_body, jeod::DynBody::mass, jeod::FrameDerivs::non\_grav\_accel, non\_grav\_state, jeod::VehicleNonGravState::omega\_body, jeod::VehicleNonGravState::omega\_dot\_body, jeod::VehicleNonGravState::omega\_dot\_struct, jeod::VehicleNonGravState::omega\_struct, jeod::FrameDerivs::rot\_accel, jeod::DynBody::rotational\_dynamics, struct\_derivs, jeod::DynBody::structure, jeod::DynBody::translational\_dynamics, and vehicle\_properties.

#### 8.15.3.17 er7\_utils::IntegratorResult jeod::StructureIntegratedDynBody::trans\_integ ( double dyn\_dt, unsigned int target\_stage ) [override], [protected], [virtual]

Integrate the translational state of a [StructureIntegratedDynBody](#).

##### Parameters

in	<i>dyn_dt</i>	Dynamic time step, in dynamic time seconds.
in	<i>target_stage</i>	The stage of the integration process that the integrator should try to attain.

##### Returns

The status (time advance, pass/fail status) of the integration.

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

Definition at line 38 of file structure\_integrated\_dyn\_body\_integration.cc.

References struct\_derivs, jeod::DynBody::structure, jeod::FrameDerivs::trans\_accel, and jeod::DynBody::trans\_integrator.

## 8.15.4 Friends And Related Function Documentation

#### 8.15.4.1 friend class DynBodyConstraintsSolver [friend]

Definition at line 95 of file structure\_integrated\_dyn\_body.hh.

#### 8.15.4.2 void init\_attrjeod\_\_StructureIntegratedDynBody ( ) [friend]

#### 8.15.4.3 friend class InputProcessor [friend]

Definition at line 93 of file structure\_integrated\_dyn\_body.hh.

### 8.15.5 Field Documentation

#### 8.15.5.1 **DynBodyConstraintsSolver\*** `jeod::StructureIntegratedDynBody::constraints_solver` [protected]

The solver for constraint forces and torques, if there are any.

This needs to be assigned prior to initialization time in simulations that invoke member function `solve_constraints()` during runtime. This can be left unassigned (null) in simulations that do not have vehicular constraints.`trick_units(-)`

Definition at line 200 of file `structure_integrated_dyn_body.hh`.

Referenced by `add_constraint()`, `attach_update_properties()`, `detach()`, `set_solver()`, and `solve_constraints()`.

#### 8.15.5.2 **Wrench** `jeod::StructureIntegratedDynBody::effector_wrench` [protected]

**Wrench** into which the effector wrenches are accumulated.

`trick_units(-)`

Definition at line 205 of file `structure_integrated_dyn_body.hh`.

Referenced by `collect_forces_and_torques()`, `collect_local_forces_and_torques()`, and `PropagateForcesAndTorques()`.

#### 8.15.5.3 **BodyWrenchCollect** `jeod::StructureIntegratedDynBody::effector_wrench_collection`

Collection of effector wrenches.

The effector wrenches are assembled into the collection at the `S_define` level via

```
vcollect_containing_body.effector_wrench_collection.collect_wrench {
    pointer_to_wrench1,
    ...
    pointer_to_wrench_n
};
```

The collected effector wrenches are processed by the `collect_forces_and_torques` member function.

Note: For completion, there probably should be collected environmental and non-transmitted wrenches as well as effector wrenches.`trick_units(-)`

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

Referenced by `collect_local_forces_and_torques()`.

#### 8.15.5.4 **double** `jeod::StructureIntegratedDynBody::inertial_accel_inrtl[3]` [protected]

Inertial-referenced inertial acceleration at the structure frame origin.

`trick_units(m/s2)`

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

Referenced by `complete_translational_acceleration()`.

#### 8.15.5.5 **double** `jeod::StructureIntegratedDynBody::inertial_accel_struct[3]` [protected]

Structure-referenced inertial acceleration at the structure frame origin.

`trick_units(m/s2)`

Definition at line 237 of file `structure_integrated_dyn_body.hh`.

Referenced by `complete_translational_acceleration()`.



**8.15.5.6** `double jeod::StructureIntegratedDynBody::inertial_accel_struct_omega[3]` `[protected]`

Structure-referenced inertial acceleration at the structure frame origin due to vehicle angular velocity.

trick\_units(m/s<sup>2</sup>)

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

Referenced by `complete_translational_acceleration()`, and `compute_translational_acceleration()`.

**8.15.5.7** `double jeod::StructureIntegratedDynBody::inertial_accel_struct_omega_dot[3]` `[protected]`

Structure-referenced inertial acceleration at the structure frame origin due to vehicle angular acceleration.

trick\_units(m/s<sup>2</sup>)

Definition at line 232 of file `structure_integrated_dyn_body.hh`.

Referenced by `complete_translational_acceleration()`.

**8.15.5.8** `VehicleNonGravState jeod::StructureIntegratedDynBody::non_grav_state` `[protected]`

Rotational and translational behaviors, for the constraints solver.

trick\_units(—)

Definition at line 220 of file `structure_integrated_dyn_body.hh`.

Referenced by `solve_constraints()`.

**8.15.5.9** `FrameDerivs jeod::StructureIntegratedDynBody::struct_derivs` `[protected]`

Translational/rotational accelerations of the structural frame.

trick\_units(—)

Definition at line 210 of file `structure_integrated_dyn_body.hh`.

Referenced by `collect_forces_and_torques()`, `complete_translational_acceleration()`, `compute_rotational_acceleration()`, `compute_vehicle_point_derivatives()`, `rot_integ()`, `solve_constraints()`, and `trans_integ()`.

**8.15.5.10** `VehicleProperties jeod::StructureIntegratedDynBody::vehicle_properties` `[protected]`

Various properties of the vehicle, for the constraints solver.

trick\_units(—)

Definition at line 215 of file `structure_integrated_dyn_body.hh`.

Referenced by `attach_update_properties()`, `detach()`, `get_vehicle_properties()`, and `solve_constraints()`.

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

- [structure\\_integrated\\_dyn\\_body.hh](#)
- [structure\\_integrated\\_dyn\\_body.cc](#)
- [structure\\_integrated\\_dyn\\_body\\_collect.cc](#)
- [structure\\_integrated\\_dyn\\_body\\_integration.cc](#)
- [structure\\_integrated\\_dyn\\_body\\_pt\\_accel.cc](#)
- [structure\\_integrated\\_dyn\\_body\\_solve.cc](#)

## 8.16 jeod::Torque Class Reference

A [Torque](#) represents a Newtonian torque that acts on a [DynBody](#).

```
#include <torque.hh>
```

### Public Member Functions

- [Torque](#) ()  
*Torque default constructor.*
- virtual [~Torque](#) ()  
*Torque destructor.*
- double & [operator\[\]](#) (const unsigned int index)  
*Access a torque element, non-const version.*
- double [operator\[\]](#) (const unsigned int index) const  
*Access a torque element, const version.*

### Data Fields

- bool [active](#)  
*Is this torque active?*
- double [torque](#) [3]  
*Torque vector.*

### Private Member Functions

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

#### 8.16.1 Detailed Description

A [Torque](#) represents a Newtonian torque that acts on a [DynBody](#).

The class encapsulates an active flag and a 3-vector that contains the torque components. Torques are collected in one of a [DynBody](#) object's torque collection STL vectors. The torque vector is expressed in the structural frame of that [DynBody](#) object.

The [Torque](#) class is the recommended mechanism for representing torques in JEOD. While 3-vectors can also be collected into a collect STL vector, there is no way to turn off these collected 3-vectors. Even worse, there is no way to tell whether a collected 3-vector does indeed represent a torque, or even if it is a 3-vector. In comparison, [Torque](#) objects can be turned on and off, and more importantly, they are type-safe.

Definition at line 82 of file torque.hh.

#### 8.16.2 Constructor & Destructor Documentation

##### 8.16.2.1 jeod::Torque::Torque ( void )

[Torque](#) default constructor.

Definition at line 44 of file torque.cc.

References [torque](#).

**8.16.2.2** `jeod::Torque::~~Torque ( void ) [virtual]`

[Torque](#) destructor.

Definition at line 56 of file torque.cc.

**8.16.2.3** `jeod::Torque::Torque ( const Torque & ) [private]`

Not implemented.

**8.16.3 Member Function Documentation****8.16.3.1** `Torque& jeod::Torque::operator= ( const Torque & ) [private]`

Not implemented.

**8.16.3.2** `double & jeod::Torque::operator[] ( const unsigned int index ) [inline]`

Access a torque element, non-const version.

**Returns**

[Torque](#) component at specified index  
Units: NM

**Parameters**

<i>in</i>	<i>index</i>	Index number
-----------	--------------	--------------

Definition at line 76 of file torque\_inline.hh.

References [torque](#).

**8.16.3.3** `double jeod::Torque::operator[] ( const unsigned int index ) const [inline]`

Access a torque element, const version.

**Returns**

[Torque](#) component at specified index  
Units: NM

**Parameters**

<i>in</i>	<i>index</i>	Index number
-----------	--------------	--------------

Definition at line 89 of file torque\_inline.hh.

References [torque](#).

**8.16.4 Field Documentation****8.16.4.1** `bool jeod::Torque::active`

Is this torque active?

`trick_units(-)`

Definition at line 97 of file torque.hh.

#### 8.16.4.2 double jeod::Torque::torque[3]

[Torque](#) vector.

trick\_units(N\*m)

Definition at line 101 of file torque.hh.

Referenced by operator[](), and Torque().

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

- [torque.hh](#)
- [torque\\_inline.hh](#)
- [torque.cc](#)

## 8.17 jeod::VehicleNonGravState Class Reference

Encapsulates various aspects of a vehicle's state with respect to inertial.

```
#include <vehicle_non_grav_state.hh>
```

### Data Fields

- double [omega\\_body](#) [3]  
*Vehicle angular velocity with respect to inertial, in root body body frame coordinates.*
- double [omega\\_struct](#) [3]  
*Vehicle angular velocity with respect to inertial, in root body structural frame coordinates.*
- double [omega\\_dot\\_body](#) [3]  
*Vehicle angular acceleration with respect to inertial, in root body body frame coordinates.*
- double [omega\\_dot\\_struct](#) [3]  
*Vehicle angular acceleration with respect to inertial, in root body structural frame coordinates.*
- double [inertial\\_torque\\_struct](#) [3]  
*Vehicle inertial torque ( $w \times lw$ ) in root body structural coordinates.*
- double [accel\\_struct](#) [3]  
*Vehicle non-gravitational translational acceleration at the center of mass, in root body structural frame coordinates.*

### Friends

- class [InputProcessor](#)
- void [init\\_attrjeod\\_\\_VehicleNonGravState](#) ()

#### 8.17.1 Detailed Description

Encapsulates various aspects of a vehicle's state with respect to inertial.

Definition at line 67 of file vehicle\_non\_grav\_state.hh.

#### 8.17.2 Friends And Related Function Documentation

8.17.2.1 void [init\\_attrjeod\\_\\_VehicleNonGravState](#) ( ) [[friend](#)]

8.17.2.2 friend class [InputProcessor](#) [[friend](#)]

Definition at line 69 of file vehicle\_non\_grav\_state.hh.

### 8.17.3 Field Documentation

#### 8.17.3.1 double jeod::VehicleNonGravState::accel\_struct[3]

Vehicle non-gravitational translational acceleration at the center of mass, in root body structural frame coordinates.

trick\_units( $\text{m/s}^2$ )

Definition at line 106 of file vehicle\_non\_grav\_state.hh.

Referenced by jeod::StructureIntegratedDynBody::solve\_constraints().

#### 8.17.3.2 double jeod::VehicleNonGravState::inertial\_torque\_struct[3]

Vehicle inertial torque ( $w \times lw$ ) in root body structural coordinates.

trick\_units( $\text{N}\cdot\text{m}$ )

Definition at line 100 of file vehicle\_non\_grav\_state.hh.

