

## WSNet : Wireless Network Simulator

Generated by Doxygen 1.7.6.1

Thu Feb 23 2012 11:25:34



# Contents

<b>1</b>	<b>Deprecated List</b>	<b>1</b>
<b>2</b>	<b>Directory Hierarchy</b>	<b>3</b>
2.1	Directories . . . . .	3
<b>3</b>	<b>Data Structure Index</b>	<b>5</b>
3.1	Data Structures . . . . .	5
<b>4</b>	<b>File Index</b>	<b>9</b>
4.1	File List . . . . .	9
<b>5</b>	<b>Directory Documentation</b>	<b>11</b>
5.1	include/ Directory Reference . . . . .	11
<b>6</b>	<b>Data Structure Documentation</b>	<b>13</b>
6.1	_angle Struct Reference . . . . .	13
6.1.1	Detailed Description . . . . .	13
6.1.2	Field Documentation . . . . .	13
6.1.2.1	xy . . . . .	13
6.1.2.2	z . . . . .	14
6.2	_antenna_methods Struct Reference . . . . .	14
6.2.1	Detailed Description . . . . .	14
6.2.2	Field Documentation . . . . .	14
6.2.2.1	cs . . . . .	14
6.2.2.2	gain_rx . . . . .	14
6.2.2.3	gain_tx . . . . .	14
6.2.2.4	get_angle . . . . .	15

6.2.2.5	<a href="#">get_loss</a>	15
6.2.2.6	<a href="#">rx</a>	15
6.2.2.7	<a href="#">set_angle</a>	15
6.3	<a href="#">_application_methods Struct Reference</a>	15
6.3.1	<a href="#">Detailed Description</a>	15
6.3.2	<a href="#">Field Documentation</a>	15
6.3.2.1	<a href="#">rx</a>	15
6.4	<a href="#">_array Struct Reference</a>	16
6.4.1	<a href="#">Detailed Description</a>	16
6.4.2	<a href="#">Field Documentation</a>	16
6.4.2.1	<a href="#">elts</a>	16
6.4.2.2	<a href="#">size</a>	16
6.5	<a href="#">_call Struct Reference</a>	16
6.5.1	<a href="#">Detailed Description</a>	17
6.5.2	<a href="#">Field Documentation</a>	17
6.5.2.1	<a href="#">entity</a>	17
6.5.2.2	<a href="#">from</a>	17
6.5.2.3	<a href="#">node</a>	17
6.6	<a href="#">_destination Struct Reference</a>	17
6.6.1	<a href="#">Detailed Description</a>	18
6.6.2	<a href="#">Field Documentation</a>	18
6.6.2.1	<a href="#">id</a>	18
6.6.2.2	<a href="#">position</a>	18
6.7	<a href="#">_energy_methods Struct Reference</a>	18
6.7.1	<a href="#">Detailed Description</a>	19
6.7.2	<a href="#">Field Documentation</a>	19
6.7.2.1	<a href="#">consume</a>	19
6.7.2.2	<a href="#">consume_idle</a>	19
6.7.2.3	<a href="#">consume_rx</a>	19
6.7.2.4	<a href="#">consume_tx</a>	19
6.7.2.5	<a href="#">energy_consumed</a>	19
6.7.2.6	<a href="#">energy_remaining</a>	19
6.7.2.7	<a href="#">energy_status</a>	19
6.8	<a href="#">_entityid Struct Reference</a>	20

6.8.1	Detailed Description	20
6.9	<code>_environment_methods</code> Struct Reference	20
6.9.1	Detailed Description	20
6.9.2	Field Documentation	20
6.9.2.1	<code>read_measure</code>	20
6.10	<code>_event</code> Struct Reference	21
6.10.1	Detailed Description	21
6.10.2	Field Documentation	21
6.10.2.1	<code>arg</code>	21
6.10.2.2	<code>call</code>	21
6.10.2.3	<code>callback</code>	21
6.10.2.4	<code>cb</code>	22
6.10.2.5	<code>clock</code>	22
6.10.2.6	<code>id</code>	22
6.10.2.7	<code>nodeid</code>	22
6.10.2.8	<code>packet</code>	22
6.10.2.9	<code>priority</code>	22
6.10.2.10	<code>rx</code>	22
6.10.2.11	<code>u</code>	22
6.11	<code>_fading_methods</code> Struct Reference	22
6.11.1	Detailed Description	23
6.11.2	Field Documentation	23
6.11.2.1	<code>fading</code>	23
6.12	<code>_interferences_methods</code> Struct Reference	23
6.12.1	Detailed Description	23
6.12.2	Field Documentation	23
6.12.2.1	<code>interfere</code>	23
6.13	<code>_io_ctl_message</code> Struct Reference	24
6.13.1	Detailed Description	24
6.14	<code>_ioctl_message</code> Struct Reference	24
6.14.1	Detailed Description	24
6.14.2	Field Documentation	24
6.14.2.1	<code>body</code>	24
6.14.2.2	<code>real_size</code>	24

