

DynamicBodyModel

5.0

Generated by Doxygen 1.8.5

Wed Jun 1 2022 12:08:57

Contents

1	Module Index	1
1.1	Modules	1
2	Namespace Index	3
2.1	Namespace List	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Data Structure Index	7
4.1	Data Structures	7
5	File Index	9
5.1	File List	9
6	Module Documentation	11
6.1	Models	11
6.1.1	Detailed Description	11
6.2	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
7	Namespace Documentation	15
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	16
7.1.2.3	accumulate_torques	17
7.1.2.4	accumulate_torques	17
7.1.2.5	check_frame_ownership	17

7.1.2.6	release_vector	17
8	Data Structure Documentation	19
8.1	jeod::BodyForceCollect Class Reference	19
8.1.1	Detailed Description	20
8.1.2	Constructor & Destructor Documentation	20
8.1.2.1	BodyForceCollect	20
8.1.2.2	BodyForceCollect	20
8.1.2.3	~BodyForceCollect	20
8.1.3	Member Function Documentation	21
8.1.3.1	operator=	21
8.1.4	Field Documentation	21
8.1.4.1	collect_effector_forc	21
8.1.4.2	collect_effector_torq	21
8.1.4.3	collect_environ_forc	21
8.1.4.4	collect_environ_torq	21
8.1.4.5	collect_no_xmit_forc	21
8.1.4.6	collect_no_xmit_torq	22
8.1.4.7	effector_forc	22
8.1.4.8	effector_torq	22
8.1.4.9	environ_forc	22
8.1.4.10	environ_torq	22
8.1.4.11	extern_forc_inrtl	22
8.1.4.12	extern_forc_struct	23
8.1.4.13	extern_torq_body	23
8.1.4.14	extern_torq_struct	23
8.1.4.15	inertial_torq	23
8.1.4.16	no_xmit_forc	23
8.1.4.17	no_xmit_torq	24
8.2	jeod::BodyRefFrame Class Reference	24
8.2.1	Detailed Description	25
8.2.2	Constructor & Destructor Documentation	25
8.2.2.1	BodyRefFrame	25
8.2.2.2	BodyRefFrame	25
8.2.2.3	~BodyRefFrame	25
8.2.3	Member Function Documentation	25
8.2.3.1	operator=	25
8.2.4	Friends And Related Function Documentation	25
8.2.4.1	init_attrjeod__BodyRefFrame	25
8.2.4.2	InputProcessor	25

8.2.5	Field Documentation	25
8.2.5.1	initialized_items	25
8.2.5.2	mass_point	25
8.3	jeod::BodyWrenchCollect Class Reference	26
8.3.1	Detailed Description	26
8.3.2	Constructor & Destructor Documentation	26
8.3.2.1	BodyWrenchCollect	26
8.3.2.2	~BodyWrenchCollect	27
8.3.2.3	BodyWrenchCollect	27
8.3.3	Member Function Documentation	27
8.3.3.1	accumulate	27
8.3.3.2	accumulate	27
8.3.3.3	operator=	27
8.3.4	Field Documentation	27
8.3.4.1	collect_wrench	27
8.4	jeod::CInterfaceForce Class Reference	28
8.4.1	Detailed Description	28
8.4.2	Constructor & Destructor Documentation	28
8.4.2.1	CInterfaceForce	28
8.4.2.2	CInterfaceForce	29
8.4.2.3	~CInterfaceForce	29
8.4.2.4	CInterfaceForce	29
8.4.3	Member Function Documentation	29
8.4.3.1	operator=	29
8.5	jeod::CInterfaceTorque Class Reference	29
8.5.1	Detailed Description	30
8.5.2	Constructor & Destructor Documentation	30
8.5.2.1	CInterfaceTorque	30
8.5.2.2	CInterfaceTorque	30
8.5.2.3	~CInterfaceTorque	30
8.5.2.4	CInterfaceTorque	31
8.5.3	Member Function Documentation	31
8.5.3.1	operator=	31
8.6	jeod::CollectForce Class Reference	31
8.6.1	Detailed Description	32
8.6.2	Constructor & Destructor Documentation	32
8.6.2.1	CollectForce	32
8.6.2.2	CollectForce	33
8.6.2.3	CollectForce	34
8.6.2.4	CollectForce	34

8.6.2.5	~CollectForce	34
8.6.2.6	CollectForce	34
8.6.3	Member Function Documentation	34
8.6.3.1	create	34
8.6.3.2	create	35
8.6.3.3	create	35
8.6.3.4	create	35
8.6.3.5	create	35
8.6.3.6	is_active	36
8.6.3.7	operator=	36
8.6.3.8	operator[]	36
8.6.3.9	operator[]	36
8.6.4	Field Documentation	37
8.6.4.1	active	37
8.6.4.2	force	37
8.7	jeod::CollectTorque Class Reference	37
8.7.1	Detailed Description	38
8.7.2	Constructor & Destructor Documentation	39
8.7.2.1	CollectTorque	39
8.7.2.2	CollectTorque	39
8.7.2.3	CollectTorque	39
8.7.2.4	CollectTorque	39
8.7.2.5	~CollectTorque	39
8.7.2.6	CollectTorque	39
8.7.3	Member Function Documentation	39
8.7.3.1	create	40
8.7.3.2	create	40
8.7.3.3	create	40
8.7.3.4	create	40
8.7.3.5	create	41
8.7.3.6	is_active	41
8.7.3.7	operator=	41
8.7.3.8	operator[]	41
8.7.3.9	operator[]	42
8.7.4	Field Documentation	42
8.7.4.1	active	42
8.7.4.2	torque	42
8.8	jeod::DynBody Class Reference	42
8.8.1	Detailed Description	48
8.8.2	Constructor & Destructor Documentation	48

8.8.2.1	DynBody	48
8.8.2.2	~DynBody	48
8.8.2.3	DynBody	48
8.8.3	Member Function Documentation	48
8.8.3.1	activate	48
8.8.3.2	add_control	48
8.8.3.3	add_integrable_object	49
8.8.3.4	add_mass_body	49
8.8.3.5	add_mass_body	49
8.8.3.6	add_mass_body_frames	49
8.8.3.7	add_mass_body_validate	49
8.8.3.8	add_mass_point	50
8.8.3.9	attach	50
8.8.3.10	attach	50
8.8.3.11	attach_establish_links	50
8.8.3.12	attach_update_properties	51
8.8.3.13	attach_validate_child	51
8.8.3.14	attach_validate_parent	52
8.8.3.15	clear_integrable_objects	52
8.8.3.16	collect_forces_and_torques	53
8.8.3.17	compute_derived_state_forward	53
8.8.3.18	compute_derived_state_reverse	53
8.8.3.19	compute_ref_point_transform	53
8.8.3.20	compute_state_elements_forward	54
8.8.3.21	compute_state_elements_reverse	54
8.8.3.22	compute_vehicle_point_derivatives	54
8.8.3.23	compute_vehicle_point_states	55
8.8.3.24	create_body_integrators	55
8.8.3.25	create_integrators	55
8.8.3.26	deactivate	56
8.8.3.27	destroy_integrators	56
8.8.3.28	detach	56
8.8.3.29	detach	56
8.8.3.30	detach_mass_body_frames	57
8.8.3.31	detach_mass_internal	57
8.8.3.32	find_body_frame	58
8.8.3.33	find_vehicle_point	58
8.8.3.34	get_dynamics_integration_group	58
8.8.3.35	get_initialized_states	59
8.8.3.36	get_integrable_objects	59

8.8.3.37	get_parent_body	59
8.8.3.38	get_parent_body_internal	59
8.8.3.39	get_root_body	59
8.8.3.40	get_root_body_internal	60
8.8.3.41	initialize_controls	60
8.8.3.42	initialize_model	60
8.8.3.43	initialized_states_contains	60
8.8.3.44	integrate	61
8.8.3.45	is_root_body	61
8.8.3.46	migrate_integrable_objects	61
8.8.3.47	operator=	61
8.8.3.48	process_dynamic_attachment	62
8.8.3.49	propagate_state	62
8.8.3.50	propagate_state_from_composite	62
8.8.3.51	propagate_state_from_structure	63
8.8.3.52	remove_integrable_object	63
8.8.3.53	remove_mass_body	63
8.8.3.54	reset_controls	64
8.8.3.55	reset_integrators	64
8.8.3.56	rot_integ	64
8.8.3.57	set_attitude_left_quaternion	64
8.8.3.58	set_attitude_matrix	65
8.8.3.59	set_attitude_rate	65
8.8.3.60	set_attitude_right_quaternion	65
8.8.3.61	set_integ_frame	66
8.8.3.62	set_integ_frame	66
8.8.3.63	set_name	66
8.8.3.64	set_position	67
8.8.3.65	set_state	68
8.8.3.66	set_state_source	68
8.8.3.67	set_state_source_internal	68
8.8.3.68	set_velocity	69
8.8.3.69	sort_controls	69
8.8.3.70	switch_integration_frames	69
8.8.3.71	switch_integration_frames	70
8.8.3.72	trans_integ	70
8.8.3.73	update_integrated_state	70
8.8.4	Friends And Related Function Documentation	71
8.8.4.1	init_attrjeod__DynBody	71
8.8.4.2	InputProcessor	71

8.8.5	Field Documentation	71
8.8.5.1	associated_integrable_objects	71
8.8.5.2	attitude_source	71
8.8.5.3	autoupdate_vehicle_points	71
8.8.5.4	collect	71
8.8.5.5	composite_body	71
8.8.5.6	core_body	72
8.8.5.7	derivs	72
8.8.5.8	dyn_children	72
8.8.5.9	dyn_manager	72
8.8.5.10	dyn_parent	72
8.8.5.11	grav_interaction	73
8.8.5.12	initialized_states	73
8.8.5.13	integ_frame	73
8.8.5.14	integ_frame_name	73
8.8.5.15	integ_results_merger	74
8.8.5.16	integrated_frame	74
8.8.5.17	mass	74
8.8.5.18	mass_children	74
8.8.5.19	name	74
8.8.5.20	position_source	75
8.8.5.21	rate_source	75
8.8.5.22	rot_integrator	75
8.8.5.23	rotation_integration	75
8.8.5.24	rotational_dynamics	75
8.8.5.25	structure	75
8.8.5.26	three_dof	76
8.8.5.27	time_manager	76
8.8.5.28	trans_integrator	76
8.8.5.29	translational_dynamics	76
8.8.5.30	vehicle_points	77
8.8.5.31	velocity_source	77
8.9	jeod::DynBodyMessages Class Reference	77
8.9.1	Detailed Description	78
8.9.2	Constructor & Destructor Documentation	78
8.9.2.1	DynBodyMessages	78
8.9.2.2	DynBodyMessages	78
8.9.3	Member Function Documentation	78
8.9.3.1	operator=	78
8.9.4	Friends And Related Function Documentation	78

8.9.4.1	init_attrjeod__DynBodyMessages	78
8.9.4.2	InputProcessor	78
8.9.5	Field Documentation	78
8.9.5.1	internal_error	78
8.9.5.2	invalid_attachment	79
8.9.5.3	invalid_body	79
8.9.5.4	invalid_frame	79
8.9.5.5	invalid_group	80
8.9.5.6	invalid_name	80
8.9.5.7	invalid_technique	80
8.9.5.8	not_dyn_body	80
8.10	jeod::Force Class Reference	81
8.10.1	Detailed Description	81
8.10.2	Constructor & Destructor Documentation	81
8.10.2.1	Force	81
8.10.2.2	~Force	82
8.10.2.3	Force	82
8.10.3	Member Function Documentation	82
8.10.3.1	operator=	82
8.10.3.2	operator[]	82
8.10.3.3	operator[]	82
8.10.4	Field Documentation	82
8.10.4.1	active	82
8.10.4.2	force	83
8.11	jeod::FrameDerivs Class Reference	83
8.11.1	Detailed Description	83
8.11.2	Constructor & Destructor Documentation	83
8.11.2.1	FrameDerivs	83
8.11.3	Field Documentation	84
8.11.3.1	non_grav_accel	84
8.11.3.2	Qdot_parent_this	84
8.11.3.3	rot_accel	84
8.11.3.4	trans_accel	84
8.12	jeod::JPVCollectForce Class Reference	84
8.12.1	Detailed Description	85
8.12.2	Member Function Documentation	85
8.12.2.1	perform_cleanup_action	85
8.13	jeod::JPVCollectTorque Class Reference	85
8.13.1	Detailed Description	86
8.13.2	Member Function Documentation	86