Referenced by jeod::StructureIntegratedDynBody::solve\_constraints().

#### 8.17.3.3 double jeod::VehicleNonGravState::omega\_body[3]

Vehicle angular velocity with respect to inertial, in root body body frame coordinates.

trick\_units( $1/\text{s}$ )

Definition at line 77 of file vehicle\_non\_grav\_state.hh.

Referenced by jeod::StructureIntegratedDynBody::solve\_constraints().

#### 8.17.3.4 double jeod::VehicleNonGravState::omega\_dot\_body[3]

Vehicle angular acceleration with respect to inertial, in root body body frame coordinates.

trick\_units( $1/\text{s}^2$ )

Definition at line 89 of file vehicle\_non\_grav\_state.hh.

Referenced by jeod::StructureIntegratedDynBody::solve\_constraints().

#### 8.17.3.5 double jeod::VehicleNonGravState::omega\_dot\_struct[3]

Vehicle angular acceleration with respect to inertial, in root body structural frame coordinates.

trick\_units( $1/\text{s}^2$ )

Definition at line 95 of file vehicle\_non\_grav\_state.hh.

Referenced by jeod::StructureIntegratedDynBody::solve\_constraints().

#### 8.17.3.6 double jeod::VehicleNonGravState::omega\_struct[3]

Vehicle angular velocity with respect to inertial, in root body structural frame coordinates.

trick\_units( $1/\text{s}$ )

Definition at line 83 of file vehicle\_non\_grav\_state.hh.

Referenced by jeod::StructureIntegratedDynBody::solve\_constraints().

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

- [vehicle\\_non\\_grav\\_state.hh](#)

## 8.18 jeod::VehicleProperties Class Reference

Captures pointers to various vehicle properties that are commonly used in the constraint concept.

```
#include <vehicle_properties.hh>
```

### Public Member Functions

- [VehicleProperties](#) ()  
*Default constructor, for use by Trick only.*
- [VehicleProperties](#) (SolverTypes::Vector3RefT parent\_to\_structure\_offset\_in, SolverTypes::Matrix3x3RefT parent\_to\_structure\_transform\_in, double &mass\_in, SolverTypes::Vector3RefT structure\_to\_body\_offset\_in, SolverTypes::Matrix3x3RefT inertia\_in, SolverTypes::Matrix3x3RefT structure\_to\_body\_transform\_in, double &inverse\_mass\_in, SolverTypes::Matrix3x3RefT inverse\_inertia\_in)  
*Non-default constructor that sets all elements.*
- SolverTypes::ConstDecayedVector3T [get\\_parent\\_to\\_structure\\_offset](#) () const
- SolverTypes::ConstMatrix3x3RefT [get\\_parent\\_to\\_structure\\_transform](#) () const
- double [get\\_mass](#) () const
- SolverTypes::ConstDecayedVector3T [get\\_structure\\_to\\_body\\_offset](#) () const
- SolverTypes::ConstMatrix3x3RefT [get\\_inertia](#) () const
- SolverTypes::Matrix3x3RefT [get\\_structure\\_to\\_body\\_transform](#) () const
- double [get\\_inverse\\_mass](#) () const
- SolverTypes::Matrix3x3RefT [get\\_inverse\\_inertia](#) () const

### Private Attributes

- SolverTypes::Vector3PointerT [parent\\_to\\_structure\\_offset](#)  
*Pointer to the vehicle's structure\_point.position vector.*
- SolverTypes::Matrix3x3PointerT [parent\\_to\\_structure\\_transform](#)  
*Pointer to the vehicle's structure\_point.T\_parent\_this matrix.*
- double \* [mass](#)  
*Pointer to the vehicle's composite\_properties.mass member.*
- SolverTypes::Vector3PointerT [structure\\_to\\_body\\_offset](#)  
*Pointer to the vehicle's composite\_properties.position vector.*
- SolverTypes::Matrix3x3PointerT [inertia](#)  
*Pointer to the vehicle's composite\_properties.inertia tensor.*
- SolverTypes::Matrix3x3PointerT [structure\\_to\\_body\\_transform](#)  
*Pointer to the vehicle's composite\_properties.T\_parent\_this matrix.*
- double \* [inverse\\_mass](#)  
*Pointer to the vehicle's inverse\_mass member.*
- SolverTypes::Matrix3x3PointerT [inverse\\_inertia](#)  
*Pointer to the vehicle's inverse\_inertia member.*

### Friends

- class [InputProcessor](#)
- void [init\\_attrjeod\\_\\_VehicleProperties](#) ()

### 8.18.1 Detailed Description

Captures pointers to various vehicle properties that are commonly used in the constraint concept.

As this is potentially quite dangerous, access to the captured members is limited to const getters.

This class is not designed for extensibility.

Definition at line 73 of file vehicle\_properties.hh.

### 8.18.2 Constructor & Destructor Documentation

#### 8.18.2.1 jeod::VehicleProperties::VehicleProperties ( ) `[inline]`

Default constructor, for use by Trick only.

Definition at line 89 of file vehicle\_properties.hh.

#### 8.18.2.2 jeod::VehicleProperties::VehicleProperties ( SolverTypes::Vector3RefT *parent\_to\_structure\_offset\_in*, SolverTypes::Matrix3x3RefT *parent\_to\_structure\_transform\_in*, double & *mass\_in*, SolverTypes::Vector3RefT *structure\_to\_body\_offset\_in*, SolverTypes::Matrix3x3RefT *inertia\_in*, SolverTypes::Matrix3x3RefT *structure\_to\_body\_transform\_in*, double & *inverse\_mass\_in*, SolverTypes::Matrix3x3RefT *inverse\_inertia\_in* ) `[inline]`

Non-default constructor that sets all elements.

Parameters

<i>parent_to_structure_offset_in</i>	Reference to the vehicle's structure_point.position vector.
<i>parent_to_structure_transform_in</i>	Reference to the vehicle's structure_point.T_parent_this matrix.
<i>mass_in</i>	Reference to the vehicle's composite_properties.mass member.
<i>structure_to_body_offset_in</i>	Reference to the vehicle's composite_properties.position vector.
<i>inertia_in</i>	Reference to the vehicle's composite_properties.inertia tensor.
<i>structure_to_body_transform_in</i>	Reference to the vehicle's composite_properties.T_parent_this matrix.
<i>inverse_mass_in</i>	Reference to the vehicle's inverse_mass member.
<i>inverse_inertia_in</i>	Reference to the vehicle's inverse_inertia member.

Definition at line 121 of file vehicle\_properties.hh.

### 8.18.3 Member Function Documentation

#### 8.18.3.1 SolverTypes::ConstMatrix3x3RefT jeod::VehicleProperties::get\_inertia ( ) const `[inline]`

Returns

Const reference to the vehicle's inertia tensor, in vehicle body frame coordinates.

Definition at line 190 of file vehicle\_properties.hh.

References inertia.

**8.18.3.2** `SolverTypes::Matrix3x3RefT jeod::VehicleProperties::get_inverse_inertia ( ) const` `[inline]`

#### Returns

Const reference to the inverse of the vehicle's inertia tensor, in vehicle body frame coordinates.

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

References `inverse_inertia`.

**8.18.3.3** `double jeod::VehicleProperties::get_inverse_mass ( ) const` `[inline]`

#### Returns

The multiplicative inverse of the vehicle's mass.

Definition at line 207 of file `vehicle_properties.hh`.

References `inverse_mass`.

**8.18.3.4** `double jeod::VehicleProperties::get_mass ( ) const` `[inline]`

#### Returns

The vehicle mass.

Definition at line 171 of file `vehicle_properties.hh`.

References `mass`.

**8.18.3.5** `SolverTypes::ConstDecayedVector3T jeod::VehicleProperties::get_parent_to_structure_offset ( ) const` `[inline]`

#### Returns

Const reference to the offset from the parent vehicle's structural frame origin to this vehicle's structural origin, in parent structural coordinates.

Definition at line 154 of file `vehicle_properties.hh`.

References `parent_to_structure_offset`.

**8.18.3.6** `SolverTypes::ConstMatrix3x3RefT jeod::VehicleProperties::get_parent_to_structure_transform ( ) const` `[inline]`

#### Returns

Const reference to the transformation matrix from the parent vehicle's structural frame to this vehicle's structural frame.

Definition at line 163 of file `vehicle_properties.hh`.

References `parent_to_structure_transform`.

**8.18.3.7** `SolverTypes::ConstDecayedVector3T jeod::VehicleProperties::get_structure_to_body_offset ( ) const` `[inline]`



**Returns**

Const reference to the offset from the origin of the vehicle's structural frame to the vehicle's center of mass, in vehicle structural coordinates.

Definition at line 181 of file vehicle\_properties.hh.

References structure\_to\_body\_offset.

#### 8.18.3.8 SolverTypes::Matrix3x3RefT jeod::VehicleProperties::get\_structure\_to\_body\_transform ( ) const [inline]

**Returns**

Const reference to the transformation matrix from the vehicle's structural frame to its body frame.

Definition at line 199 of file vehicle\_properties.hh.

References structure\_to\_body\_transform.

### 8.18.4 Friends And Related Function Documentation

#### 8.18.4.1 void init\_attrjeod\_\_VehicleProperties ( ) [friend]

#### 8.18.4.2 friend class InputProcessor [friend]

Definition at line 79 of file vehicle\_properties.hh.

### 8.18.5 Field Documentation

#### 8.18.5.1 SolverTypes::Matrix3x3PointerT jeod::VehicleProperties::inertia [private]

Pointer to the vehicle's composite\_properties.inertia tensor.

trick\_units( $m^2 \cdot kg$ )

Definition at line 248 of file vehicle\_properties.hh.

Referenced by get\_inertia().

#### 8.18.5.2 SolverTypes::Matrix3x3PointerT jeod::VehicleProperties::inverse\_inertia [private]

Pointer to the vehicle's inverse\_inertia member.

trick\_units( $1/kg \cdot m^2$ )

Definition at line 263 of file vehicle\_properties.hh.

Referenced by get\_inverse\_inertia().

#### 8.18.5.3 double\* jeod::VehicleProperties::inverse\_mass [private]

Pointer to the vehicle's inverse\_mass member.

trick\_units( $1/kg$ )

Definition at line 258 of file vehicle\_properties.hh.

Referenced by get\_inverse\_mass().

**8.18.5.4** `double* jeod::VehicleProperties::mass` `[private]`

Pointer to the vehicle's `composite_properties.mass` member.

`trick_units(kg)`

Definition at line 238 of file `vehicle_properties.hh`.

Referenced by `get_mass()`.

**8.18.5.5** `SolverTypes::Vector3PointerT jeod::VehicleProperties::parent_to_structure_offset` `[private]`

Pointer to the vehicle's `structure_point.position` vector.

`trick_units(m)`

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

Referenced by `get_parent_to_structure_offset()`.

**8.18.5.6** `SolverTypes::Matrix3x3PointerT jeod::VehicleProperties::parent_to_structure_transform` `[private]`

Pointer to the vehicle's `structure_point.T_parent_this` matrix.

`trick_units(-)`

Definition at line 233 of file `vehicle_properties.hh`.

Referenced by `get_parent_to_structure_transform()`.

**8.18.5.7** `SolverTypes::Vector3PointerT jeod::VehicleProperties::structure_to_body_offset` `[private]`

Pointer to the vehicle's `composite_properties.position` vector.

`trick_units(m)`

Definition at line 243 of file `vehicle_properties.hh`.

Referenced by `get_structure_to_body_offset()`.

**8.18.5.8** `SolverTypes::Matrix3x3PointerT jeod::VehicleProperties::structure_to_body_transform` `[private]`

Pointer to the vehicle's `composite_properties.T_parent_this` matrix.

`trick_units(-)`

Definition at line 253 of file `vehicle_properties.hh`.