6.14.2.3	size	24
6.14.2.4	type	25
6.15	_mac_methods Struct Reference	25
6.15.1	Detailed Description	25
6.15.2	Field Documentation	25
6.15.2.1	get_header_real_size	25
6.15.2.2	get_header_size	25
6.15.2.3	rx	25
6.15.2.4	set_header	26
6.15.2.5	tx	26
6.16	_measureid Struct Reference	26
6.16.1	Detailed Description	26
6.17	_mobility_methods Struct Reference	26
6.17.1	Detailed Description	26
6.17.2	Field Documentation	27
6.17.2.1	update_position	27
6.18	_model Struct Reference	27
6.18.1	Detailed Description	27
6.18.2	Field Documentation	27
6.18.2.1	author	27
6.18.2.2	count	27
6.18.2.3	exported	28
6.18.2.4	measure	28
6.18.2.5	online	28
6.18.2.6	type	28
6.18.2.7	version	28
6.19	_modulation_methods Struct Reference	28
6.19.1	Detailed Description	28
6.19.2	Field Documentation	28
6.19.2.1	bit_per_symbol	29
6.19.2.2	modulate	29
6.20	_monitor_methods Struct Reference	29
6.20.1	Detailed Description	29
6.20.2	Field Documentation	29

6.20.2.1	monitor_death	29
6.20.2.2	monitor_event	29
6.20.2.3	monitor_register_callback	30
6.21	_nodeid Struct Reference	30
6.21.1	Detailed Description	30
6.22	_noise_methods Struct Reference	30
6.22.1	Detailed Description	30
6.22.2	Field Documentation	30
6.22.2.1	noise	31
6.23	_packet Struct Reference	31
6.23.1	Detailed Description	32
6.23.2	Field Documentation	32
6.23.2.1	antenna	32
6.23.2.2	ber	32
6.23.2.3	channel	32
6.23.2.4	clock0	32
6.23.2.5	clock1	33
6.23.2.6	data	33
6.23.2.7	duration	33
6.23.2.8	id	33
6.23.2.9	modulation	33
6.23.2.10	node	33
6.23.2.11	noise_mW	33
6.23.2.12	PER	33
6.23.2.13	real_size	34
6.23.2.14	rxdBm	34
6.23.2.15	rxmW	34
6.23.2.16	size	34
6.23.2.17	Tb	34
6.23.2.18	txdBm	34
6.23.2.19	type	34
6.23.2.20	worldsens_freq	34
6.23.2.21	worldsens_mod	35
6.24	_packetid Struct Reference	35

6.24.1 Detailed Description . . . . .	35
6.25 <code>_param</code> Struct Reference . . . . .	35
6.25.1 Detailed Description . . . . .	35
6.25.2 Field Documentation . . . . .	36
6.25.2.1 <code>key</code> . . . . .	36
6.25.2.2 <code>value</code> . . . . .	36
6.26 <code>_position</code> Struct Reference . . . . .	36
6.26.1 Detailed Description . . . . .	36
6.26.2 Field Documentation . . . . .	36
6.26.2.1 <code>x</code> . . . . .	36
6.26.2.2 <code>y</code> . . . . .	37
6.26.2.3 <code>z</code> . . . . .	37
6.27 <code>_propagation_methods</code> Struct Reference . . . . .	37
6.27.1 Detailed Description . . . . .	37
6.27.2 Field Documentation . . . . .	37
6.27.2.1 <code>propagation</code> . . . . .	37
6.28 <code>_radio_methods</code> Struct Reference . . . . .	38
6.28.1 Detailed Description . . . . .	38
6.28.2 Field Documentation . . . . .	38
6.28.2.1 <code>cs</code> . . . . .	38
6.28.2.2 <code>get_channel</code> . . . . .	39
6.28.2.3 <code>get_cs</code> . . . . .	39
6.28.2.4 <code>get_header_real_size</code> . . . . .	39
6.28.2.5 <code>get_header_size</code> . . . . .	39
6.28.2.6 <code>get_modulation</code> . . . . .	39
6.28.2.7 <code>get_modulation_bit_per_symbol</code> . . . . .	39
6.28.2.8 <code>get_noise</code> . . . . .	39
6.28.2.9 <code>get_power</code> . . . . .	39
6.28.2.10 <code>get_sensibility</code> . . . . .	39
6.28.2.11 <code>get_Tb</code> . . . . .	39
6.28.2.12 <code>get_Ts</code> . . . . .	40
6.28.2.13 <code>rx</code> . . . . .	40
6.28.2.14 <code>set_channel</code> . . . . .	40
6.28.2.15 <code>set_header</code> . . . . .	40