8.13.2.1	<code>perform_cleanup_action</code>	86
8.14	<code>jeod::StructureIntegratedDynBody</code> Class Reference	86
8.14.1	Detailed Description	88
8.14.2	Constructor & Destructor Documentation	88
8.14.2.1	<code>StructureIntegratedDynBody</code>	88
8.14.2.2	<code>~StructureIntegratedDynBody</code>	88
8.14.3	Member Function Documentation	88
8.14.3.1	<code>add_constraint</code>	88
8.14.3.2	<code>attach_update_properties</code>	88
8.14.3.3	<code>collect_forces_and_torques</code>	89
8.14.3.4	<code>collect_local_forces_and_torques</code>	89
8.14.3.5	<code>compute_vehicle_point_derivatives</code>	89
8.14.3.6	<code>detach</code>	90
8.14.3.7	<code>get_vehicle_properties</code>	90
8.14.3.8	<code>PropagateForcesAndTorques</code>	90
8.14.3.9	<code>rot_integ</code>	90
8.14.3.10	<code>set_solver</code>	91
8.14.3.11	<code>solve_constraints</code>	91
8.14.3.12	<code>trans_integ</code>	91
8.14.4	Friends And Related Function Documentation	92
8.14.4.1	<code>DynBodyConstraintsSolver</code>	92
8.14.4.2	<code>init_attrjeod__StructureIntegratedDynBody</code>	92
8.14.4.3	<code>InputProcessor</code>	92
8.14.5	Field Documentation	92
8.14.5.1	<code>constraints_solver</code>	92
8.14.5.2	<code>effector_wrench</code>	92
8.14.5.3	<code>effector_wrench_collection</code>	92
8.14.5.4	<code>inertial_accel_inrtl</code>	92
8.14.5.5	<code>inertial_accel_struct</code>	93
8.14.5.6	<code>inertial_accel_struct_omega</code>	93
8.14.5.7	<code>inertial_accel_struct_omega_dot</code>	93
8.14.5.8	<code>non_grav_state</code>	93
8.14.5.9	<code>struct_derivs</code>	93
8.14.5.10	<code>vehicle_properties</code>	93
8.15	<code>jeod::Torque</code> Class Reference	94
8.15.1	Detailed Description	94
8.15.2	Constructor & Destructor Documentation	95
8.15.2.1	<code>Torque</code>	95
8.15.2.2	<code>~Torque</code>	95
8.15.2.3	<code>Torque</code>	95

8.15.3	Member Function Documentation	95
8.15.3.1	operator=	95
8.15.3.2	operator[]	95
8.15.3.3	operator[]	95
8.15.4	Field Documentation	96
8.15.4.1	active	96
8.15.4.2	torque	96
8.16	jeod::VehicleNonGravState Class Reference	96
8.16.1	Detailed Description	97
8.16.2	Friends And Related Function Documentation	97
8.16.2.1	init_attrjeod__VehicleNonGravState	97
8.16.2.2	InputProcessor	97
8.16.3	Field Documentation	97
8.16.3.1	accel_struct	97
8.16.3.2	inertial_torque_struct	97
8.16.3.3	omega_body	97
8.16.3.4	omega_dot_body	97
8.16.3.5	omega_dot_struct	97
8.16.3.6	omega_struct	98
8.17	jeod::VehicleProperties Class Reference	98
8.17.1	Detailed Description	99
8.17.2	Constructor & Destructor Documentation	99
8.17.2.1	VehicleProperties	99
8.17.2.2	VehicleProperties	99
8.17.3	Member Function Documentation	100
8.17.3.1	get_inertia	100
8.17.3.2	get_inverse_inertia	100
8.17.3.3	get_inverse_mass	100
8.17.3.4	get_mass	100
8.17.3.5	get_parent_to_structure_offset	100
8.17.3.6	get_parent_to_structure_transform	101
8.17.3.7	get_structure_to_body_offset	101
8.17.3.8	get_structure_to_body_transform	101
8.17.4	Friends And Related Function Documentation	101
8.17.4.1	init_attrjeod__VehicleProperties	101
8.17.4.2	InputProcessor	101
8.17.5	Field Documentation	101
8.17.5.1	inertia	101
8.17.5.2	inverse_inertia	101
8.17.5.3	inverse_mass	102

8.17.5.4	mass	102
8.17.5.5	parent_to_structure_offset	102
8.17.5.6	parent_to_structure_transform	102
8.17.5.7	structure_to_body_offset	102
8.17.5.8	structure_to_body_transform	102
8.18	jeod::Wrench Class Reference	103
8.18.1	Detailed Description	104
8.18.2	Constructor & Destructor Documentation	105
8.18.2.1	Wrench	105
8.18.2.2	Wrench	105
8.18.2.3	Wrench	105
8.18.2.4	~Wrench	105
8.18.2.5	Wrench	105
8.18.2.6	Wrench	105
8.18.3	Member Function Documentation	106
8.18.3.1	accumulate	106
8.18.3.2	accumulate	106
8.18.3.3	activate	106
8.18.3.4	deactivate	106
8.18.3.5	get_force	106
8.18.3.6	get_point	106
8.18.3.7	get_torque	107
8.18.3.8	is_active	107
8.18.3.9	operator+=	107
8.18.3.10	operator=	107
8.18.3.11	operator=	107
8.18.3.12	reset_force	107
8.18.3.13	reset_force_and_torque	108
8.18.3.14	reset_point	108
8.18.3.15	reset_torque	108
8.18.3.16	scale_force	108
8.18.3.17	scale_torque	108
8.18.3.18	set	108
8.18.3.19	set_force	109
8.18.3.20	set_force	109
8.18.3.21	set_point	109
8.18.3.22	set_torque	109
8.18.3.23	transform_to_parent	109
8.18.3.24	transform_to_point	110
8.18.4	Friends And Related Function Documentation	110

8.18.4.1	init_attrjeod__Wrench	110
8.18.4.2	InputProcessor	110
8.18.5	Field Documentation	110
8.18.5.1	active	110
8.18.5.2	force	110
8.18.5.3	point	110
8.18.5.4	torque	111
9	File Documentation	113
9.1	aux_classes.cc File Reference	113
9.1.1	Detailed Description	113
9.2	body_force_collect.hh File Reference	113
9.2.1	Detailed Description	114
9.3	body_ref_frame.hh File Reference	114
9.3.1	Detailed Description	114
9.4	body_wrench_collect.cc File Reference	115
9.4.1	Detailed Description	115
9.5	body_wrench_collect.hh File Reference	115
9.5.1	Detailed Description	115
9.6	class_declarations.hh File Reference	115
9.6.1	Detailed Description	116
9.7	dyn_body.cc File Reference	116
9.7.1	Detailed Description	116
9.8	dyn_body.hh File Reference	116
9.8.1	Detailed Description	117
9.9	dyn_body_attach.cc File Reference	117
9.9.1	Detailed Description	117
9.10	dyn_body_collect.cc File Reference	117
9.10.1	Detailed Description	118
9.11	dyn_body_detach.cc File Reference	118
9.11.1	Detailed Description	118
9.12	dyn_body_find_body_frame.cc File Reference	119
9.12.1	Detailed Description	119
9.13	dyn_body_initialize_model.cc File Reference	119
9.13.1	Detailed Description	119
9.14	dyn_body_integration.cc File Reference	119
9.14.1	Detailed Description	120
9.15	dyn_body_messages.cc File Reference	120
9.15.1	Detailed Description	120
9.16	dyn_body_messages.hh File Reference	121

9.16.1 Detailed Description	121
9.17 dyn_body_propagate_state.cc File Reference	121
9.17.1 Detailed Description	121
9.18 dyn_body_set_state.cc File Reference	121
9.18.1 Detailed Description	122
9.19 dyn_body_vehicle_point.cc File Reference	122
9.19.1 Detailed Description	122
9.20 force.cc File Reference	123
9.20.1 Detailed Description	123
9.21 force.hh File Reference	123
9.21.1 Detailed Description	123
9.22 force_inline.hh File Reference	124
9.22.1 Detailed Description	124
9.23 frame_derivs.hh File Reference	124
9.23.1 Detailed Description	124
9.24 structure_integrated_dyn_body.cc File Reference	124
9.24.1 Detailed Description	125
9.25 structure_integrated_dyn_body.hh File Reference	125
9.25.1 Detailed Description	125
9.26 structure_integrated_dyn_body_collect.cc File Reference	125
9.26.1 Detailed Description	126
9.27 structure_integrated_dyn_body_integration.cc File Reference	126
9.27.1 Detailed Description	126
9.28 structure_integrated_dyn_body_pt_accel.cc File Reference	126
9.28.1 Detailed Description	127
9.29 structure_integrated_dyn_body_solve.cc File Reference	127
9.29.1 Detailed Description	127
9.30 torque.cc File Reference	127
9.30.1 Detailed Description	127
9.31 torque.hh File Reference	128
9.31.1 Detailed Description	128
9.32 torque_inline.hh File Reference	128
9.32.1 Detailed Description	128
9.33 vehicle_non_grav_state.hh File Reference	128
9.33.1 Detailed Description	129
9.34 vehicle_properties.hh File Reference	129
9.34.1 Detailed Description	129
9.35 wrench.hh File Reference	129
9.35.1 Detailed Description	130

Index**131**

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Models	11
Dynamics	12
DynBody	13

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

jeod	Namespace jeod	15
----------------------	--------------------------	----

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

jeod::BodyForceCollect	19
jeod::BodyWrenchCollect	26
jeod::CollectForce	31
jeod::CInterfaceForce	28
jeod::CollectTorque	37
jeod::CInterfaceTorque	29
jeod::DynBodyMessages	77
jeod::Force	81
jeod::FrameDerivs	83
IntegrableObject	
jeod::DynBody	42
jeod::StructureIntegratedDynBody	86
RefFrame	
jeod::BodyRefFrame	24
RefFrameOwner	
jeod::DynBody	42
jeod::Torque	94
type	
jeod::JPVCollectForce	84
jeod::JPVCollectTorque	85
jeod::VehicleNonGravState	96
jeod::VehicleProperties	98
jeod::Wrench	103

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

jeod::BodyForceCollect	Serves as the collection point for forces and torques that act on a vehicle	19
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	24
jeod::BodyWrenchCollect	Serves as the collection point for wrenches that act on a vehicle	26
jeod::CInterfaceForce	This class is deprecated	28
jeod::CInterfaceTorque	This class is deprecated	29
jeod::CollectForce	A CollectForce represents a collected force that acts on a vehicle	31
jeod::CollectTorque	A CollectTorque represents a collected torque that acts on a vehicle	37
jeod::DynBody	Class DynBody is the base class for all dynamic bodies	42
jeod::DynBodyMessages	Specify the message IDs used in the DynBody model	77
jeod::Force	A Force represents a Newtonian force that acts on a DynBody	81
jeod::FrameDerivs	Contains translational and rotational second derivatives	83
jeod::JPVCollectForce	This is a derived version of the template class <code>JeodPointerVector<CollectForce>::type</code> with an implementation of the method <code>perform_cleanup_action</code> which frees and clears stale data following a restore	84
jeod::JPVCollectTorque	This is a derived version of the template class <code>JeodPointerVector<CollectTorque>::type</code> with an implementation of the method <code>perform_cleanup_action</code> which frees and clears stale data following a restore	85
jeod::StructureIntegratedDynBody	Extends DynBody to integrate an object's structural reference frame as opposed to its center of mass	86
jeod::Torque	A Torque represents a Newtonian torque that acts on a DynBody	94
jeod::VehicleNonGravState	Encapsulates various aspects of a vehicle's state with respect to inertial	96

[jeod::VehicleProperties](#)

Captures pointers to various vehicle properties that are commonly used in the constraint concept [98](#)

[jeod::Wrench](#)

A wrench comprises a torque and a force applied at a point on a [DynBody](#) [103](#)

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

aux_classes.cc	Define base methods for various small JEOD DynBody classes	113
body_force_collect.hh	Define the class BodyForceCollect	113
body_ref_frame.hh	Define the class BodyRefFrame	114
body_wrench_collect.cc	Define BodyWrenchCollect member functions	115
body_wrench_collect.hh	Defines the class BodyWrenchCollect	115
class_declarations.hh	Forward declarations of classes defined in dyn_body.hh	115
dyn_body.cc	Define base methods for the DynBody class	116
dyn_body.hh	Define the class DynBody	116
dyn_body_attach.cc	Define DynBody attachment methods	117
dyn_body_collect.cc	Define DynBody methods related to force and torque accumulation and propagation	117
dyn_body_detach.cc	Define DynBody detachment methods	118
dyn_body_find_body_frame.cc	Define DynBody::find_body_frame	119
dyn_body_initialize_model.cc	Define DynBody::initialize_model	119
dyn_body_integration.cc	Define methods for frame switching	119
dyn_body_messages.cc	Implement the class De4xxMessages	120
dyn_body_messages.hh	Define the class DynBodyMessages	121
dyn_body_propagate_state.cc	Define DynBody state propagation / update methods	121
dyn_body_set_state.cc	Define methods related to setting aspects of a vehicle's state	121
dyn_body_vehicle_point.cc	Define methods that support vehicle points	122

force.cc	Define force model member functions	123
force.hh	Define the JEOD force model	123
force_inline.hh	Inline functions for the JEOD force model	124
frame_derivs.hh	Define the FrameDerivs class	124
structure_integrated_dyn_body.cc	Define base member functions for StructureIntegratedDynBody	124
structure_integrated_dyn_body.hh	Define the class StructureIntegratedDynBody, which integrates a DynBody object's structural state	125
structure_integrated_dyn_body_collect.cc	Define StructureIntegratedDynBody methods related to force and torque accumulation and propagation	125
structure_integrated_dyn_body_integration.cc	Define StructureIntegratedDynBody member functions related to state integration	126
structure_integrated_dyn_body_pt_accel.cc	Define StructureIntegratedDynBody::compute_vehicle_point_derivatives	126
structure_integrated_dyn_body_solve.cc	Define StructureIntegratedDynBody methods related to force and torque accumulation and propagation	127
torque.cc	Define torque model member functions	127
torque.hh	Define the JEOD torque model	128
torque_inline.hh	Define the JEOD torque model	128
vehicle_non_grav_state.hh	Define the class VehicleNonGravState	128
vehicle_properties.hh	Define the class VehicleProperties	129
wrench.hh	Define the class Wrench	129

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_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 38 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 [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.
- 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 72 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 94 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 FILE
in	<i>line</i>	Typically LINE

Definition at line 63 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<class CollectType > void jeod::release_vector (CollectType & vec)`

Release JEOD-allocated memory in the collect vector.