Referenced by `get_structure_to_body_transform()`.

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

- [vehicle\\_properties.hh](#)

## 8.19 jeod::Wrench Class Reference

A wrench comprises a torque and a force applied at a point on a [DynBody](#).

```
#include <wrench.hh>
```

## Public Member Functions

- [Wrench](#) (bool active\_in=true)  
*Default constructor.*
- [Wrench](#) (const double torque\_in[3], const double force\_in[3], const double point\_in[3], bool active\_in=true)  
*Non-default constructor that sets all elements of the wrench.*
- [Wrench](#) (const double point\_in[3], bool active\_in=true)  
*Non-default constructor that sets the point and active flag.*
- virtual [~Wrench](#) ()=default  
*Destructor.*
- [Wrench](#) (const [Wrench](#) &)=default  
*Copy constructor.*
- [Wrench](#) & [operator=](#) (const [Wrench](#) &)=default  
*Copy assignment operator.*
- [Wrench](#) ([Wrench](#) &&)=default  
*Move constructor.*
- [Wrench](#) & [operator=](#) ([Wrench](#) &&)=default  
*Move assignment operator.*
- [Wrench](#) & [operator+=](#) (const [Wrench](#) &other)  
*Increment this wrench by the other, but only if both are active.*
- void [activate](#) ()  
*Mark this wrench as active.*
- void [deactivate](#) ()  
*Mark this wrench as inactive.*
- bool [is\\_active](#) () const  
*Is this wrench active?*
- void [reset\\_force\\_and\\_torque](#) ()  
*Set the force and torque to zero.*
- void [reset\\_torque](#) ()  
*Set the torque to zero.*
- void [reset\\_force](#) ()  
*Set the force to zero.*
- void [reset\\_point](#) ()  
*Set the point to zero.*
- void [set](#) (const double torque\_in[3], const double force\_in[3], const double point\_in[3])  
*Set all vector elements of the wrench.*
- void [set\\_torque](#) (const double torque\_in[3])  
*Set the torque to the specified value.*
- void [set\\_force](#) (const double force\_in[3])  
*Set the force to the specified value.*
- void [set\\_force](#) (const double force\_in[3], const double point\_in[3])  
*Set the force and the point of application to the specified values.*
- void [set\\_point](#) (const double point\_in[3])  
*Set the point of application to the specified value.*
- void [scale\\_torque](#) (double scale)  
*Scale the torque by the specified value.*
- void [scale\\_force](#) (double scale)  
*Scale the force by the specified value.*
- const double \* [get\\_torque](#) () const  
*Const getter of the torque vector.*
- const double \* [get\\_force](#) () const

- *Const getter of the force vector.*
- `const double * get\_point () const`
- *Const getter of the point vector.*
- `Wrench & accumulate (const std::vector< Wrench * > &collection)`
- *Accumulate the wrenches in the collection to form a combined wrench about the current wrench point, which remains unchanged.*
- `Wrench & accumulate (const std::vector< Wrench * > &collection, const double new_point[3])`
- *Accumulate the wrenches in the collection to form a combined wrench about the specified wrench point.*
- `Wrench transform\_to\_point (const double new_point[3]) const`
- *Construct an equivalent [Wrench](#) about the specified point.*
- `Wrench transform\_to\_parent (const MassPointState &point_state) const`
- *Construct an equivalent [Wrench](#) about the current point, but in a different reference frame.*

## Private Attributes

- `double torque [3]`
- *The torque exerted on the [DynBody](#) by the force/torque agent, expressed in structural coordinates.*
- `double force [3]`
- *The force exerted on the [DynBody](#) by the force/torque agent, expressed in structural coordinates.*
- `double point [3]`
- *The structural coordinates of the point at which the force is applied.*
- `bool active`
- *Indicated whether the wrench is active (true) or inactive (false).*

## Friends

- `class InputProcessor`
- `void init\_attrjeod\_\_Wrench ()`

### 8.19.1 Detailed Description

A wrench comprises a torque and a force applied at a point on a [DynBody](#).

The torque should not include the torque due to the application of the force.

A Trick simulation issues vcollect statements such as

```
vcollect vehicle.dyn_body.collect_wrench.collection
{
    wrench_model1.wrench,
    wrench_model2.wrench
};
```

Definition at line 81 of file wrench.hh.

### 8.19.2 Constructor & Destructor Documentation

#### 8.19.2.1 `jeod::Wrench::Wrench ( bool active_in = true )` `[inline]`, `[explicit]`

Default constructor.

The wrench is marked as active, and the torque, force, and point vectors are all initialized to zero. This constructor can also be used as a non-default constructor that marks the wrench as inactive by calling it with one argument (a boolean) whose value is false.

## Parameters

<i>active_in</i>	True (default) indicates the wrench is active.
------------------	--

Definition at line 97 of file wrench.hh.

References force, point, and torque.

**8.19.2.2** `jeod::Wrench::Wrench ( const double torque_in[3], const double force_in[3], const double point_in[3], bool active_in =true ) [inline],[explicit]`

Non-default constructor that sets all elements of the wrench.

## Parameters

<i>torque_in</i>	The intrinsic torque for this wrench.
<i>force_in</i>	The force applied at the point.
<i>point_in</i>	The point at which forces are applied.
<i>active_in</i>	True (default) indicates the wrench is active.

Definition at line 114 of file wrench.hh.

References force, point, and torque.

**8.19.2.3** `jeod::Wrench::Wrench ( const double point_in[3], bool active_in =true ) [inline],[explicit]`

Non-default constructor that sets the point and active flag.

The torque and force and initialized to zero.

## Parameters

<i>point_in</i>	The point at which forces are applied.
<i>active_in</i>	True (default) indicates the wrench is active.

Definition at line 133 of file wrench.hh.

References force, point, and torque.

**8.19.2.4** `virtual jeod::Wrench::~~Wrench ( ) [virtual],[default]`

Destructor.

**8.19.2.5** `jeod::Wrench::Wrench ( const Wrench & ) [default]`

Copy constructor.

**8.19.2.6** `jeod::Wrench::Wrench ( Wrench && ) [default]`

Move constructor.

## 8.19.3 Member Function Documentation

**8.19.3.1** `Wrench& jeod::Wrench::accumulate ( const std::vector< Wrench * > & collection ) [inline]`

Accumulate the wrenches in the collection to form a combined wrench about the current wrench point, which remains unchanged.

**Parameters**

<i>collection</i>	The wrenches to be accumulated.
-------------------	---------------------------------

Definition at line 372 of file wrench.hh.

References `reset_force_and_torque()`.

Referenced by `jeod::BodyWrenchCollect::accumulate()`, and `accumulate()`.

**8.19.3.2** `Wrench& jeod::Wrench::accumulate ( const std::vector< Wrench * > & collection, const double new_point[3] )`  
`[inline]`

Accumulate the wrenches in the collection to form a combined wrench about the specified wrench point.

**Parameters**

<i>collection</i>	The wrenches to be accumulated.
<i>new_point</i>	The point about which the wrenches to be accumulated.

Definition at line 390 of file wrench.hh.

References `accumulate()`, and `set_point()`.

**8.19.3.3** `void jeod::Wrench::activate ( )` `[inline]`

Mark this wrench as active.

Definition at line 198 of file wrench.hh.

References `active`.

**8.19.3.4** `void jeod::Wrench::deactivate ( )` `[inline]`

Mark this wrench as inactive.

Definition at line 207 of file wrench.hh.

References `active`.

**8.19.3.5** `const double* jeod::Wrench::get_force ( ) const` `[inline]`

Const getter of the force vector.

Definition at line 352 of file wrench.hh.

References `force`.

Referenced by `jeod::StructureIntegratedDynBody::collect_forces_and_torques()`.

**8.19.3.6** `const double* jeod::Wrench::get_point ( ) const` `[inline]`

Const getter of the point vector.

Definition at line 361 of file wrench.hh.

References `point`.

**8.19.3.7** `const double* jeod::Wrench::get_torque ( ) const` `[inline]`

Const getter of the torque vector.

Definition at line 343 of file wrench.hh.

References torque.

Referenced by jeod::StructureIntegratedDynBody::collect\_forces\_and\_torques().

#### 8.19.3.8 bool jeod::Wrench::is\_active( void ) const [inline]

Is this wrench active?

Definition at line 216 of file wrench.hh.

References active.

#### 8.19.3.9 Wrench& jeod::Wrench::operator+=( const Wrench & other ) [inline]

Increment this wrench by the other, but only if both are active.

The other wrench is effectively reseated to this wrench's point prior to incrementing.

Parameters

<i>other</i>	Wrench with which this wrench is to be incremented.
--------------	---

Returns

\*this.

Definition at line 180 of file wrench.hh.

References active, force, point, and torque.

#### 8.19.3.10 Wrench& jeod::Wrench::operator=( const Wrench & ) [default]

Copy assignment operator.

#### 8.19.3.11 Wrench& jeod::Wrench::operator=( Wrench && ) [default]

Move assignment operator.

#### 8.19.3.12 void jeod::Wrench::reset\_force( ) [inline]

Set the force to zero.

The torque and point remain unaltered.

Definition at line 244 of file wrench.hh.

References force.

#### 8.19.3.13 void jeod::Wrench::reset\_force\_and\_torque( ) [inline]

Set the force and torque to zero.

The point remains unaltered.

Definition at line 225 of file wrench.hh.

References force, and torque.

Referenced by accumulate(), and jeod::StructureIntegratedDynBody::collect\_local\_forces\_and\_torques().

#### 8.19.3.14 void jeod::Wrench::reset\_point ( ) [inline]

Set the point to zero.

The torque and force remain unaltered.

Definition at line 253 of file wrench.hh.

References point.

#### 8.19.3.15 void jeod::Wrench::reset\_torque ( ) [inline]

Set the torque to zero.

The force and point remain unaltered.

Definition at line 235 of file wrench.hh.

References torque.

#### 8.19.3.16 void jeod::Wrench::scale\_force ( double *scale* ) [inline]

Scale the force by the specified value.

The torque and point of application remain unchanged.

Definition at line 334 of file wrench.hh.

References force.

#### 8.19.3.17 void jeod::Wrench::scale\_torque ( double *scale* ) [inline]

Scale the torque by the specified value.

The force and point of application remain unaltered.

Definition at line 324 of file wrench.hh.

References torque.

#### 8.19.3.18 void jeod::Wrench::set ( const double *torque\_in*[3], const double *force\_in*[3], const double *point\_in*[3] ) [inline]

Set all vector elements of the wrench.

Parameters

<i>torque_in</i>	The intrinsic torque for this wrench.
<i>force_in</i>	The force applied at the point.
<i>point_in</i>	The point at which forces are applied.

Definition at line 265 of file wrench.hh.

References force, point, and torque.

#### 8.19.3.19 void jeod::Wrench::set\_force ( const double *force\_in*[3] ) [inline]

Set the force to the specified value.

The torque and point of application remain unchanged.

Definition at line 290 of file wrench.hh.

References force.



8.19.3.20 `void jeod::Wrench::set_force ( const double force_in[3], const double point_in[3] ) [inline]`

Set the force and the point of application to the specified values.

The torque remain unchanged.

Definition at line 300 of file `wrench.hh`.

References `force`, and `point`.

8.19.3.21 `void jeod::Wrench::set_point ( const double point_in[3] ) [inline]`

Set the point of application to the specified value.

The force and torque remain unchanged.

Definition at line 313 of file `wrench.hh`.

References `point`.

Referenced by `jeod::BodyWrenchCollect::accumulate()`, and `accumulate()`.

8.19.3.22 `void jeod::Wrench::set_torque ( const double torque_in[3] ) [inline]`

Set the torque to the specified value.

The force and point of application remain unaltered.

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

References `torque`.