6.28.2.16	set_modulation	40
6.28.2.17	set_power	40
6.28.2.18	set_sensibility	40
6.28.2.19	set_Ts	40
6.28.2.20	sleep	40
6.28.2.21	tx	40
6.28.2.22	tx_end	41
6.28.2.23	wakeup	41
6.29	_routing_methods Struct Reference	41
6.29.1	Detailed Description	41
6.29.2	Field Documentation	41
6.29.2.1	get_header_real_size	41
6.29.2.2	get_header_size	41
6.29.2.3	rx	42
6.29.2.4	set_header	42
6.29.2.5	tx	42
6.30	_shadowing_methods Struct Reference	42
6.30.1	Detailed Description	42
6.30.2	Field Documentation	42
6.30.2.1	shadowing	42
6.31	_worldsens_c_byte_tx Struct Reference	43
6.31.1	Detailed Description	43
6.31.2	Field Documentation	43
6.31.2.1	antenna_id	43
6.31.2.2	data	43
6.31.2.3	duration	43
6.31.2.4	freq	43
6.31.2.5	node_id	43
6.31.2.6	period	44
6.31.2.7	power_dbm	44
6.31.2.8	type	44
6.31.2.9	wsim_mod_id	44
6.31.2.10	wsnet_mod_id	44
6.32	_worldsens_c_connect_req Struct Reference	44

6.32.1 Detailed Description . . . . .	44
6.32.2 Field Documentation . . . . .	44
6.32.2.1 node_id . . . . .	44
6.32.2.2 type . . . . .	45
6.33 _worldsens_c_disconnect Struct Reference . . . . .	45
6.33.1 Detailed Description . . . . .	45
6.33.2 Field Documentation . . . . .	45
6.33.2.1 node_id . . . . .	45
6.33.2.2 type . . . . .	45
6.34 _worldsens_c_header Struct Reference . . . . .	45
6.34.1 Detailed Description . . . . .	46
6.34.2 Field Documentation . . . . .	46
6.34.2.1 id . . . . .	46
6.34.2.2 type . . . . .	46
6.35 _worldsens_c_measure_req Struct Reference . . . . .	46
6.35.1 Detailed Description . . . . .	46
6.35.2 Field Documentation . . . . .	46
6.35.2.1 measure_id . . . . .	46
6.35.2.2 node_id . . . . .	46
6.35.2.3 period . . . . .	47
6.35.2.4 type . . . . .	47
6.36 _worldsens_c_sync_ack Struct Reference . . . . .	47
6.36.1 Detailed Description . . . . .	47
6.36.2 Field Documentation . . . . .	47
6.36.2.1 node_id . . . . .	47
6.36.2.2 rp_id . . . . .	47
6.36.2.3 type . . . . .	47
6.37 _worldsens_pkt Union Reference . . . . .	48
6.37.1 Detailed Description . . . . .	48
6.37.2 Field Documentation . . . . .	48
6.37.2.1 bktrk . . . . .	48
6.37.2.2 byte_rx . . . . .	48
6.37.2.3 byte_sr_rx . . . . .	48
6.37.2.4 byte_tx . . . . .	49

6.37.2.5	c_header	49
6.37.2.6	cnx_req	49
6.37.2.7	cnx_rsp_nok	49
6.37.2.8	cnx_rsp_ok	49
6.37.2.9	disconnect	49
6.37.2.10	kill	49
6.37.2.11	killsim	49
6.37.2.12	measure_req	49
6.37.2.13	measure_rsp	49
6.37.2.14	measure_sr_rsp	50
6.37.2.15	s_header	50
6.37.2.16	sync_ack	50
6.37.2.17	sync_release	50
6.37.2.18	sync_reminder	50
6.38	_worldsens_s_backtrack Struct Reference	50
6.38.1	Detailed Description	50
6.38.2	Field Documentation	51
6.38.2.1	rp_duration	51
6.38.2.2	rp_next	51
6.38.2.3	seq	51
6.38.2.4	type	51
6.39	_worldsens_s_byte_rx Struct Reference	51
6.39.1	Detailed Description	51
6.39.2	Field Documentation	52
6.39.2.1	antenna_id	52
6.39.2.2	data	52
6.39.2.3	freq	52
6.39.2.4	node_id	52
6.39.2.5	power_dbm	52
6.39.2.6	seq	52
6.39.2.7	sinr	52
6.39.2.8	type	52
6.39.2.9	wsim_mod_id	52
6.40	_worldsens_s_byte_sr_rx Struct Reference	53