Parameters

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

Definition at line 49 of file `body_force_collect.hh`.

Referenced by `jeod::JPVCollectForce::perform_cleanup_action()`, `jeod::JPVCollectTorque::perform_cleanup_action()`, and `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 125 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 44 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 84 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 204 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 219 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 209 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 224 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 214 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 229 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 148 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 173 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 153 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 178 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 168 of file body_force_collect.hh.

Referenced by BodyForceCollect(), jeod::StructureIntegratedDynBody::collect_forces_and_torques(), and jeod::DynBody::collect_forces_and_torques().

8.1.4.12 double jeod::BodyForceCollect::extern_forc_struct[3]

Sum of external forces, struct ref.

trick_units(N)

Definition at line 163 of file body_force_collect.hh.

Referenced by BodyForceCollect(), jeod::StructureIntegratedDynBody::collect_forces_and_torques(), jeod::DynBody::collect_forces_and_torques(), 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 199 of file body_force_collect.hh.

Referenced by BodyForceCollect(), jeod::StructureIntegratedDynBody::collect_forces_and_torques(), and jeod::DynBody::collect_forces_and_torques().

8.1.4.14 double jeod::BodyForceCollect::extern_torq_struct[3]

Sum of external torques, struct ref.

trick_units(N*m)

Definition at line 194 of file body_force_collect.hh.

Referenced by BodyForceCollect(), jeod::StructureIntegratedDynBody::collect_forces_and_torques(), and jeod::DynBody::collect_forces_and_torques().

8.1.4.15 double jeod::BodyForceCollect::inertial_torq[3]

Induced inertial torques from second order rotational dynamics, w x lw, body ref.

trick_units(N*m)

Definition at line 189 of file body_force_collect.hh.

Referenced by BodyForceCollect(), jeod::StructureIntegratedDynBody::collect_forces_and_torques(), jeod::DynBody::collect_forces_and_torques(), 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 158 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 183 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:

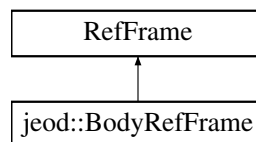
- [body_force_collect.hh](#)
- [aux_classes.cc](#)

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)
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 44 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 91 of file body_ref_frame.hh.

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

Destructor.

Definition at line 105 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 46 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 57 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 64 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()`, `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 44 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>	Wrench into which the accumulated sum is to be placed. The summation is about <code>sum.point</code> .
------------	--

Returns

Reference to the input wrench.

Definition at line 95 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>	Wrench into which the accumulated sum is to be placed.

Returns

Reference to the input wrench.

Definition at line 107 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 64 of file `body_wrench_collect.hh`.

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

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

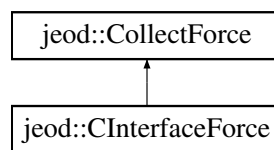
- [body_wrench_collect.hh](#)
- [body_wrench_collect.cc](#)

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.
- virtual [~CInterfaceForce](#) ()
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 188 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 142 of file force.cc.

8.4.2.2 jeod::CInterfaceForce::CInterfaceForce (double * force_3vec)

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

Note that the new [CInterfaceForce](#)'s force *is* the force_3vec.

Parameters

<i>in, out</i>	<i>force_3vec</i>	Force vector to encapsulate Units: N
----------------	-------------------	---

Definition at line 156 of file force.cc.

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

8.4.2.3 jeod::CInterfaceForce::~~CInterfaceForce (void) [virtual]

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

Definition at line 169 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:

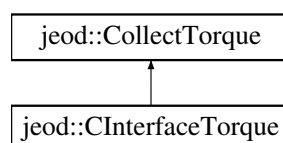
- [force.hh](#)
- [force.cc](#)

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.
- virtual [~CInterfaceTorque](#) ()
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 184 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 142 of file torque.cc.

8.5.2.2 jeod::CInterfaceTorque::CInterfaceTorque (double * torque_3vec)

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

Note that the new [CInterfaceTorque](#)'s torque *is* the torque_3vec.

Parameters

in, out	<i>torque_3vec</i>	Torque vector to encapsulate Units: NM
---------	--------------------	---

Definition at line 156 of file torque.cc.

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

8.5.2.3 jeod::CInterfaceTorque::~~CInterfaceTorque (void) [virtual]

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

Definition at line 169 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:

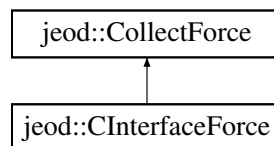
- [torque.hh](#)
- [torque.cc](#)

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.

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 115 of file force.hh.

8.6.2 Constructor & Destructor Documentation

8.6.2.1 jeod::CollectForce::CollectForce (void)

[CollectForce](#) default constructor.

Definition at line 69 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>	Force vector to encapsulate Units: N
----------------	-------------------	---

Definition at line 99 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>	Force to encapsulate
----------------	---------------------	--------------------------------------

Definition at line 84 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>	Force to encapsulate
----------------	---------------------	--------------------------------------

Definition at line 114 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 128 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>	Force vector to encapsulate Units: N
----------------	-------------------	---

Definition at line 216 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>	Force object to encapsulate
----------------	---------------------	---

Definition at line 187 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>	Force to copy
----------------	---------------------	-------------------------------

Definition at line 231 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>	Force object to encapsulate
----------------	---------------------	---

Definition at line 202 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>	Force to copy
----------------	---------------------	-------------------------------

Definition at line 246 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 70 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 `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 84 of file force_inline.hh.

References [force](#).

8.6.3.9 `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 97 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 158 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 163 of file `force.hh`.

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

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

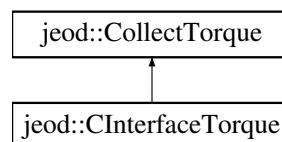
- [force.hh](#)
- [force_inline.hh](#)
- [force.cc](#)

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.

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 113 of file `torque.hh`.

8.7.2 Constructor & Destructor Documentation

8.7.2.1 jeod::CollectTorque::CollectTorque (void)

[CollectTorque](#) default constructor.

Definition at line 69 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>	Torque vector to encapsulate Units: NM
----------------	--------------------	---

Definition at line 99 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>	Torque to encapsulate
----------------	----------------------	---------------------------------------

Definition at line 84 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>	Torque to encapsulate
----------------	----------------------	---------------------------------------

Definition at line 114 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 128 of file torque.cc.

8.7.2.6 jeod::CollectTorque::CollectTorque (const [CollectTorque](#) &) [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

<code>in, out</code>	<code>torque_3vec</code>	Torque vector to encapsulate Units: NM
----------------------	--------------------------	---

Definition at line 216 of file torque.cc.

Referenced by `create()`.

8.7.3.2 `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

<code>in, out</code>	<code>source_torque</code>	Torque object to encapsulate
----------------------	----------------------------	--

Definition at line 187 of file torque.cc.

8.7.3.3 `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

<code>in, out</code>	<code>source_torque</code>	Torque to copy
----------------------	----------------------------	--------------------------------

Definition at line 231 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

<i>in, out</i>	<i>source_torque</i>	Torque object to encapsulate
----------------	----------------------	--

Definition at line 202 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

<i>in, out</i>	<i>source_torque</i>	Torque to copy
----------------	----------------------	--------------------------------

Definition at line 246 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 70 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 `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 84 of file torque_inline.hh.

References [torque](#).

8.7.3.9 `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

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

Definition at line 97 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 154 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 159 of file `torque.hh`.

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

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

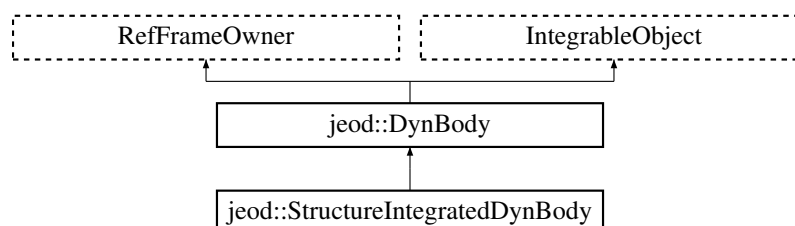
- [torque.hh](#)
- [torque_inline.hh](#)
- [torque.cc](#)

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.
- virtual [~DynBody](#) ()
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](#) (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.
- virtual er7_utils::IntegratorResult [integrate](#) (double dyn_dt, unsigned int target_stage)
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.
- virtual void [create_integrators](#) (const er7_utils::IntegratorConstructor &generator, er7_utils::IntegrationControls &controls, const er7_utils::TimeInterface &time_if)
This interface is required by er7_utils::IntegrableObject.
- virtual void [destroy_integrators](#) (void)
Destroy the integrators.
- virtual void [reset_integrators](#) (void)
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](#) (double offset[3], double T_pstr_cstr[3][3], MassBody &child)
 - virtual bool [attach](#) (const char *this_point_name, const char *child_point_name, [DynBody](#) &child)
 - virtual bool [attach](#) (double offset[3], double T_pstr_cstr[3][3], [DynBody](#) &child)
 - 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.
- 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 [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](#) (double offset_pstr_cstr_pstr[3], 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](#) (double offset_pstr_cstr_pstr[3], 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.
- 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 87 of file `dyn_body.hh`.

8.8.2 Constructor & Destructor Documentation

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

[DynBody](#) default constructor.

Definition at line 69 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 ()` `[virtual]`

[DynBody](#) destructor.

Definition at line 118 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 125 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 227 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

in	<i>associated_integrable_object</i>	The IntegrableObject to be associated with this DynBody .
----	-------------------------------------	---

Definition at line 295 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 453 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 (double *offset[3]*, double *T_pstr_cstr[3][3]*, MassBody & *child*) [virtual]

Definition at line 577 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 665 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 194 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 54 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 (const char * *this_point_name*, const char * *child_point_name*, DynBody & *child*) [virtual]

Definition at line 238 of file dyn_body_attach.cc.

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

8.8.3.10 bool jeod::DynBody::attach (double *offset[3]*, double *T_pstr_cstr[3][3]*, DynBody & *child*) [virtual]

Definition at line 371 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

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

Definition at line 708 of file `dyn_body_attach.cc`.

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

Referenced by `attach()`.

8.8.3.12 `void jeod::DynBody::attach_update_properties (double offset_pstr_cstr_pstr[3], 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

<i>in</i>	<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
<i>in</i>	<i>T_pstr_cstr</i>	Transformation matrix from the new parent body's structural frame to this body's structural frame.
<i>in, out</i>	<i>child</i>	The child body; the body newly attached to this body.

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

Definition at line 734 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()`, and `jeod::StructureIntegratedDynBody::attach_update_properties()`.

8.8.3.13 `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 119 of file dyn_body_attach.cc.

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

Referenced by `attach()`.

8.8.3.14 `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 67 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()`.

8.8.3.15 `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 335 of file dyn_body.cc.

References `associated_integrable_objects`.

8.8.3.16 void jeod::DynBody::collect_forces_and_torques () [virtual]

Collect forces and torques acting on the vehicle.

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

Definition at line 111 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_environt_forc](#), [jeod::BodyForceCollect::collect_environt_torq](#), [collect_forces_and_torques\(\)](#), [jeod::BodyForceCollect::collect_no_xmit_forc](#), [jeod::BodyForceCollect::collect_no_xmit_torq](#), [composite_body](#), [derivs](#), [dyn_children](#), [dyn_parent](#), [jeod::BodyForceCollect::effector_forc](#), [jeod::BodyForceCollect::effector_torq](#), [jeod::BodyForceCollect::environt_forc](#), [jeod::BodyForceCollect::environt_torq](#), [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\(\)](#).

8.8.3.17 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 161 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\(\)](#).

8.8.3.18 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 280 of file dyn_body_propagate_state.cc.

References [jeod::BodyRefFrame::initialized_items](#).

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

8.8.3.19 void jeod::DynBody::compute_ref_point_transform (const BodyRefFrame & source_frame, const MassPoint & 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 52 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()`.

8.8.3.20 `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 216 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()`.

8.8.3.21 `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 335 of file `dyn_body_propagate_state.cc`.

References `jeod::BodyRefFrame::initialized_items`.

Referenced by `propagate_state_from_composite()`.

8.8.3.22 `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>frame</i>	Vehicle point reference frame
out	<i>derivs</i>	Computed derivatives

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

Definition at line 131 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`.

8.8.3.23 `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 792 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()`.

8.8.3.24 `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 223 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()`.

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

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 263 of file dyn_body_integration.cc.

References `create_body_integrators()`, and `jeod::DynBodyMessages::internal_error`.

8.8.3.26 void jeod::DynBody::deactivate () [inline]

Deactivate a [DynBody](#) object.

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

Definition at line 133 of file dyn_body.hh.

8.8.3.27 void jeod::DynBody::destroy_integrators (void) [virtual]

Destroy the integrators.

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

Definition at line 289 of file dyn_body_integration.cc.

8.8.3.28 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>dyn_body</i>	The other body at which the detach will occur
----	-----------------	---

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

Definition at line 60 of file dyn_body_detach.cc.

References `detach_mass_internal()`, `dyn_children`, `dyn_parent`, `jeod::DynBodyMessages::invalid_attachment`, `mass`, and `name`.

Referenced by `~DynBody()`.

8.8.3.29 bool jeod::DynBody::detach (void) [virtual]

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

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 148 of file dyn_body_detach.cc.

References dyn_parent, jeod::DynBodyMessages::invalid_technique, and name.