8.19.3.23 `Wrench jeod::Wrench::transform_to_parent ( const MassPointState & point_state ) const [inline]`

Construct an equivalent [Wrench](#) about the current point, but in a different reference frame.

Parameters

<i>point_state</i>	Contains the position and orientation of the current frame in the parent frame.
--------------------	---

Returns

Equivalent wrench in the parent frame.

Definition at line 421 of file `wrench.hh`.

References `force`, `point`, and `torque`.

Referenced by `jeod::StructureIntegratedDynBody::PropagateForcesAndTorques()`.

8.19.3.24 `Wrench jeod::Wrench::transform_to_point ( const double new_point[3] ) const [inline]`

Construct an equivalent [Wrench](#) about the specified point.

Parameters

<i>new_point</i>	The point about which this is to be represented.
------------------	--

**Returns**

Equivalent wrench about the specified point.

Definition at line 404 of file wrench.hh.

References active, force, point, and torque.

Referenced by jeod::StructureIntegratedDynBody::collect\_forces\_and\_torques().

**8.19.4 Friends And Related Function Documentation**

**8.19.4.1** `void init_attrjeod__Wrench ( )` `[friend]`

**8.19.4.2** `friend class InputProcessor` `[friend]`

Definition at line 83 of file wrench.hh.

**8.19.5 Field Documentation**

**8.19.5.1** `bool jeod::Wrench::active` `[private]`

Indicated whether the wrench is active (true) or inactive (false).

inactive wrenches are not collected.trick\_units(-)

Definition at line 466 of file wrench.hh.

Referenced by activate(), deactivate(), is\_active(), operator+=( ), and transform\_to\_point().

**8.19.5.2** `double jeod::Wrench::force[3]` `[private]`

The force exerted on the [DynBody](#) by the force/torque agent, expressed in structural coordinates.

trick\_units(N)

Definition at line 455 of file wrench.hh.

Referenced by get\_force(), operator+=( ), reset\_force(), reset\_force\_and\_torque(), scale\_force(), set(), set\_force(), transform\_to\_parent(), transform\_to\_point(), and Wrench().

**8.19.5.3** `double jeod::Wrench::point[3]` `[private]`

The structural coordinates of the point at which the force is applied.

trick\_units(m)

Definition at line 460 of file wrench.hh.

Referenced by get\_point(), operator+=( ), reset\_point(), set(), set\_force(), set\_point(), transform\_to\_parent(), transform\_to\_point(), and Wrench().

**8.19.5.4** `double jeod::Wrench::torque[3]` `[private]`

The torque exerted on the [DynBody](#) by the force/torque agent, expressed in structural coordinates.

This torque should not include the torque that results from the force not passing through the center of mass. A typical thruster, for example, should have the torque set to zero. On the other hand, a Hall effect thruster will have a non-zero torque due to the swirling of the exhaust.trick\_units(N\*m)

Definition at line 449 of file wrench.hh.

Referenced by `get_torque()`, `operator+=()`, `reset_force_and_torque()`, `reset_torque()`, `scale_torque()`, `set()`, `set_torque()`, `transform_to_parent()`, `transform_to_point()`, and `Wrench()`.

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

- [wrench.hh](#)



## Chapter 9

# File Documentation

### 9.1 `aux_classes.cc` File Reference

Define base methods for various small JEOD DynBody classes.

```
#include "utils/math/include/vector3.hh"
#include "../include/body_force_collect.hh"
#include "../include/frame_derivs.hh"
```

#### Namespaces

- [jeod](#)  
*Namespace jeod.*

#### 9.1.1 Detailed Description

Define base methods for various small JEOD DynBody classes.

Definition in file [aux\\_classes.cc](#).

### 9.2 `body_force_collect.hh` File Reference

Define the class BodyForceCollect.

```
#include "utils/container/include/pointer_vector.hh"
#include "utils/memory/include/jeod_alloc.hh"
#include "force.hh"
#include "torque.hh"
```

#### Data Structures

- class [jeod::JPVCollectForce](#)  
*This is a derived version of the template class `JeodPointerVector<CollectForce>::type` with an implementation of the method `perform_cleanup_action` which frees and clears stale data following a restore.*
- class [jeod::JPVCollectTorque](#)  
*This is a derived version of the template class `JeodPointerVector<CollectTorque>::type` with an implementation of the method `perform_cleanup_action` which frees and clears stale data following a restore.*

- class [jeod::BodyForceCollect](#)

*Serves as the collection point for forces and torques that act on a vehicle.*

## Namespaces

- [jeod](#)

*Namespace jeod.*

## Functions

- template<class CollectType >  
void [jeod::release\\_vector](#) (CollectType &vec)  
*Release JEOD-allocated memory in the collect vector.*
- template<typename CollectType , typename value\_type >  
void [jeod::collect\\_insert](#) (CollectType &collect\_in, value\_type &elem)
- template<typename CollectType , typename value\_type >  
void [jeod::collect\\_push\\_back](#) (CollectType &collect\_in, value\_type &elem)

### 9.2.1 Detailed Description

Define the class BodyForceCollect.

Definition in file [body\\_force\\_collect.hh](#).

## 9.3 body\_ref\_frame.hh File Reference

Define the class BodyRefFrame.

```
#include <cstdint>
#include "dynamics/mass/include/class_declarations.hh"
#include "utils/sim_interface/include/jeod_class.hh"
#include "utils/ref_frames/include/ref_frame.hh"
#include "utils/ref_frames/include/ref_frame_items.hh"
```

## Data Structures

- class [jeod::BodyRefFrame](#)

*Extend RefFrame to add coupling between the reference frame tree and the mass tree and to keep track of which state items have been set.*

## Namespaces

- [jeod](#)

*Namespace jeod.*

### 9.3.1 Detailed Description

Define the class BodyRefFrame.

Definition in file [body\\_ref\\_frame.hh](#).

## 9.4 body\_wrench\_collect.cc File Reference

Define BodyWrenchCollect member functions.

```
#include "../include/body_wrench_collect.hh"  
#include "utils/memory/include/jeod_alloc.hh"
```

### Namespaces

- [jeod](#)  
*Namespace jeod.*

#### 9.4.1 Detailed Description

Define BodyWrenchCollect member functions.

Definition in file [body\\_wrench\\_collect.cc](#).

## 9.5 body\_wrench\_collect.hh File Reference

Defines the class BodyWrenchCollect.

```
#include "wrench.hh"  
#include "utils/container/include/pointer_vector.hh"
```

### Data Structures

- class [jeod::BodyWrenchCollect](#)  
*Serves as the collection point for wrenches that act on a vehicle.*

### Namespaces

- [jeod](#)  
*Namespace jeod.*

#### 9.5.1 Detailed Description

Defines the class BodyWrenchCollect.

Definition in file [body\\_wrench\\_collect.hh](#).

## 9.6 class\_declarations.hh File Reference

Forward declarations of classes defined in [dyn\\_body.hh](#).

### Namespaces

- [jeod](#)  
*Namespace jeod.*

### 9.6.1 Detailed Description

Forward declarations of classes defined in [dyn\\_body.hh](#).

Definition in file [class\\_declarations.hh](#).

## 9.7 dyn\_body.cc File Reference

Define base methods for the DynBody class.

```
#include <cstdint>
#include <algorithm>
#include "dynamics/dyn_manager/include/dyn_manager.hh"
#include "dynamics/dyn_manager/include/dynamics_integration_group.hh"
#include "utils/math/include/matrix3x3.hh"
#include "utils/math/include/vector3.hh"
#include "utils/memory/include/jeod_alloc.hh"
#include "../include/dyn_body.hh"
#include "../include/dyn_body_messages.hh"
```

### Namespaces

- [jeod](#)

*Namespace jeod.*

### 9.7.1 Detailed Description

Define base methods for the DynBody class.

Definition in file [dyn\\_body.cc](#).

## 9.8 dyn\_body.hh File Reference

Define the class DynBody.

```
#include <vector>
#include <list>
#include "body_ref_frame.hh"
#include "body_force_collect.hh"
#include "frame_derivs.hh"
#include "dyn_body_generic_rigid_attach.hh"
#include "dynamics/mass/include/mass.hh"
#include "environment/gravity/include/gravity_interaction.hh"
#include "utils/container/include/simple_checkpointable.hh"
#include "utils/integration/include/generalized_second_order_ode_technique.-
hh"
#include "utils/integration/include/restartable_state_integrator.hh"
#include "utils/sim_interface/include/jeod_class.hh"
#include "utils/ref_frames/include/ref_frame_interface.hh"
#include "er7_utils/integration/core/include/integrable_object.hh"
#include "er7_utils/integration/core/include/integrator_result.hh"
#include "er7_utils/integration/core/include/integrator_result_merger_-
container.hh"
```



## Data Structures

- class [jeod::DynBody](#)

Class [DynBody](#) is the base class for all dynamic bodies.

## Namespaces

- [jeod](#)

Namespace [jeod](#).

### 9.8.1 Detailed Description

Define the class `DynBody`.

Definition in file [dyn\\_body.hh](#).

## 9.9 dyn\_body\_attach.cc File Reference

Define `DynBody` attachment methods.

```
#include <cstdint>
#include <string>
#include <list>
#include "dynamics/dyn_manager/include/base_dyn_manager.hh"
#include "dynamics/mass/include/mass.hh"
#include "utils/math/include/matrix3x3.hh"
#include "utils/math/include/vector3.hh"
#include "utils/message/include/message_handler.hh"
#include "../include/dyn_body.hh"
#include "../include/dyn_body_messages.hh"
#include "../include/body_ref_frame.hh"
#include "../../dyn_manager/include/dynamics_integration_group.hh"
#include "environment/ephemerides/ephem_interface/include/ephem_ref_frame.-
hh"
#include "../../derived_state/include/relative_derived_state.hh"
```

## Namespaces

- [jeod](#)

Namespace [jeod](#).

### 9.9.1 Detailed Description

Define `DynBody` attachment methods.

Definition in file [dyn\\_body\\_attach.cc](#).

## 9.10 dyn\_body\_collect.cc File Reference

Define `DynBody` methods related to force and torque accumulation and propagation.

```
#include <cstdint>
#include "dynamics/dyn_manager/include/base_dyn_manager.hh"
#include "utils/math/include/matrix3x3.hh"
#include "utils/math/include/vector3.hh"
#include "../include/dyn_body.hh"
```

## Namespaces

- [jeod](#)

*Namespace jeod.*

## Functions

- static void [jeod::accumulate\\_forces](#) (const JeodPointerVector< CollectForce >::type &vec, double \*cumulation)

*Accumulate forces acting on a vehicle.*

- static void [jeod::accumulate\\_torques](#) (const JeodPointerVector< CollectTorque >::type &vec, double \*cumulation)

*Accumulate torques acting on a vehicle.*

### 9.10.1 Detailed Description

Define DynBody methods related to force and torque accumulation and propagation.

Definition in file [dyn\\_body\\_collect.cc](#).

## 9.11 dyn\_body\_detach.cc File Reference

Define DynBody detachment methods.

```
#include <cstdint>
#include <algorithm>
#include "dynamics/dyn_manager/include/dyn_manager.hh"
#include "utils/math/include/matrix3x3.hh"
#include "utils/math/include/vector3.hh"
#include "utils/message/include/message_handler.hh"
#include "utils/ref_frames/include/tree_links_iterator.hh"
#include "../include/dyn_body.hh"
#include "../include/dyn_body_messages.hh"
```

## Namespaces

- [jeod](#)

*Namespace jeod.*

### 9.11.1 Detailed Description

Define DynBody detachment methods.

Definition in file [dyn\\_body\\_detach.cc](#).

## 9.12 dyn\_body\_find\_body\_frame.cc File Reference

Define DynBody::find\_body\_frame.

```
#include <cstdint>
#include "dynamics/dyn_manager/include/base_dyn_manager.hh"
#include "utils/message/include/message_handler.hh"
#include "utils/named_item/include/named_item.hh"
#include "../include/dyn_body.hh"
#include "../include/dyn_body_messages.hh"
```

### Namespaces

- [jeod](#)

*Namespace jeod.*

### 9.12.1 Detailed Description

Define DynBody::find\_body\_frame.