6.40.1 Detailed Description . . . . .	53
6.40.2 Field Documentation . . . . .	53
6.40.2.1 antenna_id . . . . .	53
6.40.2.2 data . . . . .	53
6.40.2.3 freq . . . . .	53
6.40.2.4 node_id . . . . .	53
6.40.2.5 power_dbm . . . . .	53
6.40.2.6 rp_duration . . . . .	54
6.40.2.7 rp_next . . . . .	54
6.40.2.8 seq . . . . .	54
6.40.2.9 sinr . . . . .	54
6.40.2.10 type . . . . .	54
6.40.2.11 wsim_mod_id . . . . .	54
6.41 _worldsens_s_connect_rsp_nok Struct Reference . . . . .	54
6.41.1 Detailed Description . . . . .	54
6.41.2 Field Documentation . . . . .	55
6.41.2.1 seq . . . . .	55
6.41.2.2 type . . . . .	55
6.42 _worldsens_s_connect_rsp_ok Struct Reference . . . . .	55
6.42.1 Detailed Description . . . . .	55
6.42.2 Field Documentation . . . . .	55
6.42.2.1 n_antenna_id . . . . .	55
6.42.2.2 n_measure_id . . . . .	55
6.42.2.3 n_modulation_id . . . . .	56
6.42.2.4 names_and_ids . . . . .	56
6.42.2.5 rp_duration . . . . .	56
6.42.2.6 rp_next . . . . .	56
6.42.2.7 seq . . . . .	56
6.42.2.8 type . . . . .	56
6.43 _worldsens_s_header Struct Reference . . . . .	56
6.43.1 Detailed Description . . . . .	56
6.43.2 Field Documentation . . . . .	57
6.43.2.1 seq . . . . .	57
6.43.2.2 type . . . . .	57

6.44	<a href="#">_worldsens_s_kill Struct Reference</a>	57
6.44.1	<a href="#">Detailed Description</a>	57
6.44.2	<a href="#">Field Documentation</a>	57
6.44.2.1	<a href="#">node_id</a>	57
6.44.2.2	<a href="#">seq</a>	57
6.44.2.3	<a href="#">type</a>	57
6.45	<a href="#">_worldsens_s_killsim Struct Reference</a>	58
6.45.1	<a href="#">Detailed Description</a>	58
6.45.2	<a href="#">Field Documentation</a>	58
6.45.2.1	<a href="#">seq</a>	58
6.45.2.2	<a href="#">type</a>	58
6.46	<a href="#">_worldsens_s_measure_rsp Struct Reference</a>	58
6.46.1	<a href="#">Detailed Description</a>	59
6.46.2	<a href="#">Field Documentation</a>	59
6.46.2.1	<a href="#">measure_id</a>	59
6.46.2.2	<a href="#">measure_val</a>	59
6.46.2.3	<a href="#">node_id</a>	59
6.46.2.4	<a href="#">seq</a>	59
6.46.2.5	<a href="#">type</a>	59
6.47	<a href="#">_worldsens_s_measure_sr_rsp Struct Reference</a>	59
6.47.1	<a href="#">Detailed Description</a>	60
6.47.2	<a href="#">Field Documentation</a>	60
6.47.2.1	<a href="#">measure_id</a>	60
6.47.2.2	<a href="#">measure_val</a>	60
6.47.2.3	<a href="#">node_id</a>	60
6.47.2.4	<a href="#">rp_duration</a>	60
6.47.2.5	<a href="#">rp_next</a>	60
6.47.2.6	<a href="#">seq</a>	60
6.47.2.7	<a href="#">type</a>	60
6.48	<a href="#">_worldsens_s_sync_release Struct Reference</a>	60
6.48.1	<a href="#">Detailed Description</a>	61
6.48.2	<a href="#">Field Documentation</a>	61
6.48.2.1	<a href="#">rp_duration</a>	61
6.48.2.2	<a href="#">rp_next</a>	61