Referenced by jeod::StructureIntegratedDynBody::detach(), remove_mass_body(), and ~DynBody().

8.8.3.30 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 245 of file dyn_body_detach.cc.

References dyn_manager, find_body_frame(), and vehicle_points.

Referenced by remove_mass_body().

8.8.3.31 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 291 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().

8.8.3.32 **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 51 of file dyn_body_find_body_frame.cc.

References dyn_manager, jeod::DynBodyMessages::invalid_name, and name.

Referenced by detach_mass_body_frames().

8.8.3.33 **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 102 of file dyn_body_vehicle_point.cc.

References name, and vehicle_points.

Referenced by add_mass_body(), and attach().

8.8.3.34 **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 266 of file dyn_body.cc.

References jeod::DynBodyMessages::internal_error.

Referenced by attach_update_properties(), and set_integ_frame().

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

Indicate which state elements have been initialized.

Returns

Initialized states indicator.

Definition at line 499 of file `dyn_body.hh`.

References `initialized_states`.

8.8.3.36 `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 282 of file `dyn_body.hh`.

References `associated_integrable_objects`.

8.8.3.37 `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 182 of file `dyn_body.cc`.

References `dyn_parent`.

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

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

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

Returns

Pointer to parent body.

Definition at line 191 of file `dyn_body.cc`.

References `dyn_parent`.

8.8.3.39 `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 199 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()`.

8.8.3.40 `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 210 of file `dyn_body.cc`.

References `dyn_parent`.

Referenced by `attach()`, `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()`.

8.8.3.41 `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 237 of file `dyn_body.cc`.

References `dyn_manager`, and `grav_interaction`.

8.8.3.42 `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 48 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`.

8.8.3.43 `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 509 of file dyn_body.hh.

References `initialized_states`.

8.8.3.44 `er7_utils::IntegratorResult jeod::DynBody::integrate (double dyn_dt, unsigned int target_stage) [virtual]`

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 320 of file dyn_body_integration.cc.

References `initialized_states`, `integ_results_merger`, `propagate_state()`, `rot_integ()`, `rotational_dynamics`, `trans_integ()`, and `translational_dynamics`.

8.8.3.45 `bool jeod::DynBody::is_root_body ()`

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

Returns

Is this a root body?

Definition at line 174 of file dyn_body.cc.

References `dyn_parent`.

8.8.3.46 `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 342 of file dyn_body.cc.

References `associated_integrable_objects`, `jeod::DynBodyMessages::invalid_group`, and `name`.

8.8.3.47 `DynBody& jeod::DynBody::operator= (const DynBody &) [private]`

Not implemented.

8.8.3.48 `void jeod::DynBody::process_dynamic_attachment (double offset_pstr_cstr_pstr[3], 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 816 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()`.

8.8.3.49 `void jeod::DynBody::propagate_state ()` `[virtual]`

Propagate state from the integrated state to attached bodies.

Definition at line 577 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()`.

8.8.3.50 `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 703 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()`.

8.8.3.51 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 611 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()`.

8.8.3.52 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 DynBody .
----	-------------------------------------	---

Definition at line 314 of file dyn_body.cc.

References `associated_integrable_objects`.

8.8.3.53 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>parent</i>	The parent body; the body from which this body is to be detached.
----------------	---------------	---

Definition at line 172 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()`.

8.8.3.54 `void jeod::DynBody::reset_controls () [virtual]`

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

Definition at line 248 of file `dyn_body.cc`.

References `dyn_manager`, and `grav_interaction`.

8.8.3.55 `void jeod::DynBody::reset_integrators (void) [virtual]`

Reset the translational and rotational integrators.

Definition at line 299 of file `dyn_body_integration.cc`.

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

8.8.3.56 `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 380 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()`.

8.8.3.57 `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 219 of file dyn_body_set_state.cc.

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

8.8.3.58 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 257 of file dyn_body_set_state.cc.

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

8.8.3.59 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 276 of file dyn_body_set_state.cc.

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

8.8.3.60 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 238 of file `dyn_body_set_state.cc`.

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

8.8.3.61 `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 66 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()`.

8.8.3.62 `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 133 of file `dyn_body_integration.cc`.

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

8.8.3.63 `void jeod::DynBody::set_name (std::string name_in)`

Set the name of the vehicle.

Parameters

in	<i>vehicle</i>	name
----	----------------	------

Definition at line 166 of file dyn_body.cc.

References mass.

8.8.3.64 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 185 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_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 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 80 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_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 133 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().

8.8.3.67 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 294 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().

8.8.3.68 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 202 of file dyn_body_set_state.cc.

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

8.8.3.69 void jeod::DynBody::sort_controls () [virtual]

Sort the gravity controls in ascending acceleration magnitude order.

Definition at line 257 of file dyn_body.cc.

References grav_interaction.

8.8.3.70 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 153 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()`.

8.8.3.71 `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 195 of file dyn_body_integration.cc.

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

8.8.3.72 `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 358 of file `dyn_body_integration.cc`.

References `composite_body`, `derivs`, `jeod::FrameDerivs::trans_accel`, and `trans_integrator`.

Referenced by `integrate()`.

8.8.3.73 void jeod::DynBody::update_integrated_state () [virtual]

Propagate state from state owners to the integrated state.

Definition at line 399 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 90 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 1145 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 1127 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 690 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 709 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::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 633 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 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 625 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().

8.8.5.7 FrameDerivs jeod::DynBody::derivs

Translational/rotational accelerations.

trick_units(-)

Definition at line 703 of file dyn_body.hh.

Referenced by jeod::StructureIntegratedDynBody::collect_forces_and_torques(), collect_forces_and_torques(), compute_vehicle_point_derivatives(), jeod::StructureIntegratedDynBody::rot_integ(), rot_integ(), jeod::StructureIntegratedDynBody::solve_constraints(), and trans_integ().

8.8.5.8 std::list<DynBody*> jeod::DynBody::dyn_children [protected]

The subset of the dynamic bodies attached to this dynamic body.

trick_io(**)

Definition at line 1095 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().

8.8.5.9 BaseDynManager* & jeod::DynBody::dyn_manager [protected]

The dynamics manager for the simulation.

trick_units(-)

Definition at line 1077 of file dyn_body.hh.

Referenced by add_mass_body_frames(), add_mass_body_validate(), add_mass_point(), attach_validate_parent(), detach_mass_body_frames(), find_body_frame(), initialize_controls(), initialize_model(), reset_controls(), set_integ_frame(), switch_integration_frames(), and ~DynBody().

8.8.5.10 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 1090 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().

8.8.5.11 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 698 of file dyn_body.hh.

Referenced by add_control(), collect_forces_and_torques(), jeod::StructureIntegratedDynBody::compute_vehicle_point_derivatives(), compute_vehicle_point_derivatives(), initialize_controls(), reset_controls(), set_integ_frame(), and sort_controls().

8.8.5.12 RefFrameItems jeod::DynBody::initialized_states [protected]

Enum value indicating which of position, velocity, attitude, and rate have been initialized.

trick_units(-)

Definition at line 1112 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().

8.8.5.13 EphemerisRefFrame* jeod::DynBody::integ_frame

The current integration frame.

trick_units(-)

Definition at line 617 of file dyn_body.hh.

Referenced by add_mass_body_frames(), add_mass_point(), attach_establish_links(), initialize_model(), and set_integ_frame().

8.8.5.14 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 612 of file dyn_body.hh.

Referenced by initialize_model().

8.8.5.15 er7_utils::IntegratorResultMergerContainer jeod::DynBody::integ_results_merger [protected]

The object that merges integration results.

trick_units(-)

Definition at line 1151 of file dyn_body.hh.

Referenced by create_body_integrators(), and integrate().

8.8.5.16 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 1138 of file dyn_body.hh.

Referenced by compute_ref_point_transform(), DynBody(), propagate_state(), jeod::StructureIntegratedDynBody::StructureIntegratedDynBody(), switch_integration_frames(), and update_integrated_state().

8.8.5.17 MassBody jeod::DynBody::mass

Mass properties of the vehicle, defined about the structure reference frame.

Definition at line 594 of file dyn_body.hh.

Referenced by add_mass_body(), add_mass_point(), attach(), attach_establish_links(), attach_update_properties(), jeod::StructureIntegratedDynBody::collect_forces_and_torques(), collect_forces_and_torques(), compute_ref_point_transform(), 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().

8.8.5.18 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.

trick_io(**)

Definition at line 1101 of file dyn_body.hh.

Referenced by add_mass_body(), remove_mass_body(), and ~DynBody().

8.8.5.19 NamedItem& jeod::DynBody::name

Body name, reference linked to mass.name.

trick_units(–)

Definition at line 599 of file dyn_body.hh.

Referenced by add_mass_body(), add_mass_body_frames(), add_mass_body_validate(), add_mass_point(), attach(), 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().

8.8.5.20 BodyRefFrame* jeod::DynBody::position_source [protected]

The reference frame that contains the user-set position.

trick_units(–)

Definition at line 1117 of file dyn_body.hh.

Referenced by set_state_source_internal(), and update_integrated_state().

8.8.5.21 BodyRefFrame* jeod::DynBody::rate_source [protected]

The reference frame that contains the user-set attitude rate.

trick_units(–)

Definition at line 1132 of file dyn_body.hh.

Referenced by set_state_source_internal(), and update_integrated_state().

8.8.5.22 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 1168 of file dyn_body.hh.

Referenced by create_body_integrators(), DynBody(), reset_integrators(), jeod::StructureIntegratedDynBody::rot_integ(), rot_integ(), and ~DynBody().

8.8.5.23 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 680 of file dyn_body.hh.

Referenced by create_body_integrators().

8.8.5.24 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 659 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().

8.8.5.25 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 641 of file dyn_body.hh.

Referenced by attach_update_properties(), collect_forces_and_torques(), compute_ref_point_transform(), jeod::StructureIntegratedDynBody::compute_vehicle_point_derivatives(), compute_vehicle_point_states(), DynBody(), initialize_model(), 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().

8.8.5.26 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 671 of file dyn_body.hh.

Referenced by create_body_integrators().

8.8.5.27 const JeodIntegrationTime* jeod::DynBody::time_manager [protected]

The time manager to be used to obtain timestamp information.

trick_units(-)

Definition at line 1082 of file dyn_body.hh.

Referenced by create_body_integrators(), and update_integrated_state().

8.8.5.28 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 1159 of file dyn_body.hh.

Referenced by `create_body_integrators()`, `DynBody()`, `reset_integrators()`, `jeod::StructureIntegratedDynBody::trans_integ()`, `trans_integ()`, and `~DynBody()`.

8.8.5.29 `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 650 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()`.

8.8.5.30 `std::list<BodyRefFrame*> jeod::DynBody::vehicle_points` `[protected]`

An array of vehicle points associated with this dynamic body.

`trick_io(**)`

Definition at line 1106 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()`.

8.8.5.31 `BodyRefFrame* jeod::DynBody::velocity_source` `[protected]`

The reference frame that contains the user-set velocity.

`trick_units(-)`

Definition at line 1122 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::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](#) ()

8.9.1 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 46 of file [dyn_body_messages.hh](#).

8.9.2 Constructor & Destructor Documentation

8.9.2.1 [jeod::DynBodyMessages::DynBodyMessages \(void \)](#) `[private]`

8.9.2.2 [jeod::DynBodyMessages::DynBodyMessages \(const DynBodyMessages & \)](#) `[private]`

8.9.3 Member Function Documentation

8.9.3.1 [DynBodyMessages& jeod::DynBodyMessages::operator= \(const DynBodyMessages & \)](#) `[private]`

8.9.4 Friends And Related Function Documentation

8.9.4.1 `void init_attrjeod__DynBodyMessages () [friend]`

8.9.4.2 `friend class InputProcessor [friend]`

Definition at line 48 of file `dyn_body_messages.hh`.

8.9.5 Field Documentation

8.9.5.1 `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 95 of file `dyn_body_messages.hh`.

Referenced by `jeod::DynBody::create_integrators()`, and `jeod::DynBody::get_dynamics_integration_group()`.

8.9.5.2 `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 78 of file `dyn_body_messages.hh`.

Referenced by `jeod::DynBody::add_mass_body()`, `jeod::DynBody::attach()`, `jeod::StructureIntegratedDynBody::attach_update_properties()`, `jeod::DynBody::attach_validate_child()`, `jeod::DynBody::attach_validate_parent()`, `jeod::StructureIntegratedDynBody::detach()`, and `jeod::DynBody::detach()`.

8.9.5.3 `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 58 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()`.

8.9.5.4 `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 73 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()`.

8.9.5.5 `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 63 of file `dyn_body_messages.hh`.

Referenced by `jeod::DynBody::migrate_integrable_objects()`.

8.9.5.6 `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 68 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()`.

8.9.5.7 `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 83 of file `dyn_body_messages.hh`.

Referenced by `jeod::DynBody::detach()`, and `jeod::DynBody::remove_mass_body()`.

8.9.5.8 `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 89 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](#)

8.10 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.

8.10.1 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 48 of file force.hh.

8.10.2 Constructor & Destructor Documentation

8.10.2.1 `jeod::Force::Force (void)`

[Force](#) default constructor.

Definition at line 46 of file force.cc.

References [force](#).

8.10.2.2 `jeod::Force::~~Force (void) [virtual]`

[Force](#) destructor.

Definition at line 58 of file force.cc.

8.10.2.3 `jeod::Force::Force (const Force &) [private]`

Not implemented.

8.10.3 Member Function Documentation

8.10.3.1 `Force& jeod::Force::operator= (const Force &) [private]`

Not implemented.