Definition in file [dyn\\_body\\_find\\_body\\_frame.cc](#).

## 9.13 dyn\_body\_generic\_rigid\_attach.hh File Reference

Define the class Wrench.

```
#include "utils/sim_interface/include/jeod_class.hh"
#include "body_ref_frame.hh"
#include "../../mass/include/mass_point_state.hh"
```

### Data Structures

- class [jeod::DynBodyGenericFrameAttachment](#)

*A wrench comprises a torque and a force applied at a point on a [DynBody](#).*

### Namespaces

- [jeod](#)

*Namespace jeod.*

### 9.13.1 Detailed Description

Define the class Wrench.

Definition in file [dyn\\_body\\_generic\\_rigid\\_attach.hh](#).

## 9.14 dyn\_body\_initialize\_model.cc File Reference

Define DynBody::initialize\_model.

```
#include <cstddef>
#include "dynamics/dyn_manager/include/base_dyn_manager.hh"
#include "utils/message/include/message_handler.hh"
#include "../include/dyn_body.hh"
#include "../include/dyn_body_messages.hh"
```

## Namespaces

- [jeod](#)

*Namespace jeod.*

### 9.14.1 Detailed Description

Define DynBody::initialize\_model.

Definition in file [dyn\\_body\\_initialize\\_model.cc](#).

## 9.15 dyn\_body\_integration.cc File Reference

Define methods for frame switching.

```
#include <cstddef>
#include "er7_utils/integration/core/include/second_order_ode_integrator.-
hh"
#include "dynamics/dyn_manager/include/base_dyn_manager.hh"
#include "dynamics/dyn_manager/include/dynamics_integration_group.hh"
#include "environment/ephemerides/ephem_interface/include/ephem_ref_frame.-
hh"
#include "utils/math/include/matrix3x3.hh"
#include "utils/math/include/vector3.hh"
#include "utils/memory/include/jeod_alloc.hh"
#include "utils/message/include/message_handler.hh"
#include "utils/named_item/include/named_item.hh"
#include "utils/integration/include/jeod_integration_time.hh"
#include "utils/integration/include/generalized_second_order_ode_technique.-
hh"
#include "../include/dyn_body.hh"
#include "../include/dyn_body_messages.hh"
```

## Namespaces

- [jeod](#)

*Namespace jeod.*

### 9.15.1 Detailed Description

Define methods for frame switching.

Definition in file [dyn\\_body\\_integration.cc](#).

## 9.16 dyn\_body\_messages.cc File Reference

Implement the class De4xxMessages.

```
#include "../include/dyn_body_messages.hh"
```

### Namespaces

- [jeod](#)

*Namespace jeod.*

### Macros

- #define [PATH](#) "dynamics/dyn\_body/"

#### 9.16.1 Detailed Description

Implement the class De4xxMessages.

Definition in file [dyn\\_body\\_messages.cc](#).

## 9.17 dyn\_body\_messages.hh File Reference

Define the class DynBodyMessages.

```
#include "utils/sim_interface/include/jeod_class.hh"
```

### Data Structures

- class [jeod::DynBodyMessages](#)

*Specify the message IDs used in the [DynBody](#) model.*

### Namespaces

- [jeod](#)

*Namespace jeod.*

#### 9.17.1 Detailed Description

Define the class DynBodyMessages.

Definition in file [dyn\\_body\\_messages.hh](#).

## 9.18 dyn\_body\_propagate\_state.cc File Reference

Define DynBody state propagation / update methods.

```
#include <cstdlib>
#include "utils/integration/include/jeod_integration_time.hh"
#include "utils/math/include/matrix3x3.hh"
#include "utils/math/include/vector3.hh"
#include "utils/message/include/message_handler.hh"
#include "../include/dyn_body.hh"
#include "../include/dyn_body_messages.hh"
```

## Namespaces

- [jeod](#)

*Namespace jeod.*

### 9.18.1 Detailed Description

Define DynBody state propagation / update methods.

Definition in file [dyn\\_body\\_propagate\\_state.cc](#).

## 9.19 dyn\_body\_set\_state.cc File Reference

Define methods related to setting aspects of a vehicle's state.

```
#include <cstdlib>
#include "utils/ref_frames/include/ref_frame_items.hh"
#include "utils/math/include/vector3.hh"
#include "utils/message/include/message_handler.hh"
#include "../include/dyn_body.hh"
#include "../include/dyn_body_messages.hh"
```

## Namespaces

- [jeod](#)

*Namespace jeod.*

## Functions

- static void [jeod::check\\_frame\\_ownership](#) (const BodyRefFrame &frame, const DynBody \*dyn\_body, const char \*file, unsigned int line)

*Check that the dyn\_body 'owns' the subject frame.*

### 9.19.1 Detailed Description

Define methods related to setting aspects of a vehicle's state.

Definition in file [dyn\\_body\\_set\\_state.cc](#).

## 9.20 dyn\_body\_vehicle\_point.cc File Reference

Define methods that support vehicle points.

```
#include <cstdint>
#include "dynamics/dyn_manager/include/base_dyn_manager.hh"
#include "environment/ephemerides/ephem_interface/include/ephem_ref_frame.-
hh"
#include "utils/math/include/matrix3x3.hh"
#include "utils/math/include/vector3.hh"
#include "utils/memory/include/jeod_alloc.hh"
#include "utils/message/include/message_handler.hh"
#include "utils/named_item/include/named_item.hh"
#include "utils/quaternion/include/quat.hh"
#include "../include/dyn_body.hh"
#include "../include/dyn_body_messages.hh"
```

### Namespaces

- [jeod](#)

*Namespace jeod.*

#### 9.20.1 Detailed Description

Define methods that support vehicle points.

Definition in file [dyn\\_body\\_vehicle\\_point.cc](#).

## 9.21 force.cc File Reference

Define force model member functions.

```
#include <cstdint>
#include "utils/math/include/vector3.hh"
#include "utils/memory/include/jeod_alloc.hh"
#include "../include/force.hh"
```

### Namespaces

- [jeod](#)

*Namespace jeod.*

#### 9.21.1 Detailed Description

Define force model member functions.

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

## 9.22 force.hh File Reference

Define the JEOD force model.

```
#include "force_inline.hh"
```

## Data Structures

- class [jeod::Force](#)  
*A [Force](#) represents a Newtonian force that acts on a [DynBody](#).*
- class [jeod::CollectForce](#)  
*A [CollectForce](#) represents a collected force that acts on a vehicle.*
- class [jeod::CInterfaceForce](#)  
*This class is deprecated.*

## Namespaces

- [jeod](#)  
*Namespace [jeod](#).*

### 9.22.1 Detailed Description

Define the JEOD force model.

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

## 9.23 [force\\_inline.hh](#) File Reference

Inline functions for the JEOD force model.

```
#include "force.hh"  
#include <cstdlib>
```

## Namespaces

- [jeod](#)  
*Namespace [jeod](#).*

### 9.23.1 Detailed Description

Inline functions for the JEOD force model.

Definition in file [force\\_inline.hh](#).

## 9.24 [frame\\_derivs.hh](#) File Reference

Define the FrameDerivs class.

```
#include "utils/quaternion/include/quat.hh"
```



## Data Structures

- class [jeod::FrameDerivs](#)  
*Contains translational and rotational second derivatives.*

## Namespaces

- [jeod](#)  
*Namespace jeod.*

### 9.24.1 Detailed Description

Define the FrameDerivs class.

Definition in file [frame\\_derivs.hh](#).

## 9.25 structure\_integrated\_dyn\_body.cc File Reference

Define base member functions for StructureIntegratedDynBody.

```
#include "../include/structure_integrated_dyn_body.hh"
#include "utils/memory/include/jeod_alloc.hh"
#include <cstdlib>
```

## Namespaces

- [jeod](#)  
*Namespace jeod.*

### 9.25.1 Detailed Description

Define base member functions for StructureIntegratedDynBody.

Definition in file [structure\\_integrated\\_dyn\\_body.cc](#).

## 9.26 structure\_integrated\_dyn\_body.hh File Reference

Define the class StructureIntegratedDynBody, which integrates a DynBody object's structural state.

```
#include "body_wrench_collect.hh"
#include "vehicle_properties.hh"
#include "vehicle_non_grav_state.hh"
#include "dynamics/dyn_body/include/dyn_body.hh"
#include "utils/sim_interface/include/jeod_class.hh"
```

## Data Structures

- class [jeod::StructureIntegratedDynBody](#)  
*Extends [DynBody](#) to integrate an object's structural reference frame as opposed to its center of mass.*

## Namespaces

- [jeod](#)

*Namespace jeod.*

### 9.26.1 Detailed Description

Define the class `StructureIntegratedDynBody`, which integrates a `DynBody` object's structural state.

Definition in file [structure\\_integrated\\_dyn\\_body.hh](#).

## 9.27 structure\_integrated\_dyn\_body\_collect.cc File Reference

Define `StructureIntegratedDynBody` methods related to force and torque accumulation and propagation.

```
#include "../include/structure_integrated_dyn_body.hh"
#include "dynamics/dyn_manager/include/base_dyn_manager.hh"
#include "utils/math/include/matrix3x3.hh"
#include "utils/math/include/vector3.hh"
#include <cstdint>
```

## Namespaces

- [jeod](#)

*Namespace jeod.*

## Functions

- static void [jeod::accumulate\\_forces](#) (const JeodPointerVector< CollectForce >::type &vec, double \*cumulation)  
*Accumulate forces acting on a vehicle.*
- static void [jeod::accumulate\\_torques](#) (const JeodPointerVector< CollectTorque >::type &vec, double \*cumulation)  
*Accumulate torques acting on a vehicle.*

### 9.27.1 Detailed Description

Define `StructureIntegratedDynBody` methods related to force and torque accumulation and propagation.

Definition in file [structure\\_integrated\\_dyn\\_body\\_collect.cc](#).

## 9.28 structure\_integrated\_dyn\_body\_integration.cc File Reference

Define `StructureIntegratedDynBody` member functions related to state integration.

```
#include "../include/structure_integrated_dyn_body.hh"
#include "dynamics/dyn_body/include/dyn_body_messages.hh"
#include "utils/math/include/vector3.hh"
#include "utils/memory/include/jeod_alloc.hh"
#include "utils/message/include/message_handler.hh"
#include "utils/ref_frames/include/ref_frame_items.hh"
#include "er7_utils/integration/core/include/second_order_ode_integrator.-
```

```
hh"
#include <cstdlib>
#include <cmath>
```

## Namespaces

- [jeod](#)

*Namespace jeod.*

### 9.28.1 Detailed Description

Define StructureIntegratedDynBody member functions related to state integration.

Definition in file [structure\\_integrated\\_dyn\\_body\\_integration.cc](#).

## 9.29 structure\_integrated\_dyn\_body\_pt\_accel.cc File Reference

Define StructureIntegratedDynBody::compute\_vehicle\_point\_derivatives.

```
#include "../include/structure_integrated_dyn_body.hh"
#include "dynamics/dyn_body/include/dyn_body_messages.hh"
#include "utils/math/include/vector3.hh"
#include "utils/message/include/message_handler.hh"
#include <cstring>
#include <cstdio>
```

## Namespaces

- [jeod](#)

*Namespace jeod.*

### 9.29.1 Detailed Description

Define StructureIntegratedDynBody::compute\_vehicle\_point\_derivatives.

Definition in file [structure\\_integrated\\_dyn\\_body\\_pt\\_accel.cc](#).