8.10.3.2 `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>index</code>	Index number
-----------------	--------------------	--------------

Definition at line 42 of file force_inline.hh.

References [force](#).

8.10.3.3 `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><i>index</i></code>	Index number
-----------------	---------------------------	--------------

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

References `force`.

8.10.4 Field Documentation

8.10.4.1 `bool jeod::Force::active`

Is this force active?

`trick_units(-)`

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

8.10.4.2 `double jeod::Force::force[3]`

[Force](#) vector.

`trick_units(N)`

Definition at line 69 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](#)

8.11 `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)

8.11.1 Detailed Description

Contains translational and rotational second derivatives.

Definition at line 38 of file `frame_derivs.hh`.

8.11.2 Constructor & Destructor Documentation

8.11.2.1 `jeod::FrameDerivs::FrameDerivs (void)`

Default constructor.

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

References `non_grav_accel`, `rot_accel`, and `trans_accel`.

8.11.3 Field Documentation

8.11.3.1 `double jeod::FrameDerivs::non_grav_accel[3]`

Non-gravitational acceleration.

`trick_units(m/s2)`

Definition at line 48 of file `frame_derivs.hh`.

Referenced by `jeod::StructureIntegratedDynBody::collect_forces_and_torques()`, `jeod::DynBody::collect_forces_and_torques()`, `jeod::StructureIntegratedDynBody::compute_vehicle_point_derivatives()`, `jeod::DynBody::compute_vehicle_point_derivatives()`, `FrameDerivs()`, and `jeod::StructureIntegratedDynBody::solve_constraints()`.

8.11.3.2 `Quaternion jeod::FrameDerivs::Qdot_parent_this`

Time derivative of `Q_parent_this`.

`trick_units(1/s)`

Definition at line 58 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()`.

8.11.3.3 `double jeod::FrameDerivs::rot_accel[3]`

Total rotational acceleration (expressed in body frame)

`trick_units(radian/s2)`

Definition at line 63 of file `frame_derivs.hh`.

Referenced by `jeod::StructureIntegratedDynBody::collect_forces_and_torques()`, `jeod::DynBody::collect_forces_and_torques()`, `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()`.

8.11.3.4 double jeod::FrameDerivs::trans_accel[3]

Total acceleration.

trick_units(m/s2)

Definition at line 53 of file frame_derivs.hh.

Referenced by jeod::StructureIntegratedDynBody::collect_forces_and_torques(), jeod::DynBody::collect_forces_and_torques(), 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:

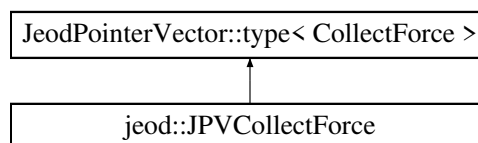
- [frame_derivs.hh](#)
- [aux_classes.cc](#)

8.12 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

- virtual void [perform_cleanup_action](#) (const std::string &)
Free stale data, typically following a restore from checkpoint.

8.12.1 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 69 of file body_force_collect.hh.

8.12.2 Member Function Documentation

8.12.2.1 virtual void jeod::JPVCollectForce::perform_cleanup_action (const std::string &) [inline], [virtual]

Free stale data, typically following a restore from checkpoint.

Definition at line 76 of file body_force_collect.hh.

References jeod::release_vector().

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

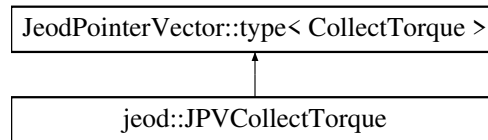
- [body_force_collect.hh](#)

8.13 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

- virtual void [perform_cleanup_action](#) (const std::string &)
Free stale data, typically following a restore from checkpoint.

8.13.1 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 94 of file `body_force_collect.hh`.

8.13.2 Member Function Documentation

8.13.2.1 virtual void `jeod::JPVCollectTorque::perform_cleanup_action (const std::string &)` `[inline]`, `[virtual]`

Free stale data, typically following a restore from checkpoint.

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

References `jeod::release_vector()`.

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

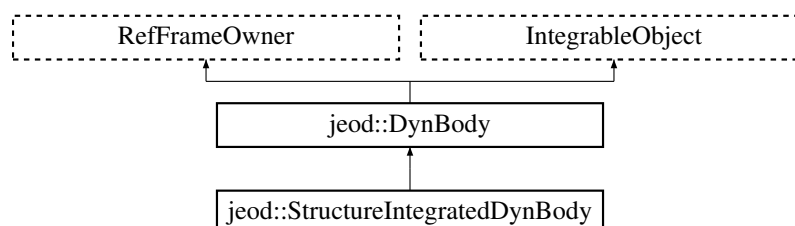
- [body_force_collect.hh](#)

8.14 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.
- virtual [~StructureIntegratedDynBody](#) ()
Destructor.
- virtual void [collect_forces_and_torques](#) ()
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.
- virtual void [compute_vehicle_point_derivatives](#) (const BodyRefFrame &frame, FrameDerivs &derivs)
Compute the state derivatives at a vehicle point.
- virtual bool [detach](#) (DynBody &other_body)
Break the logical connectivity between parent and child.

Data Fields

- [BodyWrenchCollect effector_wrench_collection](#)
Collection of effector wrenches.

Protected Member Functions

- virtual void [attach_update_properties](#) (double offset_pstr_cstr_pstr[3], double T_pstr_cstr[3][3], DynBody &child)
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.
- virtual er7_utils::IntegratorResult [trans_integ](#) (double dyn_dt, unsigned int target_stage)
Integrate the translational state of a [StructureIntegratedDynBody](#).
- virtual er7_utils::IntegratorResult [rot_integ](#) (double dyn_dt, unsigned int target_stage)
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.

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.

Friends

- class [InputProcessor](#)
- class [DynBodyConstraintsSolver](#)
- void [init_attrjeod__StructureIntegratedDynBody](#) ()

8.14.1 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 55 of file `structure_integrated_dyn_body.hh`.

8.14.2 Constructor & Destructor Documentation

8.14.2.1 `jeod::StructureIntegratedDynBody::StructureIntegratedDynBody ()`

Constructor.

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

References `jeod::DynBody::integrated_frame`, and `jeod::DynBody::structure`.

8.14.2.2 `jeod::StructureIntegratedDynBody::~StructureIntegratedDynBody () [virtual]`

Destructor.

Definition at line 59 of file `structure_integrated_dyn_body.cc`.

8.14.3 Member Function Documentation

8.14.3.1 `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 124 of file `structure_integrated_dyn_body_solve.cc`.

References `constraints_solver`, and `jeod::DynBodyMessages::invalid_body`.

8.14.3.2 `void jeod::StructureIntegratedDynBody::attach_update_properties (double offset_pstr_cstr_pstr[3], double T_pstr_cstr[3][3], DynBody & child) [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`.

8.14.3.3 `void jeod::StructureIntegratedDynBody::collect_forces_and_torques () [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()`, `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`.

8.14.3.4 `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_enviro_n_forc, jeod::BodyForceCollect::collect_enviro_n_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::enviro_n_forc, jeod::BodyForceCollect::enviro_n_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().

8.14.3.5 void jeod::StructureIntegratedDynBody::compute_vehicle_point_derivatives (const BodyRefFrame & frame, FrameDerivs & derivs) [virtual]

Compute the state derivatives at a vehicle point.

Parameters

<i>frame</i>	The vehicle point, as a BodyRefFrame , 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.

8.14.3.6 bool jeod::StructureIntegratedDynBody::detach (DynBody & other_body) [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 69 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().

8.14.3.7 const VehicleProperties& jeod::StructureIntegratedDynBody::get_vehicle_properties () const [inline], [protected]

Get the vehicle properties as a const reference.

Definition at line 233 of file structure_integrated_dyn_body.hh.

References vehicle_properties.

8.14.3.8 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::enviro_n_

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().

8.14.3.9 `er7_utils::IntegratorResult jeod::StructureIntegratedDynBody::rot_integ (double dyn_dt, unsigned int target_stage)`
`[protected], [virtual]`

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

Parameters

<code>in</code>	<code><i>dyn_dt</i></code>	Dynamic time step, in dynamic time seconds.
<code>in</code>	<code><i>target_stage</i></code>	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 53 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`.

8.14.3.10 `void jeod::StructureIntegratedDynBody::set_solver (DynBodyConstraintsSolver & solver_in)`

Set the solver to be used to solve constraints.

Definition at line 107 of file `structure_integrated_dyn_body_solve.cc`.

References `constraints_solver`, `jeod::DynBodyMessages::invalid_body`, and `jeod::DynBody::name`.

8.14.3.11 `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 140 of file `structure_integrated_dyn_body_solve.cc`.

References `jeod::VehicleNonGravState::accel_struct`, `jeod::DynBody::collect`, `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`.

8.14.3.12 `er7_utils::IntegratorResult jeod::StructureIntegratedDynBody::trans_integ (double dyn_dt, unsigned int target_stage)`
`[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 37 of file `structure_integrated_dyn_body_integration.cc`.

References `struct_derivs`, `jeod::DynBody::structure`, `jeod::FrameDerivs::trans_accel`, and `jeod::DynBody::trans_integrator`.

8.14.4 Friends And Related Function Documentation

8.14.4.1 friend class DynBodyConstraintsSolver [friend]

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

8.14.4.2 void init_attrjeod__StructureIntegratedDynBody () [friend]

8.14.4.3 friend class InputProcessor [friend]

Definition at line 57 of file `structure_integrated_dyn_body.hh`.

8.14.5 Field Documentation

8.14.5.1 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 164 of file `structure_integrated_dyn_body.hh`.

Referenced by `add_constraint()`, `attach_update_properties()`, `detach()`, `set_solver()`, and `solve_constraints()`.

8.14.5.2 Wrench jeod::StructureIntegratedDynBody::effector_wrench [protected]

[Wrench](#) into which the effector wrenches are accumulated.

`trick_units(-)`

Definition at line 169 of file `structure_integrated_dyn_body.hh`.

Referenced by `collect_forces_and_torques()`, `collect_local_forces_and_torques()`, and `PropagateForcesAndTorques()`.

8.14.5.3 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 82 of file `structure_integrated_dyn_body.hh`.

Referenced by `collect_local_forces_and_torques()`.

8.14.5.4 `double jeod::StructureIntegratedDynBody::inertial_accel_inrtl[3]` `[protected]`

Inertial-referenced inertial acceleration at the structure frame origin.

`trick_units(m/s2)`

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

8.14.5.5 `double jeod::StructureIntegratedDynBody::inertial_accel_struct[3]` `[protected]`

Structure-referenced inertial acceleration at the structure frame origin.

`trick_units(m/s2)`

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

8.14.5.6 `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/s2)`

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

8.14.5.7 `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/s2)`

Definition at line 196 of file `structure_integrated_dyn_body.hh`.

8.14.5.8 `VehicleNonGravState jeod::StructureIntegratedDynBody::non_grav_state` `[protected]`

Rotational and translational behaviors, for the constraints solver.

`trick_units(-)`

Definition at line 184 of file `structure_integrated_dyn_body.hh`.

Referenced by `solve_constraints()`.

8.14.5.9 `FrameDerivs jeod::StructureIntegratedDynBody::struct_derivs` `[protected]`

Translational/rotational accelerations of the structural frame.

`trick_units(-)`

Definition at line 174 of file structure_integrated_dyn_body.hh.

Referenced by collect_forces_and_torques(), compute_vehicle_point_derivatives(), rot_integ(), solve_constraints(), and trans_integ().

8.14.5.10 VehicleProperties jeod::StructureIntegratedDynBody::vehicle_properties [protected]

Various properties of the vehicle, for the constraints solver.

trick_units(–)

Definition at line 179 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](#)

8.15 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.

8.15.1 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 48 of file torque.hh.

8.15.2 Constructor & Destructor Documentation

8.15.2.1 `jeod::Torque::Torque (void)`

[Torque](#) default constructor.

Definition at line 46 of file torque.cc.

References [torque](#).

8.15.2.2 `jeod::Torque::~~Torque (void) [virtual]`

[Torque](#) destructor.

Definition at line 58 of file torque.cc.

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

Not implemented.

8.15.3 Member Function Documentation

8.15.3.1 `Torque& jeod::Torque::operator= (const Torque &) [private]`

Not implemented.

8.15.3.2 `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

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

Definition at line 42 of file torque_inline.hh.

References [torque](#).

8.15.3.3 `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

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

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

References [torque](#).

8.15.4 Field Documentation

8.15.4.1 `bool jeod::Torque::active`

Is this torque active?

`trick_units(-)`

Definition at line 63 of file `torque.hh`.

8.15.4.2 `double jeod::Torque::torque[3]`

[Torque](#) vector.

`trick_units(N*m)`

Definition at line 67 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](#)

8.16 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` ()

8.16.1 Detailed Description

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

Definition at line 31 of file `vehicle_non_grav_state.hh`.

8.16.2 Friends And Related Function Documentation

8.16.2.1 void `init_attrjeod__VehicleNonGravState` () [`friend`]

8.16.2.2 friend class `InputProcessor` [`friend`]

Definition at line 33 of file `vehicle_non_grav_state.hh`.

8.16.3 Field Documentation

8.16.3.1 double `jeod::VehicleNonGravState::accel_struct`[3]

Vehicle non-gravitational translational acceleration at the center of mass, in root body structural frame coordinates.

trick_units(m/s^2)

Definition at line 70 of file `vehicle_non_grav_state.hh`.

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

8.16.3.2 double `jeod::VehicleNonGravState::inertial_torque_struct`[3]

Vehicle inertial torque ($w \times lw$) in root body structural coordinates.

trick_units($N \cdot m$)

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

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

8.16.3.3 double `jeod::VehicleNonGravState::omega_body`[3]

Vehicle angular velocity with respect to inertial, in root body body frame coordinates.

trick_units($1/s$)

Definition at line 41 of file `vehicle_non_grav_state.hh`.

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

8.16.3.4 double jeod::VehicleNonGravState::omega_dot_body[3]

Vehicle angular acceleration with respect to inertial, in root body body frame coordinates.

trick_units(1/s²)

Definition at line 53 of file vehicle_non_grav_state.hh.

Referenced by jeod::StructureIntegratedDynBody::solve_constraints().

8.16.3.5 double jeod::VehicleNonGravState::omega_dot_struct[3]

Vehicle angular acceleration with respect to inertial, in root body structural frame coordinates.

trick_units(1/s²)

Definition at line 59 of file vehicle_non_grav_state.hh.

Referenced by jeod::StructureIntegratedDynBody::solve_constraints().

8.16.3.6 double jeod::VehicleNonGravState::omega_struct[3]

Vehicle angular velocity with respect to inertial, in root body structural frame coordinates.

trick_units(1/s)

Definition at line 47 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](#)

8.17 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](#) ()

8.17.1 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 37 of file vehicle_properties.hh.

8.17.2 Constructor & Destructor Documentation

8.17.2.1 `jeod::VehicleProperties::VehicleProperties ()` `[inline]`

Default constructor, for use by Trick only.

Definition at line 53 of file vehicle_properties.hh.

8.17.2.2 `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 85 of file vehicle_properties.hh.

8.17.3 Member Function Documentation

8.17.3.1 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 154 of file vehicle_properties.hh.

References inertia.

8.17.3.2 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 180 of file vehicle_properties.hh.

References inverse_inertia.

8.17.3.3 double jeod::VehicleProperties::get_inverse_mass () const [inline]

Returns

The multiplicative inverse of the vehicle's mass.

Definition at line 171 of file vehicle_properties.hh.

References inverse_mass.

8.17.3.4 double jeod::VehicleProperties::get_mass () const [inline]

Returns

The vehicle mass.

Definition at line 135 of file vehicle_properties.hh.

References mass.

8.17.3.5 `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 118 of file vehicle_properties.hh.

References parent_to_structure_offset.

8.17.3.6 `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 127 of file vehicle_properties.hh.

References parent_to_structure_transform.

8.17.3.7 `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 145 of file vehicle_properties.hh.

References structure_to_body_offset.

8.17.3.8 `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 163 of file vehicle_properties.hh.

References structure_to_body_transform.

8.17.4 Friends And Related Function Documentation

8.17.4.1 `void init_attrjeod__VehicleProperties ()` `[friend]`

8.17.4.2 `friend class InputProcessor` `[friend]`

Definition at line 43 of file vehicle_properties.hh.

8.17.5 Field Documentation

8.17.5.1 SolverTypes::Matrix3x3PointerT jeod::VehicleProperties::inertia [private]

Pointer to the vehicle's composite_properties.inertia tensor.

trick_units($m^2 \cdot kg$)

Definition at line 212 of file vehicle_properties.hh.

Referenced by get_inertia().

8.17.5.2 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 227 of file vehicle_properties.hh.

Referenced by get_inverse_inertia().

8.17.5.3 double* jeod::VehicleProperties::inverse_mass [private]

Pointer to the vehicle's inverse_mass member.

trick_units($1/kg$)

Definition at line 222 of file vehicle_properties.hh.

Referenced by get_inverse_mass().

8.17.5.4 double* jeod::VehicleProperties::mass [private]

Pointer to the vehicle's composite_properties.mass member.

trick_units(kg)

Definition at line 202 of file vehicle_properties.hh.

Referenced by get_mass().

8.17.5.5 SolverTypes::Vector3PointerT jeod::VehicleProperties::parent_to_structure_offset [private]

Pointer to the vehicle's structure_point.position vector.

trick_units(m)

Definition at line 192 of file vehicle_properties.hh.

Referenced by get_parent_to_structure_offset().

8.17.5.6 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 197 of file vehicle_properties.hh.

Referenced by get_parent_to_structure_transform().

8.17.5.7 SolverTypes::Vector3PointerT jeod::VehicleProperties::structure_to_body_offset [private]

Pointer to the vehicle's composite_properties.position vector.

trick_units(m)

Definition at line 207 of file vehicle_properties.hh.

Referenced by get_structure_to_body_offset().

8.17.5.8 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 217 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](#)

8.18 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](#) ()

8.18.1 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 45 of file wrench.hh.

8.18.2 Constructor & Destructor Documentation

8.18.2.1 `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 61 of file wrench.hh.

References [force](#), [point](#), and [torque](#).

8.18.2.2 `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 78 of file wrench.hh.

References [force](#), [point](#), and [torque](#).

8.18.2.3 `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 97 of file wrench.hh.

References force, point, and torque.

8.18.2.4 virtual jeod::Wrench::~~Wrench () [virtual],[default]

Destructor.

8.18.2.5 jeod::Wrench::Wrench (const Wrench &) [default]

Copy constructor.

8.18.2.6 jeod::Wrench::Wrench (Wrench &&) [default]

Move constructor.

8.18.3 Member Function Documentation

8.18.3.1 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 336 of file wrench.hh.

References reset_force_and_torque().

Referenced by jeod::BodyWrenchCollect::accumulate(), and accumulate().

8.18.3.2 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 354 of file wrench.hh.

References accumulate(), and set_point().

8.18.3.3 void jeod::Wrench::activate () [inline]

Mark this wrench as active.

Definition at line 162 of file wrench.hh.

References active.

8.18.3.4 `void jeod::Wrench::deactivate () [inline]`

Mark this wrench as inactive.

Definition at line 171 of file `wrench.hh`.

References active.

8.18.3.5 `const double* jeod::Wrench::get_force () const [inline]`

Const getter of the force vector.

Definition at line 316 of file `wrench.hh`.

References force.

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

8.18.3.6 `const double* jeod::Wrench::get_point () const [inline]`

Const getter of the point vector.

Definition at line 325 of file `wrench.hh`.

References point.

8.18.3.7 `const double* jeod::Wrench::get_torque () const [inline]`

Const getter of the torque vector.

Definition at line 307 of file `wrench.hh`.

References torque.

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

8.18.3.8 `bool jeod::Wrench::is_active (void) const [inline]`

Is this wrench active?

Definition at line 180 of file `wrench.hh`.

References active.

8.18.3.9 `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 144 of file `wrench.hh`.

References active, force, point, and torque.

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

Copy assignment operator.

8.18.3.11 `Wrench& jeod::Wrench::operator= (Wrench &&) [default]`

Move assignment operator.

8.18.3.12 `void jeod::Wrench::reset_force () [inline]`

Set the force to zero.

The torque and point remain unaltered.

Definition at line 208 of file wrench.hh.

References force.

8.18.3.13 `void jeod::Wrench::reset_force_and_torque () [inline]`

Set the force and torque to zero.

The point remains unaltered.

Definition at line 189 of file wrench.hh.

References force, and torque.

Referenced by accumulate(), and jeod::StructureIntegratedDynBody::collect_local_forces_and_torques().

8.18.3.14 `void jeod::Wrench::reset_point () [inline]`

Set the point to zero.

The torque and force remain unaltered.

Definition at line 217 of file wrench.hh.

References point.

8.18.3.15 `void jeod::Wrench::reset_torque () [inline]`

Set the torque to zero.

The force and point remain unaltered.

Definition at line 199 of file wrench.hh.

References torque.

8.18.3.16 `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 298 of file wrench.hh.

References force.

8.18.3.17 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 288 of file wrench.hh.

References torque.

8.18.3.18 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 229 of file wrench.hh.

References force, point, and torque.

8.18.3.19 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 254 of file wrench.hh.

References force.

8.18.3.20 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 264 of file wrench.hh.

References force, and point.

8.18.3.21 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 277 of file wrench.hh.

References point.

Referenced by jeod::BodyWrenchCollect::accumulate(), and accumulate().

8.18.3.22 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 244 of file wrench.hh.

References torque.

8.18.3.23 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 385 of file wrench.hh.

References force, point, and torque.

Referenced by jeod::StructureIntegratedDynBody::PropagateForcesAndTorques().

8.18.3.24 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 368 of file wrench.hh.

References active, force, point, and torque.

Referenced by jeod::StructureIntegratedDynBody::collect_forces_and_torques().

8.18.4 Friends And Related Function Documentation

8.18.4.1 void init_attrjeod__Wrench () [friend]

8.18.4.2 friend class InputProcessor [friend]

Definition at line 47 of file wrench.hh.

8.18.5 Field Documentation

8.18.5.1 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 430 of file wrench.hh.

Referenced by activate(), deactivate(), is_active(), operator+=(), and transform_to_point().

8.18.5.2 `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 419 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()`.

8.18.5.3 `double jeod::Wrench::point[3]` [private]

The structural coordinates of the point at which the force is applied.

trick_units(m)

Definition at line 424 of file wrench.hh.

Referenced by `get_point()`, `operator+=()`, `reset_point()`, `set()`, `set_force()`, `set_point()`, `transform_to_parent()`, `transform_to_point()`, and `Wrench()`.

8.18.5.4 `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 413 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.

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 <cstdlib>
#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 "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"
```

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_initialize_model.cc File Reference

Define DynBody::initialize_model.

```
#include <cstdint>
#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.

9.13.1 Detailed Description

Define DynBody::initialize_model.

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

9.14 dyn_body_integration.cc File Reference

Define methods for frame switching.

```
#include <cstdint>
#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.

9.14.1 Detailed Description

Define methods for frame switching.

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

9.15 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/"

9.15.1 Detailed Description

Implement the class De4xxMessages.

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

9.16 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.

9.16.1 Detailed Description

Define the class DynBodyMessages.

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

9.17 dyn_body_propagate_state.cc File Reference

Define DynBody state propagation / update methods.

```
#include <cstddef>
#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.

9.17.1 Detailed Description

Define DynBody state propagation / update methods.

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

9.18 dyn_body_set_state.cc File Reference

Define methods related to setting aspects of a vehicle's state.

```
#include <cstdint>
#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.

9.18.1 Detailed Description

Define methods related to setting aspects of a vehicle's state.

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

9.19 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.

9.19.1 Detailed Description

Define methods that support vehicle points.

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

9.20 force.cc File Reference

Define force model member functions.

```
#include <stddef>
#include "utils/math/include/vector3.hh"
#include "utils/memory/include/jeod_alloc.hh"
#include "../include/force.hh"
```

Namespaces

- [jeod](#)

Namespace jeod.

9.20.1 Detailed Description

Define force model member functions.

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

9.21 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.

9.21.1 Detailed Description

Define the JEOD force model.

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

9.22 force_inline.hh File Reference

Inline functions for the JEOD force model.

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

Namespaces

- [jeod](#)

Namespace jeod.

9.22.1 Detailed Description

Inline functions for the JEOD force model.

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

9.23 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.

9.23.1 Detailed Description

Define the FrameDerivs class.

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

9.24 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 <cstdint>
```

Namespaces

- [jeod](#)
Namespace jeod.

9.24.1 Detailed Description

Define base member functions for StructureIntegratedDynBody.

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

9.25 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.

9.25.1 Detailed Description

Define the class StructureIntegratedDynBody, which integrates a DynBody object's structural state.

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

9.26 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 <cstdlib>
```

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.

9.26.1 Detailed Description

Define StructureIntegratedDynBody methods related to force and torque accumulation and propagation.

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

9.27 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 <cstdint>
```

Namespaces

- [jeod](#)

Namespace jeod.

9.27.1 Detailed Description

Define StructureIntegratedDynBody member functions related to state integration.