## 9.30 structure\_integrated\_dyn\_body\_solve.cc File Reference

Define StructureIntegratedDynBody methods related to force and torque accumulation and propagation.

```
#include "../include/structure_integrated_dyn_body.hh"
#include "../include/dyn_body_messages.hh"
#include "utils/message/include/message_handler.hh"
#include "utils/math/include/vector3.hh"
#include "experimental/constraints/include/dyn_body_constraints_solver.hh"
```

## Namespaces

- [jeod](#)  
*Namespace jeod.*

### 9.30.1 Detailed Description

Define StructureIntegratedDynBody methods related to force and torque accumulation and propagation.

Definition in file [structure\\_integrated\\_dyn\\_body\\_solve.cc](#).

## 9.31 torque.cc File Reference

Define torque model member functions.

```
#include <cstdlib>
#include "utils/math/include/vector3.hh"
#include "utils/memory/include/jeod_alloc.hh"
#include "../include/torque.hh"
```

## Namespaces

- [jeod](#)  
*Namespace jeod.*

### 9.31.1 Detailed Description

Define torque model member functions.

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

## 9.32 torque.hh File Reference

Define the JEOD torque model.

```
#include "torque_inline.hh"
```

## Data Structures

- class [jeod::Torque](#)  
*A [Torque](#) represents a Newtonian torque that acts on a [DynBody](#).*
- class [jeod::CollectTorque](#)  
*A [CollectTorque](#) represents a collected torque that acts on a vehicle.*
- class [jeod::CInterfaceTorque](#)  
*This class is deprecated.*

## Namespaces

- [jeod](#)  
*Namespace jeod.*

### 9.32.1 Detailed Description

Define the JEOD torque model.

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

## 9.33 torque\_inline.hh File Reference

Define the JEOD torque model.

```
#include "torque.hh"
#include <cstdint>
```

### Namespaces

- [jeod](#)

*Namespace jeod.*

### 9.33.1 Detailed Description

Define the JEOD torque model.

Definition in file [torque\\_inline.hh](#).

## 9.34 vehicle\_non\_grav\_state.hh File Reference

Define the class VehicleNonGravState.

```
#include "utils/sim_interface/include/jeod_class.hh"
```

### Data Structures

- class [jeod::VehicleNonGravState](#)

*Encapsulates various aspects of a vehicle's state with respect to inertial.*

### Namespaces

- [jeod](#)

*Namespace jeod.*

### 9.34.1 Detailed Description

Define the class VehicleNonGravState.

Definition in file [vehicle\\_non\\_grav\\_state.hh](#).

## 9.35 vehicle\_properties.hh File Reference

Define the class VehicleProperties.

```
#include "experimental/math/include/solver_types.hh"
#include "utils/sim_interface/include/jeod_class.hh"
```

### Data Structures

- class [jeod::VehicleProperties](#)

*Captures pointers to various vehicle properties that are commonly used in the constraint concept.*

### Namespaces

- [jeod](#)

*Namespace jeod.*

#### 9.35.1 Detailed Description

Define the class VehicleProperties.

Definition in file [vehicle\\_properties.hh](#).

## 9.36 wrench.hh File Reference

Define the class Wrench.

```
#include "dynamics/mass/include/mass_point_state.hh"
#include "utils/math/include/vector3.hh"
#include "utils/sim_interface/include/jeod_class.hh"
#include <vector>
```

### Data Structures

- class [jeod::Wrench](#)

*A wrench comprises a torque and a force applied at a point on a [DynBody](#).*

### Namespaces

- [jeod](#)

*Namespace jeod.*

#### 9.36.1 Detailed Description

Define the class Wrench.

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

# Index

- ~BodyForceCollect
  - jeod::BodyForceCollect, [22](#)
- ~BodyRefFrame
  - jeod::BodyRefFrame, [27](#)
- ~BodyWrenchCollect
  - jeod::BodyWrenchCollect, [29](#)
- ~CInterfaceForce
  - jeod::CInterfaceForce, [31](#)
- ~CInterfaceTorque
  - jeod::CInterfaceTorque, [32](#)
- ~CollectForce
  - jeod::CollectForce, [35](#)
- ~CollectTorque
  - jeod::CollectTorque, [40](#)
- ~DynBody
  - jeod::DynBody, [49](#)
- ~Force
  - jeod::Force, [88](#)
- ~StructureIntegratedDynBody
  - jeod::StructureIntegratedDynBody, [96](#)
- ~Torque
  - jeod::Torque, [104](#)
- ~Wrench
  - jeod::Wrench, [115](#)
- accel\_struct
  - jeod::VehicleNonGravState, [107](#)
- accumulate
  - jeod::BodyWrenchCollect, [29](#)
  - jeod::Wrench, [115](#), [116](#)
- accumulate\_forces
  - jeod, [16](#)
- accumulate\_torques
  - jeod, [18](#)
- activate
  - jeod::DynBody, [50](#)
  - jeod::Wrench, [116](#)
- active
  - jeod::CollectForce, [38](#)
  - jeod::CollectTorque, [43](#)
  - jeod::DynBodyGenericFrameAttachment, [83](#)
  - jeod::Force, [89](#)
  - jeod::Torque, [105](#)
  - jeod::Wrench, [120](#)
- add\_constraint
  - jeod::StructureIntegratedDynBody, [96](#)
- add\_control
  - jeod::DynBody, [50](#)
- add\_integrable\_object
  - jeod::DynBody, [50](#)
- add\_mass\_body
  - jeod::DynBody, [50](#)
- add\_mass\_body\_frames
  - jeod::DynBody, [50](#)
- add\_mass\_body\_validate
  - jeod::DynBody, [51](#)
- add\_mass\_point
  - jeod::DynBody, [51](#)
- associated\_integrable\_objects
  - jeod::DynBody, [74](#)
- attach\_child
  - jeod::DynBody, [51](#), [52](#)
- attach\_establish\_links
  - jeod::DynBody, [52](#)
- attach\_to
  - jeod::DynBody, [52](#), [53](#)
- attach\_to\_frame
  - jeod::DynBody, [53](#)
- attach\_update\_properties
  - jeod::DynBody, [54](#)
  - jeod::StructureIntegratedDynBody, [96](#)
- attach\_validate\_child
  - jeod::DynBody, [54](#)
- attach\_validate\_parent
  - jeod::DynBody, [55](#)
- attitude\_source
  - jeod::DynBody, [74](#)
- autoupdate\_vehicle\_points
  - jeod::DynBody, [74](#)
- aux\_classes.cc, [123](#)
- body\_force\_collect.hh, [123](#)
- body\_ref\_frame.hh, [124](#)
- body\_wrench\_collect.cc, [125](#)
- body\_wrench\_collect.hh, [125](#)
- BodyForceCollect
  - jeod::BodyForceCollect, [22](#)
- BodyRefFrame
  - jeod::BodyRefFrame, [27](#)
- BodyWrenchCollect
  - jeod::BodyWrenchCollect, [28](#), [29](#)
- CInterfaceForce
  - jeod::CInterfaceForce, [31](#)
- CInterfaceTorque
  - jeod::CInterfaceTorque, [32](#), [33](#)
- check\_frame\_ownership
  - jeod, [18](#)
- class\_declarations.hh, [125](#)
- clear\_attachment

- jeod::DynBodyGenericFrameAttachment, 82
- clear\_integrable\_objects
  - jeod::DynBody, 55
- collect
  - jeod::DynBody, 75
- collect\_effector\_forc
  - jeod::BodyForceCollect, 23
- collect\_effector\_torq
  - jeod::BodyForceCollect, 23
- collect\_environ\_forc
  - jeod::BodyForceCollect, 23
- collect\_environ\_torq
  - jeod::BodyForceCollect, 23
- collect\_forces\_and\_torques
  - jeod::DynBody, 55
  - jeod::StructureIntegratedDynBody, 97
- collect\_insert
  - jeod, 18
  - jeod::JPVCollectForce, 92
  - jeod::JPVCollectTorque, 93
- collect\_local\_forces\_and\_torques
  - jeod::StructureIntegratedDynBody, 97
- collect\_no\_xmit\_forc
  - jeod::BodyForceCollect, 23
- collect\_no\_xmit\_torq
  - jeod::BodyForceCollect, 23
- collect\_push\_back
  - jeod, 18
  - jeod::JPVCollectForce, 92
  - jeod::JPVCollectTorque, 93
- collect\_wrench
  - jeod::BodyWrenchCollect, 29
- CollectForce
  - jeod::CollectForce, 34, 35
- CollectTorque
  - jeod::CollectTorque, 40
- complete\_translational\_acceleration
  - jeod::StructureIntegratedDynBody, 97
- composite\_body
  - jeod::DynBody, 75
- compute\_derived\_state\_forward
  - jeod::DynBody, 56
- compute\_derived\_state\_reverse
  - jeod::DynBody, 56
- compute\_inertial\_torque
  - jeod::StructureIntegratedDynBody, 98
- compute\_point\_derivative
  - jeod::DynBody, 75
- compute\_ref\_point\_transform
  - jeod::DynBody, 56
- compute\_rotational\_acceleration
  - jeod::StructureIntegratedDynBody, 98
- compute\_state\_elements\_forward
  - jeod::DynBody, 57
- compute\_state\_elements\_reverse
  - jeod::DynBody, 57
- compute\_translational\_acceleration
  - jeod::StructureIntegratedDynBody, 98
- compute\_vehicle\_point\_derivatives
  - jeod::DynBody, 57
  - jeod::StructureIntegratedDynBody, 98
- compute\_vehicle\_point\_states
  - jeod::DynBody, 58
- constraints\_solver
  - jeod::StructureIntegratedDynBody, 102
- core\_body
  - jeod::DynBody, 75
- create
  - jeod::CollectForce, 35, 36
  - jeod::CollectTorque, 41, 42
- create\_body\_integrators
  - jeod::DynBody, 58
- create\_integrators
  - jeod::DynBody, 58
- deactivate
  - jeod::DynBody, 60
  - jeod::Wrench, 116
- derivs
  - jeod::DynBody, 75
- destroy\_integrators
  - jeod::DynBody, 60
- detach
  - jeod::DynBody, 60
  - jeod::StructureIntegratedDynBody, 98
- detach\_mass\_body\_frames
  - jeod::DynBody, 61
- detach\_mass\_internal
  - jeod::DynBody, 61
- dyn\_body.cc, 126
- dyn\_body.hh, 126
- dyn\_body\_attach.cc, 127
- dyn\_body\_collect.cc, 127
- dyn\_body\_detach.cc, 128
- dyn\_body\_find\_body\_frame.cc, 129
- dyn\_body\_generic\_rigid\_attach.hh, 129
- dyn\_body\_initialize\_model.cc, 129
- dyn\_body\_integration.cc, 130
- dyn\_body\_messages.cc, 131
- dyn\_body\_messages.hh, 131
- dyn\_body\_propagate\_state.cc, 131
- dyn\_body\_set\_state.cc, 132
- dyn\_body\_vehicle\_point.cc, 133
- dyn\_children
  - jeod::DynBody, 76
- dyn\_manager
  - jeod::DynBody, 76
- dyn\_parent
  - jeod::DynBody, 76
- DynBody, 13
  - jeod::DynBody, 49, 50
  - PATH, 14
- DynBodyConstraintsSolver
  - jeod::StructureIntegratedDynBody, 101
- DynBodyGenericFrameAttachment
  - jeod::DynBodyGenericFrameAttachment, 82
- DynBodyMessages