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

9.28 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 <stdio>
```

Namespaces

- [jeod](#)

Namespace jeod.

9.28.1 Detailed Description

Define StructureIntegratedDynBody::compute_vehicle_point_derivatives.

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

9.29 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.

9.29.1 Detailed Description

Define StructureIntegratedDynBody methods related to force and torque accumulation and propagation.

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

9.30 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.

9.30.1 Detailed Description

Define torque model member functions.

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

9.31 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.

9.31.1 Detailed Description

Define the JEOD torque model.

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

9.32 torque_inline.hh File Reference

Define the JEOD torque model.

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

Namespaces

- [jeod](#)
Namespace jeod.

9.32.1 Detailed Description

Define the JEOD torque model.

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

9.33 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.

9.33.1 Detailed Description

Define the class VehicleNonGravState.

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

9.34 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.

9.34.1 Detailed Description

Define the class VehicleProperties.

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

9.35 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.

9.35.1 Detailed Description

Define the class Wrench.

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

Index

- ~BodyForceCollect
 - jeod::BodyForceCollect, [20](#)
- ~BodyRefFrame
 - jeod::BodyRefFrame, [25](#)
- ~BodyWrenchCollect
 - jeod::BodyWrenchCollect, [26](#)
- ~CInterfaceForce
 - jeod::CInterfaceForce, [29](#)
- ~CInterfaceTorque
 - jeod::CInterfaceTorque, [30](#)
- ~CollectForce
 - jeod::CollectForce, [34](#)
- ~CollectTorque
 - jeod::CollectTorque, [39](#)
- ~DynBody
 - jeod::DynBody, [48](#)
- ~Force
 - jeod::Force, [81](#)
- ~StructureIntegratedDynBody
 - jeod::StructureIntegratedDynBody, [88](#)
- ~Torque
 - jeod::Torque, [95](#)
- ~Wrench
 - jeod::Wrench, [105](#)
- accel_struct
 - jeod::VehicleNonGravState, [97](#)
- accumulate
 - jeod::BodyWrenchCollect, [27](#)
 - jeod::Wrench, [106](#)
- accumulate_forces
 - jeod, [16](#)
- accumulate_torques
 - jeod, [17](#)
- activate
 - jeod::DynBody, [48](#)
 - jeod::Wrench, [106](#)
- active
 - jeod::CollectForce, [37](#)
 - jeod::CollectTorque, [42](#)
 - jeod::Force, [82](#)
 - jeod::Torque, [96](#)
 - jeod::Wrench, [110](#)
- add_constraint
 - jeod::StructureIntegratedDynBody, [88](#)
- add_control
 - jeod::DynBody, [48](#)
- add_integrable_object
 - jeod::DynBody, [48](#)
- add_mass_body
 - jeod::DynBody, [49](#)
- add_mass_body_frames
 - jeod::DynBody, [49](#)
- add_mass_body_validate
 - jeod::DynBody, [49](#)
- add_mass_point
 - jeod::DynBody, [50](#)
- associated_integrable_objects
 - jeod::DynBody, [71](#)
- attach
 - jeod::DynBody, [50](#)
- attach_establish_links
 - jeod::DynBody, [50](#)
- attach_update_properties
 - jeod::DynBody, [51](#)
 - jeod::StructureIntegratedDynBody, [88](#)
- attach_validate_child
 - jeod::DynBody, [51](#)
- attach_validate_parent
 - jeod::DynBody, [52](#)
- attitude_source
 - jeod::DynBody, [71](#)
- autoupdate_vehicle_points
 - jeod::DynBody, [71](#)
- aux_classes.cc, [113](#)
- body_force_collect.hh, [113](#)
- body_ref_frame.hh, [114](#)
- body_wrench_collect.cc, [115](#)
- body_wrench_collect.hh, [115](#)
- BodyForceCollect
 - jeod::BodyForceCollect, [20](#)
- BodyRefFrame
 - jeod::BodyRefFrame, [25](#)
- BodyWrenchCollect
 - jeod::BodyWrenchCollect, [26](#), [27](#)
- CInterfaceForce
 - jeod::CInterfaceForce, [28](#), [29](#)
- CInterfaceTorque
 - jeod::CInterfaceTorque, [30](#)
- check_frame_ownership
 - jeod, [17](#)
- class_declarations.hh, [115](#)
- clear_integrable_objects
 - jeod::DynBody, [52](#)
- collect
 - jeod::DynBody, [71](#)
- collect_effector_forc
 - jeod::BodyForceCollect, [21](#)

- collect_effector_torq
 - jeod::BodyForceCollect, [21](#)
- collect_envir_forc
 - jeod::BodyForceCollect, [21](#)
- collect_envir_torq
 - jeod::BodyForceCollect, [21](#)
- collect_forces_and_torques
 - jeod::DynBody, [52](#)
 - jeod::StructureIntegratedDynBody, [89](#)
- collect_local_forces_and_torques
 - jeod::StructureIntegratedDynBody, [89](#)
- collect_no_xmit_forc
 - jeod::BodyForceCollect, [21](#)
- collect_no_xmit_torq
 - jeod::BodyForceCollect, [21](#)
- collect_wrench
 - jeod::BodyWrenchCollect, [27](#)
- CollectForce
 - jeod::CollectForce, [32, 34](#)
- CollectTorque
 - jeod::CollectTorque, [39](#)
- composite_body
 - jeod::DynBody, [71](#)
- compute_derived_state_forward
 - jeod::DynBody, [53](#)
- compute_derived_state_reverse
 - jeod::DynBody, [53](#)
- compute_ref_point_transform
 - jeod::DynBody, [53](#)
- compute_state_elements_forward
 - jeod::DynBody, [54](#)
- compute_state_elements_reverse
 - jeod::DynBody, [54](#)
- compute_vehicle_point_derivatives
 - jeod::DynBody, [54](#)
 - jeod::StructureIntegratedDynBody, [89](#)
- compute_vehicle_point_states
 - jeod::DynBody, [55](#)
- constraints_solver
 - jeod::StructureIntegratedDynBody, [92](#)
- core_body
 - jeod::DynBody, [72](#)
- create
 - jeod::CollectForce, [34, 35](#)
 - jeod::CollectTorque, [39–41](#)
- create_body_integrators
 - jeod::DynBody, [55](#)
- create_integrators
 - jeod::DynBody, [55](#)
- deactivate
 - jeod::DynBody, [56](#)
 - jeod::Wrench, [106](#)
- derivs
 - jeod::DynBody, [72](#)
- destroy_integrators
 - jeod::DynBody, [56](#)
- detach
 - jeod::DynBody, [56](#)
- jeod::StructureIntegratedDynBody, [90](#)
- detach_mass_body_frames
 - jeod::DynBody, [57](#)
- detach_mass_internal
 - jeod::DynBody, [57](#)
- dyn_body.cc, [116](#)
- dyn_body.hh, [116](#)
- dyn_body_attach.cc, [117](#)
- dyn_body_collect.cc, [117](#)
- dyn_body_detach.cc, [118](#)
- dyn_body_find_body_frame.cc, [119](#)
- dyn_body_initialize_model.cc, [119](#)
- dyn_body_integration.cc, [119](#)
- dyn_body_messages.cc, [120](#)
- dyn_body_messages.hh, [121](#)
- dyn_body_propagate_state.cc, [121](#)
- dyn_body_set_state.cc, [121](#)
- dyn_body_vehicle_point.cc, [122](#)
- dyn_children
 - jeod::DynBody, [72](#)
- dyn_manager
 - jeod::DynBody, [72](#)
- dyn_parent
 - jeod::DynBody, [72](#)
- DynBody, [13](#)
 - jeod::DynBody, [48](#)
 - PATH, [14](#)
- DynBodyConstraintsSolver
 - jeod::StructureIntegratedDynBody, [92](#)
- DynBodyMessages
 - jeod::DynBodyMessages, [78](#)
- Dynamics, [12](#)
- effector_forc
 - jeod::BodyForceCollect, [22](#)
- effector_torq
 - jeod::BodyForceCollect, [22](#)
- effector_wrench
 - jeod::StructureIntegratedDynBody, [92](#)
- effector_wrench_collection
 - jeod::StructureIntegratedDynBody, [92](#)
- envir_forc
 - jeod::BodyForceCollect, [22](#)
- envir_torq
 - jeod::BodyForceCollect, [22](#)
- extern_forc_intrl
 - jeod::BodyForceCollect, [22](#)
- extern_forc_struct
 - jeod::BodyForceCollect, [23](#)
- extern_torq_body
 - jeod::BodyForceCollect, [23](#)
- extern_torq_struct
 - jeod::BodyForceCollect, [23](#)
- find_body_frame
 - jeod::DynBody, [57](#)
- find_vehicle_point
 - jeod::DynBody, [58](#)
- Force

- jeod::Force, [81](#), [82](#)
- force
 - jeod::CollectForce, [37](#)
 - jeod::Force, [82](#)
 - jeod::Wrench, [110](#)
- force.cc, [123](#)
- force.hh, [123](#)
- force_inline.hh, [124](#)
- frame_derivs.hh, [124](#)
- FrameDerivs
 - jeod::FrameDerivs, [83](#)
- get_dynamics_integration_group
 - jeod::DynBody, [58](#)
- get_force
 - jeod::Wrench, [106](#)
- get_inertia
 - jeod::VehicleProperties, [100](#)
- get_initialized_states
 - jeod::DynBody, [58](#)
- get_integrable_objects
 - jeod::DynBody, [59](#)
- get_inverse_inertia
 - jeod::VehicleProperties, [100](#)
- get_inverse_mass
 - jeod::VehicleProperties, [100](#)
- get_mass
 - jeod::VehicleProperties, [100](#)
- get_parent_body
 - jeod::DynBody, [59](#)
- get_parent_body_internal
 - jeod::DynBody, [59](#)
- get_parent_to_structure_offset
 - jeod::VehicleProperties, [100](#)
- get_parent_to_structure_transform
 - jeod::VehicleProperties, [100](#)
- get_point
 - jeod::Wrench, [106](#)
- get_root_body
 - jeod::DynBody, [59](#)
- get_root_body_internal
 - jeod::DynBody, [59](#)
- get_structure_to_body_offset
 - jeod::VehicleProperties, [101](#)
- get_structure_to_body_transform
 - jeod::VehicleProperties, [101](#)
- get_torque
 - jeod::Wrench, [107](#)
- get_vehicle_properties
 - jeod::StructureIntegratedDynBody, [90](#)
- grav_interaction
 - jeod::DynBody, [73](#)
- inertia
 - jeod::VehicleProperties, [101](#)
- inertial_accel_inrtl
 - jeod::StructureIntegratedDynBody, [92](#)
- inertial_accel_struct
 - jeod::StructureIntegratedDynBody, [93](#)
- inertial_accel_struct_omega
 - jeod::StructureIntegratedDynBody, [93](#)
- inertial_accel_struct_omega_dot
 - jeod::StructureIntegratedDynBody, [93](#)
- inertial_torq
 - jeod::BodyForceCollect, [23](#)
- inertial_torque_struct
 - jeod::VehicleNonGravState, [97](#)
- init_attrjeod__BodyRefFrame
 - jeod::BodyRefFrame, [25](#)
- init_attrjeod__DynBody
 - jeod::DynBody, [71](#)
- init_attrjeod__DynBodyMessages
 - jeod::DynBodyMessages, [78](#)
- init_attrjeod__StructureIntegratedDynBody
 - jeod::StructureIntegratedDynBody, [92](#)
- init_attrjeod__VehicleNonGravState
 - jeod::VehicleNonGravState, [97](#)
- init_attrjeod__VehicleProperties
 - jeod::VehicleProperties, [101](#)
- init_attrjeod__Wrench
 - jeod::Wrench, [110](#)
- initialize_controls
 - jeod::DynBody, [60](#)
- initialize_model
 - jeod::DynBody, [60](#)
- initialized_items
 - jeod::BodyRefFrame, [25](#)
- initialized_states
 - jeod::DynBody, [73](#)
- initialized_states_contains
 - jeod::DynBody, [60](#)
- InputProcessor
 - jeod::BodyRefFrame, [25](#)
 - jeod::DynBody, [71](#)
 - jeod::DynBodyMessages, [78](#)
 - jeod::StructureIntegratedDynBody, [92](#)
 - jeod::VehicleNonGravState, [97](#)
 - jeod::VehicleProperties, [101](#)
 - jeod::Wrench, [110](#)
- integ_frame
 - jeod::DynBody, [73](#)
- integ_frame_name
 - jeod::DynBody, [73](#)
- integ_results_merger
 - jeod::DynBody, [73](#)
- integrate
 - jeod::DynBody, [61](#)
- integrated_frame
 - jeod::DynBody, [74](#)
- internal_error
 - jeod::DynBodyMessages, [78](#)
- invalid_attachment
 - jeod::DynBodyMessages, [79](#)
- invalid_body
 - jeod::DynBodyMessages, [79](#)
- invalid_frame
 - jeod::DynBodyMessages, [79](#)

- invalid_group
 - jeod::DynBodyMessages, 79
- invalid_name
 - jeod::DynBodyMessages, 80
- invalid_technique
 - jeod::DynBodyMessages, 80
- inverse_inertia
 - jeod::VehicleProperties, 101
- inverse_mass
 - jeod::VehicleProperties, 102
- is_active
 - jeod::CollectForce, 36
 - jeod::CollectTorque, 41
 - jeod::Wrench, 107
- is_root_body
 - jeod::DynBody, 61
- jeod, 15
 - accumulate_forces, 16
 - accumulate_torques, 17
 - check_frame_ownership, 17
 - release_vector, 17
- jeod::BodyForceCollect, 19
 - ~BodyForceCollect, 20
 - BodyForceCollect, 20
 - collect_effector_forc, 21
 - collect_effector_torq, 21
 - collect_envron_forc, 21
 - collect_envron_torq, 21
 - collect_no_xmit_forc, 21
 - collect_no_xmit_torq, 21
 - effector_forc, 22
 - effector_torq, 22
 - envron_forc, 22
 - envron_torq, 22
 - extern_forc_inrtl, 22
 - extern_forc_struct, 23
 - extern_torq_body, 23
 - extern_torq_struct, 23
 - inertial_torq, 23
 - no_xmit_forc, 23
 - no_xmit_torq, 23
 - operator=, 21
- jeod::BodyRefFrame, 24
 - ~BodyRefFrame, 25
 - BodyRefFrame, 25
 - init_attrjeod__BodyRefFrame, 25
 - initialized_items, 25
 - InputProcessor, 25
 - mass_point, 25
 - operator=, 25
- jeod::BodyWrenchCollect, 26
 - ~BodyWrenchCollect, 26
 - accumulate, 27
 - BodyWrenchCollect, 26, 27
 - collect_wrench, 27
 - operator=, 27
- jeod::CInterfaceForce, 28
 - ~CInterfaceForce, 29
- CInterfaceForce, 28, 29
 - operator=, 29
- jeod::CInterfaceTorque, 29
 - ~CInterfaceTorque, 30
 - CInterfaceTorque, 30
 - operator=, 31
- jeod::CollectForce, 31
 - ~CollectForce, 34
 - active, 37
 - CollectForce, 32, 34
 - create, 34, 35
 - force, 37
 - is_active, 36
 - operator=, 36
- jeod::CollectTorque, 37
 - ~CollectTorque, 39
 - active, 42
 - CollectTorque, 39
 - create, 39–41
 - is_active, 41
 - operator=, 41
 - torque, 42
- jeod::DynBody, 42
 - ~DynBody, 48
 - activate, 48
 - add_control, 48
 - add_integrable_object, 48
 - add_mass_body, 49
 - add_mass_body_frames, 49
 - add_mass_body_validate, 49
 - add_mass_point, 50
 - associated_integrable_objects, 71
 - attach, 50
 - attach_establish_links, 50
 - attach_update_properties, 51
 - attach_validate_child, 51
 - attach_validate_parent, 52
 - attitude_source, 71
 - autoupdate_vehicle_points, 71
 - clear_integrable_objects, 52
 - collect, 71
 - collect_forces_and_torques, 52
 - composite_body, 71
 - compute_derived_state_forward, 53
 - compute_derived_state_reverse, 53
 - compute_ref_point_transform, 53
 - compute_state_elements_forward, 54
 - compute_state_elements_reverse, 54
 - compute_vehicle_point_derivatives, 54
 - compute_vehicle_point_states, 55
 - core_body, 72
 - create_body_integrators, 55
 - create_integrators, 55
 - deactivate, 56
 - derivs, 72
 - destroy_integrators, 56
 - detach, 56
 - detach_mass_body_frames, 57

- detach_mass_internal, 57
- dyn_children, 72
- dyn_manager, 72
- dyn_parent, 72
- DynBody, 48
- find_body_frame, 57
- find_vehicle_point, 58
- get_dynamics_integration_group, 58
- get_initialized_states, 58
- get_integrable_objects, 59
- get_parent_body, 59
- get_parent_body_internal, 59
- get_root_body, 59
- get_root_body_internal, 59
- grav_interaction, 73
- init_attrjeod__DynBody, 71
- initialize_controls, 60
- initialize_model, 60
- initialized_states, 73
- initialized_states_contains, 60
- InputProcessor, 71
- integ_frame, 73
- integ_frame_name, 73
- integ_results_merger, 73
- integrate, 61
- integrated_frame, 74
- is_root_body, 61
- mass, 74
- mass_children, 74
- migrate_integrable_objects, 61
- name, 74
- operator=, 61
- position_source, 74
- process_dynamic_attachment, 61
- propagate_state, 62
- propagate_state_from_composite, 62
- propagate_state_from_structure, 63
- rate_source, 75
- remove_integrable_object, 63
- remove_mass_body, 63
- reset_controls, 64
- reset_integrators, 64
- rot_integ, 64
- rot_integrator, 75
- rotation_integration, 75
- rotational_dynamics, 75
- set_attitude_left_quaternion, 64
- set_attitude_matrix, 65
- set_attitude_rate, 65
- set_attitude_right_quaternion, 65
- set_integ_frame, 66
- set_name, 66
- set_position, 66
- set_state, 68
- set_state_source, 68
- set_state_source_internal, 68
- set_velocity, 69
- sort_controls, 69
- structure, 75
- switch_integration_frames, 69
- three_dof, 76
- time_manager, 76
- trans_integ, 70
- trans_integrator, 76
- translational_dynamics, 76
- update_integrated_state, 70
- vehicle_points, 76
- velocity_source, 77
- jeod::DynBodyMessages, 77
 - DynBodyMessages, 78
 - init_attrjeod__DynBodyMessages, 78
 - InputProcessor, 78
 - internal_error, 78
 - invalid_attachment, 79
 - invalid_body, 79
 - invalid_frame, 79
 - invalid_group, 79
 - invalid_name, 80
 - invalid_technique, 80
 - not_dyn_body, 80
 - operator=, 78
- jeod::Force, 81
 - ~Force, 81
 - active, 82
 - Force, 81, 82
 - force, 82
 - operator=, 82
- jeod::FrameDerivs, 83
 - FrameDerivs, 83
 - non_grav_accel, 84
 - Qdot_parent_this, 84
 - rot_accel, 84
 - trans_accel, 84
- jeod::JPVCollectForce, 84
 - perform_cleanup_action, 85
- jeod::JPVCollectTorque, 85
 - perform_cleanup_action, 86
- jeod::StructureIntegratedDynBody, 86
 - ~StructureIntegratedDynBody, 88
 - add_constraint, 88
 - attach_update_properties, 88
 - collect_forces_and_torques, 89
 - collect_local_forces_and_torques, 89
 - compute_vehicle_point_derivatives, 89
 - constraints_solver, 92
 - detach, 90
 - DynBodyConstraintsSolver, 92
 - effector_wrench, 92
 - effector_wrench_collection, 92
 - get_vehicle_properties, 90
 - inertial_accel_inrtl, 92
 - inertial_accel_struct, 93
 - inertial_accel_struct_omega, 93
 - inertial_accel_struct_omega_dot, 93
 - init_attrjeod__StructureIntegratedDynBody, 92
 - InputProcessor, 92

- non_grav_state, 93
- PropagateForcesAndTorques, 90
- rot_integ, 90
- set_solver, 91
- solve_constraints, 91
- struct_derivs, 93
- StructureIntegratedDynBody, 88
- trans_integ, 91
- vehicle_properties, 93
- jeod::Torque, 94
 - ~Torque, 95
 - active, 96
 - operator=, 95
 - Torque, 95
 - torque, 96
- jeod::VehicleNonGravState, 96
 - accel_struct, 97
 - inertial_torque_struct, 97
 - init_attrjeod__VehicleNonGravState, 97
 - InputProcessor, 97
 - omega_body, 97
 - omega_dot_body, 97
 - omega_dot_struct, 97
 - omega_struct, 98
- jeod::VehicleProperties, 98
 - get_inertia, 100
 - get_inverse_inertia, 100
 - get_inverse_mass, 100
 - get_mass, 100
 - get_parent_to_structure_offset, 100
 - get_parent_to_structure_transform, 100
 - get_structure_to_body_offset, 101
 - get_structure_to_body_transform, 101
 - inertia, 101
 - init_attrjeod__VehicleProperties, 101
 - InputProcessor, 101
 - inverse_inertia, 101
 - inverse_mass, 102
 - mass, 102
 - parent_to_structure_offset, 102
 - parent_to_structure_transform, 102
 - structure_to_body_offset, 102
 - structure_to_body_transform, 102
 - VehicleProperties, 99
- jeod::Wrench, 103
 - ~Wrench, 105
 - accumulate, 106
 - activate, 106
 - active, 110
 - deactivate, 106
 - force, 110
 - get_force, 106
 - get_point, 106
 - get_torque, 107
 - init_attrjeod__Wrench, 110
 - InputProcessor, 110
 - is_active, 107
 - operator+=, 107
 - operator=, 107
 - point, 110
 - reset_force, 107
 - reset_force_and_torque, 107
 - reset_point, 108
 - reset_torque, 108
 - scale_force, 108
 - scale_torque, 108
 - set, 108
 - set_force, 109
 - set_point, 109
 - set_torque, 109
 - torque, 111
 - transform_to_parent, 109
 - transform_to_point, 110
 - Wrench, 105
- mass
 - jeod::DynBody, 74
 - jeod::VehicleProperties, 102
- mass_children
 - jeod::DynBody, 74
- mass_point
 - jeod::BodyRefFrame, 25
- migrate_integrable_objects
 - jeod::DynBody, 61
- Models, 11
- name
 - jeod::DynBody, 74
- no_xmit_forc
 - jeod::BodyForceCollect, 23
- no_xmit_torq
 - jeod::BodyForceCollect, 23
- non_grav_accel
 - jeod::FrameDerivs, 84
- non_grav_state
 - jeod::StructureIntegratedDynBody, 93
- not_dyn_body
 - jeod::DynBodyMessages, 80
- omega_body
 - jeod::VehicleNonGravState, 97
- omega_dot_body
 - jeod::VehicleNonGravState, 97
- omega_dot_struct
 - jeod::VehicleNonGravState, 97
- omega_struct
 - jeod::VehicleNonGravState, 98
- operator+=
 - jeod::Wrench, 107
- operator=
 - jeod::BodyForceCollect, 21
 - jeod::BodyRefFrame, 25
 - jeod::BodyWrenchCollect, 27
 - jeod::CInterfaceForce, 29
 - jeod::CInterfaceTorque, 31
 - jeod::CollectForce, 36
 - jeod::CollectTorque, 41

- jeod::DynBody, 61
- jeod::DynBodyMessages, 78
- jeod::Force, 82
- jeod::Torque, 95
- jeod::Wrench, 107
- PATH
 - DynBody, 14
- parent_to_structure_offset
 - jeod::VehicleProperties, 102
- parent_to_structure_transform
 - jeod::VehicleProperties, 102
- perform_cleanup_action
 - jeod::JPVCollectForce, 85
 - jeod::JPVCollectTorque, 86
- point
 - jeod::Wrench, 110
- position_source
 - jeod::DynBody, 74
- process_dynamic_attachment
 - jeod::DynBody, 61
- propagate_state
 - jeod::DynBody, 62
- propagate_state_from_composite
 - jeod::DynBody, 62
- propagate_state_from_structure
 - jeod::DynBody, 63
- PropagateForcesAndTorques
 - jeod::StructureIntegratedDynBody, 90
- Qdot_parent_this
 - jeod::FrameDerivs, 84
- rate_source
 - jeod::DynBody, 75
- release_vector
 - jeod, 17
- remove_integrable_object
 - jeod::DynBody, 63
- remove_mass_body
 - jeod::DynBody, 63
- reset_controls
 - jeod::DynBody, 64
- reset_force
 - jeod::Wrench, 107
- reset_force_and_torque
 - jeod::Wrench, 107
- reset_integrators
 - jeod::DynBody, 64
- reset_point
 - jeod::Wrench, 108
- reset_torque
 - jeod::Wrench, 108
- rot_accel
 - jeod::FrameDerivs, 84
- rot_integ
 - jeod::DynBody, 64
 - jeod::StructureIntegratedDynBody, 90
- rot_integrator
 - jeod::DynBody, 75
- rotation_integration
 - jeod::DynBody, 75
- rotational_dynamics
 - jeod::DynBody, 75
- scale_force
 - jeod::Wrench, 108
- scale_torque
 - jeod::Wrench, 108
- set
 - jeod::Wrench, 108
- set_attitude_left_quaternion
 - jeod::DynBody, 64
- set_attitude_matrix
 - jeod::DynBody, 65
- set_attitude_rate
 - jeod::DynBody, 65
- set_attitude_right_quaternion
 - jeod::DynBody, 65
- set_force
 - jeod::Wrench, 109
- set_integ_frame
 - jeod::DynBody, 66
- set_name
 - jeod::DynBody, 66
- set_point
 - jeod::Wrench, 109
- set_position
 - jeod::DynBody, 66
- set_solver
 - jeod::StructureIntegratedDynBody, 91
- set_state
 - jeod::DynBody, 68
- set_state_source
 - jeod::DynBody, 68
- set_state_source_internal
 - jeod::DynBody, 68
- set_torque
 - jeod::Wrench, 109
- set_velocity
 - jeod::DynBody, 69
- solve_constraints
 - jeod::StructureIntegratedDynBody, 91
- sort_controls
 - jeod::DynBody, 69
- struct_derivs
 - jeod::StructureIntegratedDynBody, 93
- structure
 - jeod::DynBody, 75
- structure_integrated_dyn_body.cc, 124
- structure_integrated_dyn_body.hh, 125
- structure_integrated_dyn_body_collect.cc, 125
- structure_integrated_dyn_body_integration.cc, 126
- structure_integrated_dyn_body_pt_accel.cc, 126
- structure_integrated_dyn_body_solve.cc, 127
- structure_to_body_offset
 - jeod::VehicleProperties, 102
- structure_to_body_transform

- jeod::VehicleProperties, [102](#)
- StructureIntegratedDynBody
 - jeod::StructureIntegratedDynBody, [88](#)
- switch_integration_frames
 - jeod::DynBody, [69](#)
- three_dof
 - jeod::DynBody, [76](#)
- time_manager
 - jeod::DynBody, [76](#)
- Torque
 - jeod::Torque, [95](#)
- torque
 - jeod::CollectTorque, [42](#)
 - jeod::Torque, [96](#)
 - jeod::Wrench, [111](#)
- torque.cc, [127](#)
- torque.hh, [128](#)
- torque_inline.hh, [128](#)
- trans_accel
 - jeod::FrameDerivs, [84](#)
- trans_integ
 - jeod::DynBody, [70](#)
 - jeod::StructureIntegratedDynBody, [91](#)
- trans_integrator
 - jeod::DynBody, [76](#)
- transform_to_parent
 - jeod::Wrench, [109](#)
- transform_to_point
 - jeod::Wrench, [110](#)
- translational_dynamics
 - jeod::DynBody, [76](#)
- update_integrated_state
 - jeod::DynBody, [70](#)
- vehicle_non_grav_state.hh, [128](#)
- vehicle_points
 - jeod::DynBody, [76](#)
- vehicle_properties
 - jeod::StructureIntegratedDynBody, [93](#)
- vehicle_properties.hh, [129](#)
- VehicleProperties
 - jeod::VehicleProperties, [99](#)
- velocity_source
 - jeod::DynBody, [77](#)
- Wrench
 - jeod::Wrench, [105](#)
- wrench.hh, [129](#)