- jeod::DynBodyMessages, 84
- Dynamics, 12
- effector\_forc
  - jeod::BodyForceCollect, 24
- effector\_torq
  - jeod::BodyForceCollect, 24
- effector\_wrench
  - jeod::StructureIntegratedDynBody, 102
- effector\_wrench\_collection
  - jeod::StructureIntegratedDynBody, 102
- environ\_forc
  - jeod::BodyForceCollect, 24
- environ\_torq
  - jeod::BodyForceCollect, 24
- extern\_forc\_inrtl
  - jeod::BodyForceCollect, 24
- extern\_forc\_struct
  - jeod::BodyForceCollect, 25
- extern\_torq\_body
  - jeod::BodyForceCollect, 25
- extern\_torq\_struct
  - jeod::BodyForceCollect, 25
- find\_body\_frame
  - jeod::DynBody, 61
- find\_vehicle\_point
  - jeod::DynBody, 62
- Force
  - jeod::Force, 88
- force
  - jeod::CollectForce, 38
  - jeod::Force, 89
  - jeod::Wrench, 120
- force.cc, 133
- force.hh, 133
- force\_inline.hh, 134
- frame\_attach
  - jeod::DynBody, 76
- frame\_derivs.hh, 134
- FrameDerivs
  - jeod::FrameDerivs, 90
- get\_attach\_offset
  - jeod::DynBodyGenericFrameAttachment, 82
- get\_dynamics\_integration\_group
  - jeod::DynBody, 62
- get\_force
  - jeod::Wrench, 116
- get\_inertia
  - jeod::VehicleProperties, 109
- get\_initialized\_states
  - jeod::DynBody, 62
- get\_integrable\_objects
  - jeod::DynBody, 63
- get\_inverse\_inertia
  - jeod::VehicleProperties, 109
- get\_inverse\_mass
  - jeod::VehicleProperties, 110
- get\_mass
  - jeod::VehicleProperties, 110
- get\_parent\_body
  - jeod::DynBody, 63
- get\_parent\_body\_internal
  - jeod::DynBody, 63
- get\_parent\_frame
  - jeod::DynBodyGenericFrameAttachment, 82
- get\_parent\_to\_structure\_offset
  - jeod::VehicleProperties, 110
- get\_parent\_to\_structure\_transform
  - jeod::VehicleProperties, 110
- get\_point
  - jeod::Wrench, 116
- get\_root\_body
  - jeod::DynBody, 63
- get\_root\_body\_internal
  - jeod::DynBody, 63
- get\_structure\_to\_body\_offset
  - jeod::VehicleProperties, 110
- get\_structure\_to\_body\_transform
  - jeod::VehicleProperties, 111
- get\_torque
  - jeod::Wrench, 116
- get\_vehicle\_properties
  - jeod::StructureIntegratedDynBody, 100
- grav\_interaction
  - jeod::DynBody, 76
- inertia
  - jeod::VehicleProperties, 111
- inertial\_accel\_inrtl
  - jeod::StructureIntegratedDynBody, 102
- inertial\_accel\_struct
  - jeod::StructureIntegratedDynBody, 102
- inertial\_accel\_struct\_omega
  - jeod::StructureIntegratedDynBody, 102
- inertial\_accel\_struct\_omega\_dot
  - jeod::StructureIntegratedDynBody, 103
- inertial\_torq
  - jeod::BodyForceCollect, 25
- inertial\_torque\_struct
  - jeod::VehicleNonGravState, 107
- init\_attrjeod\_\_BodyRefFrame
  - jeod::BodyRefFrame, 27
- init\_attrjeod\_\_DynBody
  - jeod::DynBody, 74
- init\_attrjeod\_\_DynBodyGenericFrameAttachment
  - jeod::DynBodyGenericFrameAttachment, 83
- init\_attrjeod\_\_DynBodyMessages
  - jeod::DynBodyMessages, 84
- init\_attrjeod\_\_StructureIntegratedDynBody
  - jeod::StructureIntegratedDynBody, 101
- init\_attrjeod\_\_VehicleNonGravState
  - jeod::VehicleNonGravState, 106
- init\_attrjeod\_\_VehicleProperties
  - jeod::VehicleProperties, 111
- init\_attrjeod\_\_Wrench
  - jeod::Wrench, 120

initialize\_attachment  
     jeod::DynBodyGenericFrameAttachment, 82  
 initialize\_controls  
     jeod::DynBody, 64  
 initialize\_model  
     jeod::DynBody, 64  
 initialized\_items  
     jeod::BodyRefFrame, 27  
 initialized\_states  
     jeod::DynBody, 77  
 initialized\_states\_contains  
     jeod::DynBody, 64  
 InputProcessor  
     jeod::BodyRefFrame, 27  
     jeod::DynBody, 74  
     jeod::DynBodyGenericFrameAttachment, 83  
     jeod::DynBodyMessages, 84  
     jeod::StructureIntegratedDynBody, 101  
     jeod::VehicleNonGravState, 106  
     jeod::VehicleProperties, 111  
     jeod::Wrench, 120  
 integ\_frame  
     jeod::DynBody, 77  
 integ\_frame\_name  
     jeod::DynBody, 77  
 integ\_results\_merger  
     jeod::DynBody, 77  
 integrate  
     jeod::DynBody, 65  
 integrated\_frame  
     jeod::DynBody, 77  
 internal\_error  
     jeod::DynBodyMessages, 85  
 invalid\_attachment  
     jeod::DynBodyMessages, 85  
 invalid\_body  
     jeod::DynBodyMessages, 85  
 invalid\_frame  
     jeod::DynBodyMessages, 85  
 invalid\_group  
     jeod::DynBodyMessages, 86  
 invalid\_name  
     jeod::DynBodyMessages, 86  
 invalid\_technique  
     jeod::DynBodyMessages, 86  
 inverse\_inertia  
     jeod::VehicleProperties, 111  
 inverse\_mass  
     jeod::VehicleProperties, 111  
 is\_active  
     jeod::CollectForce, 37  
     jeod::CollectTorque, 42  
     jeod::Wrench, 117  
 is\_root\_body  
     jeod::DynBody, 65  
 isAttached  
     jeod::DynBodyGenericFrameAttachment, 82  
 jeod, 15  
     accumulate\_forces, 16  
     accumulate\_torques, 18  
     check\_frame\_ownership, 18  
     collect\_insert, 18  
     collect\_push\_back, 18  
     release\_vector, 19  
 jeod::BodyForceCollect, 21  
     ~BodyForceCollect, 22  
     BodyForceCollect, 22  
     collect\_effector\_forc, 23  
     collect\_effector\_torq, 23  
     collect\_environ\_forc, 23  
     collect\_environ\_torq, 23  
     collect\_no\_xmit\_forc, 23  
     collect\_no\_xmit\_torq, 23  
     effector\_forc, 24  
     effector\_torq, 24  
     environ\_forc, 24  
     environ\_torq, 24  
     extern\_forc\_inrtl, 24  
     extern\_forc\_struct, 25  
     extern\_torq\_body, 25  
     extern\_torq\_struct, 25  
     inertial\_torq, 25  
     no\_xmit\_forc, 25  
     no\_xmit\_torq, 25  
     operator=, 23  
 jeod::BodyRefFrame, 26  
     ~BodyRefFrame, 27  
     BodyRefFrame, 27  
     init\_attrjeod\_\_BodyRefFrame, 27  
     initialized\_items, 27  
     InputProcessor, 27  
     mass\_point, 27  
     operator=, 27  
 jeod::BodyWrenchCollect, 28  
     ~BodyWrenchCollect, 29  
     accumulate, 29  
     BodyWrenchCollect, 28, 29  
     collect\_wrench, 29  
     operator=, 29  
 jeod::CInterfaceForce, 30  
     ~CInterfaceForce, 31  
     CInterfaceForce, 31  
     operator=, 31  
 jeod::CInterfaceTorque, 31  
     ~CInterfaceTorque, 32  
     CInterfaceTorque, 32, 33  
     operator=, 33  
 jeod::CollectForce, 33  
     ~CollectForce, 35  
     active, 38  
     CollectForce, 34, 35  
     create, 35, 36  
     force, 38  
     is\_active, 37  
     operator=, 37  
     operator==, 37

- jeod::CollectTorque, 38
  - ~CollectTorque, 40
  - active, 43
  - CollectTorque, 40
  - create, 41, 42
  - is\_active, 42
  - operator=, 42
  - operator==, 42
  - torque, 43
- jeod::DynBody, 43
  - ~DynBody, 49
  - activate, 50
  - add\_control, 50
  - add\_integrable\_object, 50
  - add\_mass\_body, 50
  - add\_mass\_body\_frames, 50
  - add\_mass\_body\_validate, 51
  - add\_mass\_point, 51
  - associated\_integrable\_objects, 74
  - attach\_child, 51, 52
  - attach\_establish\_links, 52
  - attach\_to, 52, 53
  - attach\_to\_frame, 53
  - attach\_update\_properties, 54
  - attach\_validate\_child, 54
  - attach\_validate\_parent, 55
  - attitude\_source, 74
  - autoupdate\_vehicle\_points, 74
  - clear\_integrable\_objects, 55
  - collect, 75
  - collect\_forces\_and\_torques, 55
  - composite\_body, 75
  - compute\_derived\_state\_forward, 56
  - compute\_derived\_state\_reverse, 56
  - compute\_point\_derivative, 75
  - compute\_ref\_point\_transform, 56
  - compute\_state\_elements\_forward, 57
  - compute\_state\_elements\_reverse, 57
  - compute\_vehicle\_point\_derivatives, 57
  - compute\_vehicle\_point\_states, 58
  - core\_body, 75
  - create\_body\_integrators, 58
  - create\_integrators, 58
  - deactivate, 60
  - derivs, 75
  - destroy\_integrators, 60
  - detach, 60
  - detach\_mass\_body\_frames, 61
  - detach\_mass\_internal, 61
  - dyn\_children, 76
  - dyn\_manager, 76
  - dyn\_parent, 76
  - DynBody, 49, 50
  - find\_body\_frame, 61
  - find\_vehicle\_point, 62
  - frame\_attach, 76
  - get\_dynamics\_integration\_group, 62
  - get\_initialized\_states, 62
  - get\_integrable\_objects, 63
  - get\_parent\_body, 63
  - get\_parent\_body\_internal, 63
  - get\_root\_body, 63
  - get\_root\_body\_internal, 63
  - grav\_interaction, 76
  - init\_attrjeod\_\_DynBody, 74
  - initialize\_controls, 64
  - initialize\_model, 64
  - initialized\_states, 77
  - initialized\_states\_contains, 64
  - InputProcessor, 74
  - integ\_frame, 77
  - integ\_frame\_name, 77
  - integ\_results\_merger, 77
  - integrate, 65
  - integrated\_frame, 77
  - is\_root\_body, 65
  - mass, 78
  - mass\_children, 78
  - migrate\_integrable\_objects, 65
  - name, 78
  - operator=, 65
  - position\_source, 78
  - process\_dynamic\_attachment, 65
  - propagate\_state, 66
  - propagate\_state\_from\_composite, 66
  - propagate\_state\_from\_structure, 67
  - rate\_source, 78
  - remove\_integrable\_object, 67
  - remove\_mass\_body, 67
  - reset\_controls, 68
  - reset\_integrators, 68
  - rot\_integ, 68
  - rot\_integrator, 79
  - rotation\_integration, 79
  - rotational\_dynamics, 79
  - set\_attitude\_left\_quaternion, 68
  - set\_attitude\_matrix, 69
  - set\_attitude\_rate, 69
  - set\_attitude\_right\_quaternion, 69
  - set\_integ\_frame, 70
  - set\_name, 70
  - set\_position, 71
  - set\_state, 71
  - set\_state\_source, 71
  - set\_state\_source\_internal, 72
  - set\_velocity, 72
  - sort\_controls, 72
  - structure, 79
  - switch\_integration\_frames, 72, 73
  - three\_dof, 79
  - time\_manager, 80
  - trans\_integ, 73
  - trans\_integrator, 80
  - translational\_dynamics, 80
  - update\_integrated\_state, 74
  - vehicle\_points, 80

- velocity\_source, 80
- jeod::DynBodyGenericFrameAttachment, 81
  - active, 83
  - clear\_attachment, 82
  - DynBodyGenericFrameAttachment, 82
  - get\_attach\_offset, 82
  - get\_parent\_frame, 82
  - init\_attrjeod\_\_DynBodyGenericFrameAttachment, 83
  - initialize\_attachment, 82
  - InputProcessor, 83
  - isAttached, 82
  - rigid\_attach\_parent, 83
  - rigid\_attach\_state, 83
- jeod::DynBodyMessages, 83
  - DynBodyMessages, 84
  - init\_attrjeod\_\_DynBodyMessages, 84
  - InputProcessor, 84
  - internal\_error, 85
  - invalid\_attachment, 85
  - invalid\_body, 85
  - invalid\_frame, 85
  - invalid\_group, 86
  - invalid\_name, 86
  - invalid\_technique, 86
  - not\_dyn\_body, 86
  - operator=, 84
- jeod::Force, 87
  - ~Force, 88
  - active, 89
  - Force, 88
  - force, 89
  - operator=, 88
- jeod::FrameDerivs, 89
  - FrameDerivs, 90
  - non\_grav\_accel, 90
  - Qdot\_parent\_this, 90
  - rot\_accel, 90
  - trans\_accel, 90
- jeod::JPVCollectForce, 91
  - collect\_insert, 92
  - collect\_push\_back, 92
  - perform\_insert\_action, 91
  - push\_back, 92
- jeod::JPVCollectTorque, 92
  - collect\_insert, 93
  - collect\_push\_back, 93
  - perform\_insert\_action, 93
  - push\_back, 93
- jeod::StructureIntegratedDynBody, 94
  - ~StructureIntegratedDynBody, 96
  - add\_constraint, 96
  - attach\_update\_properties, 96
  - collect\_forces\_and\_torques, 97
  - collect\_local\_forces\_and\_torques, 97
  - complete\_translational\_acceleration, 97
  - compute\_inertial\_torque, 98
  - compute\_rotational\_acceleration, 98
  - compute\_translational\_acceleration, 98
  - compute\_vehicle\_point\_derivatives, 98
  - constraints\_solver, 102
  - detach, 98
  - DynBodyConstraintsSolver, 101
  - effector\_wrench, 102
  - effector\_wrench\_collection, 102
  - get\_vehicle\_properties, 100
  - inertial\_accel\_inrtl, 102
  - inertial\_accel\_struct, 102
  - inertial\_accel\_struct\_omega, 102
  - inertial\_accel\_struct\_omega\_dot, 103
  - init\_attrjeod\_\_StructureIntegratedDynBody, 101
  - InputProcessor, 101
  - non\_grav\_state, 103
  - operator=, 100
  - PropagateForcesAndTorques, 100
  - rot\_integ, 100
  - set\_solver, 100
  - solve\_constraints, 101
  - struct\_derivs, 103
  - StructureIntegratedDynBody, 96
  - trans\_integ, 101
  - vehicle\_properties, 103
- jeod::Torque, 104
  - ~Torque, 104
  - active, 105
  - operator=, 105
  - Torque, 104, 105
  - torque, 105
- jeod::VehicleNonGravState, 106
  - accel\_struct, 107
  - inertial\_torque\_struct, 107
  - init\_attrjeod\_\_VehicleNonGravState, 106
  - InputProcessor, 106
  - omega\_body, 107
  - omega\_dot\_body, 107
  - omega\_dot\_struct, 107
  - omega\_struct, 107
- jeod::VehicleProperties, 108
  - get\_inertia, 109
  - get\_inverse\_inertia, 109
  - get\_inverse\_mass, 110
  - get\_mass, 110
  - get\_parent\_to\_structure\_offset, 110
  - get\_parent\_to\_structure\_transform, 110
  - get\_structure\_to\_body\_offset, 110
  - get\_structure\_to\_body\_transform, 111
  - inertia, 111
  - init\_attrjeod\_\_VehicleProperties, 111
  - InputProcessor, 111
  - inverse\_inertia, 111
  - inverse\_mass, 111
  - mass, 111
  - parent\_to\_structure\_offset, 112
  - parent\_to\_structure\_transform, 112
  - structure\_to\_body\_offset, 112
  - structure\_to\_body\_transform, 112

- VehicleProperties, 109
- jeod::Wrench, 112
  - ~Wrench, 115
  - accumulate, 115, 116
  - activate, 116
  - active, 120
  - deactivate, 116
  - force, 120
  - get\_force, 116
  - get\_point, 116
  - get\_torque, 116
  - init\_attrjeod\_\_Wrench, 120
  - InputProcessor, 120
  - is\_active, 117
  - operator+=, 117
  - operator=, 117
  - point, 120
  - reset\_force, 117
  - reset\_force\_and\_torque, 117
  - reset\_point, 117
  - reset\_torque, 118
  - scale\_force, 118
  - scale\_torque, 118
  - set, 118
  - set\_force, 118
  - set\_point, 119
  - set\_torque, 119
  - torque, 120
  - transform\_to\_parent, 119
  - transform\_to\_point, 119
  - Wrench, 114, 115
- mass
  - jeod::DynBody, 78
  - jeod::VehicleProperties, 111
- mass\_children
  - jeod::DynBody, 78
- mass\_point
  - jeod::BodyRefFrame, 27
- migrate\_integrable\_objects
  - jeod::DynBody, 65
- Models, 11
- name
  - jeod::DynBody, 78
- no\_xmit\_forc
  - jeod::BodyForceCollect, 25
- no\_xmit\_torq
  - jeod::BodyForceCollect, 25
- non\_grav\_accel
  - jeod::FrameDerivs, 90
- non\_grav\_state
  - jeod::StructureIntegratedDynBody, 103
- not\_dyn\_body
  - jeod::DynBodyMessages, 86
- omega\_body
  - jeod::VehicleNonGravState, 107
- omega\_dot\_body
  - jeod::VehicleNonGravState, 107
- omega\_dot\_struct
  - jeod::VehicleNonGravState, 107
- omega\_struct
  - jeod::VehicleNonGravState, 107
- operator+=
  - jeod::Wrench, 117
- operator=
  - jeod::BodyForceCollect, 23
  - jeod::BodyRefFrame, 27
  - jeod::BodyWrenchCollect, 29
  - jeod::CInterfaceForce, 31
  - jeod::CInterfaceTorque, 33
  - jeod::CollectForce, 37
  - jeod::CollectTorque, 42
  - jeod::DynBody, 65
  - jeod::DynBodyMessages, 84
  - jeod::Force, 88
  - jeod::StructureIntegratedDynBody, 100
  - jeod::Torque, 105
  - jeod::Wrench, 117
- operator==
  - jeod::CollectForce, 37
  - jeod::CollectTorque, 42
- PATH
  - DynBody, 14
- parent\_to\_structure\_offset
  - jeod::VehicleProperties, 112
- parent\_to\_structure\_transform
  - jeod::VehicleProperties, 112
- perform\_insert\_action
  - jeod::JPVCollectForce, 91
  - jeod::JPVCollectTorque, 93
- point
  - jeod::Wrench, 120
- position\_source
  - jeod::DynBody, 78
- process\_dynamic\_attachment
  - jeod::DynBody, 65
- propagate\_state
  - jeod::DynBody, 66
- propagate\_state\_from\_composite
  - jeod::DynBody, 66
- propagate\_state\_from\_structure
  - jeod::DynBody, 67
- PropagateForcesAndTorques
  - jeod::StructureIntegratedDynBody, 100
- push\_back
  - jeod::JPVCollectForce, 92
  - jeod::JPVCollectTorque, 93
- Qdot\_parent\_this
  - jeod::FrameDerivs, 90
- rate\_source
  - jeod::DynBody, 78
- release\_vector
  - jeod, 19

- remove\_integrable\_object
  - jeod::DynBody, 67
- remove\_mass\_body
  - jeod::DynBody, 67
- reset\_controls
  - jeod::DynBody, 68
- reset\_force
  - jeod::Wrench, 117
- reset\_force\_and\_torque
  - jeod::Wrench, 117
- reset\_integrators
  - jeod::DynBody, 68
- reset\_point
  - jeod::Wrench, 117
- reset\_torque
  - jeod::Wrench, 118
- rigid\_attach\_parent
  - jeod::DynBodyGenericFrameAttachment, 83
- rigid\_attach\_state
  - jeod::DynBodyGenericFrameAttachment, 83
- rot\_accel
  - jeod::FrameDerivs, 90
- rot\_integ
  - jeod::DynBody, 68
  - jeod::StructureIntegratedDynBody, 100
- rot\_integrator
  - jeod::DynBody, 79
- rotation\_integration
  - jeod::DynBody, 79
- rotational\_dynamics
  - jeod::DynBody, 79
- scale\_force
  - jeod::Wrench, 118
- scale\_torque
  - jeod::Wrench, 118
- set
  - jeod::Wrench, 118
- set\_attitude\_left\_quaternion
  - jeod::DynBody, 68
- set\_attitude\_matrix
  - jeod::DynBody, 69
- set\_attitude\_rate
  - jeod::DynBody, 69
- set\_attitude\_right\_quaternion
  - jeod::DynBody, 69
- set\_force
  - jeod::Wrench, 118
- set\_integ\_frame
  - jeod::DynBody, 70
- set\_name
  - jeod::DynBody, 70
- set\_point
  - jeod::Wrench, 119
- set\_position
  - jeod::DynBody, 71
- set\_solver
  - jeod::StructureIntegratedDynBody, 100
- set\_state
  - jeod::DynBody, 71
- set\_state\_source
  - jeod::DynBody, 71
- set\_state\_source\_internal
  - jeod::DynBody, 72
- set\_torque
  - jeod::Wrench, 119
- set\_velocity
  - jeod::DynBody, 72
- solve\_constraints
  - jeod::StructureIntegratedDynBody, 101
- sort\_controls
  - jeod::DynBody, 72
- struct\_derivs
  - jeod::StructureIntegratedDynBody, 103
- structure
  - jeod::DynBody, 79
- structure\_integrated\_dyn\_body.cc, 135
- structure\_integrated\_dyn\_body.hh, 135
- structure\_integrated\_dyn\_body\_collect.cc, 136
- structure\_integrated\_dyn\_body\_integration.cc, 136
- structure\_integrated\_dyn\_body\_pt\_accel.cc, 137
- structure\_integrated\_dyn\_body\_solve.cc, 137
- structure\_to\_body\_offset
  - jeod::VehicleProperties, 112
- structure\_to\_body\_transform
  - jeod::VehicleProperties, 112
- StructureIntegratedDynBody
  - jeod::StructureIntegratedDynBody, 96
- switch\_integration\_frames
  - jeod::DynBody, 72, 73
- three\_dof
  - jeod::DynBody, 79
- time\_manager
  - jeod::DynBody, 80
- Torque
  - jeod::Torque, 104, 105
- torque
  - jeod::CollectTorque, 43
  - jeod::Torque, 105
  - jeod::Wrench, 120
- torque.cc, 138
- torque.hh, 138
- torque\_inline.hh, 139
- trans\_accel
  - jeod::FrameDerivs, 90
- trans\_integ
  - jeod::DynBody, 73
  - jeod::StructureIntegratedDynBody, 101
- trans\_integrator
  - jeod::DynBody, 80
- transform\_to\_parent
  - jeod::Wrench, 119
- transform\_to\_point
  - jeod::Wrench, 119
- translational\_dynamics
  - jeod::DynBody, 80

update\_integrated\_state  
    jeod::DynBody, [74](#)

vehicle\_non\_grav\_state.hh, [139](#)

vehicle\_points  
    jeod::DynBody, [80](#)

vehicle\_properties  
    jeod::StructureIntegratedDynBody, [103](#)

vehicle\_properties.hh, [140](#)

VehicleProperties  
    jeod::VehicleProperties, [109](#)

velocity\_source  
    jeod::DynBody, [80](#)

Wrench  
    jeod::Wrench, [114](#), [115](#)

wrench.hh, [140](#)