6.48.2.3	seq	61
6.48.2.4	type	61
6.49	<a href="#">_worldsens_s_sync_reminder Struct Reference</a>	61
6.49.1	Detailed Description	62
6.49.2	Field Documentation	62
6.49.2.1	rp_next	62
6.49.2.2	seq	62
6.49.2.3	type	62
6.50	<a href="#">bernoulli_args_s Struct Reference</a>	62
6.50.1	Detailed Description	62
6.50.2	Field Documentation	62
6.50.2.1	p	62
6.51	<a href="#">beta_args_s Struct Reference</a>	63
6.51.1	Detailed Description	63
6.51.2	Field Documentation	63
6.51.2.1	a	63
6.51.2.2	b	63
6.52	<a href="#">binomial_args_s Struct Reference</a>	63
6.52.1	Detailed Description	63
6.52.2	Field Documentation	64
6.52.2.1	n	64
6.52.2.2	p	64
6.53	<a href="#">bivariate_gaussian_args_s Struct Reference</a>	64
6.53.1	Detailed Description	64
6.53.2	Field Documentation	64
6.53.2.1	return_x	64
6.53.2.2	return_y	64
6.53.2.3	rho	64
6.53.2.4	sigma_x	65
6.53.2.5	sigma_y	65
6.54	<a href="#">cauchy_args_s Struct Reference</a>	65
6.54.1	Detailed Description	65
6.54.2	Field Documentation	65
6.54.2.1	a	65

6.55	chisq_args_s Struct Reference	65
6.55.1	Detailed Description	66
6.55.2	Field Documentation	66
6.55.2.1	nu	66
6.56	exponential_args_s Struct Reference	66
6.56.1	Detailed Description	66
6.56.2	Field Documentation	66
6.56.2.1	mu	66
6.57	exponential_s Struct Reference	66
6.57.1	Detailed Description	67
6.57.2	Field Documentation	67
6.57.2.1	initial_value	67
6.57.2.2	offset	67
6.57.2.3	rank	67
6.57.2.4	ratio	67
6.58	exppow_args_s Struct Reference	67
6.58.1	Detailed Description	68
6.58.2	Field Documentation	68
6.58.2.1	a	68
6.58.2.2	b	68
6.59	gamma_args_s Struct Reference	68
6.59.1	Detailed Description	68
6.59.2	Field Documentation	68
6.59.2.1	a	68
6.59.2.2	b	68
6.60	gaussian_args_s Struct Reference	69
6.60.1	Detailed Description	69
6.60.2	Field Documentation	69
6.60.2.1	sigma	69
6.61	gaussian_tail_args_s Struct Reference	69
6.61.1	Detailed Description	69
6.61.2	Field Documentation	69
6.61.2.1	a	69
6.61.2.2	sigma	70

6.62	<a href="#">geometric_args_s Struct Reference</a>	70
6.62.1	<a href="#">Detailed Description</a>	70
6.62.2	<a href="#">Field Documentation</a>	70
6.62.2.1	<a href="#">p</a>	70
6.63	<a href="#">gumbel_t1_args_s Struct Reference</a>	70
6.63.1	<a href="#">Detailed Description</a>	70
6.63.2	<a href="#">Field Documentation</a>	71
6.63.2.1	<a href="#">a</a>	71
6.63.2.2	<a href="#">b</a>	71
6.64	<a href="#">gumbel_t2_args_s Struct Reference</a>	71
6.64.1	<a href="#">Detailed Description</a>	71
6.64.2	<a href="#">Field Documentation</a>	71
6.64.2.1	<a href="#">a</a>	71
6.64.2.2	<a href="#">b</a>	71
6.65	<a href="#">hyper_geometric_args_s Struct Reference</a>	71
6.65.1	<a href="#">Detailed Description</a>	72
6.65.2	<a href="#">Field Documentation</a>	72
6.65.2.1	<a href="#">n1</a>	72
6.65.2.2	<a href="#">n2</a>	72
6.65.2.3	<a href="#">t</a>	72
6.66	<a href="#">laplace_args_s Struct Reference</a>	72
6.66.1	<a href="#">Detailed Description</a>	72
6.66.2	<a href="#">Field Documentation</a>	73
6.66.2.1	<a href="#">a</a>	73
6.67	<a href="#">levy_alpha_stable_s Struct Reference</a>	73
6.67.1	<a href="#">Detailed Description</a>	73
6.67.2	<a href="#">Field Documentation</a>	73
6.67.2.1	<a href="#">alpha</a>	73
6.67.2.2	<a href="#">c</a>	73
6.68	<a href="#">logarithmic_args_s Struct Reference</a>	73
6.68.1	<a href="#">Detailed Description</a>	74
6.68.2	<a href="#">Field Documentation</a>	74
6.68.2.1	<a href="#">p</a>	74
6.69	<a href="#">logistic_args_s Struct Reference</a>	74



6.69.1 Detailed Description . . . . .	74
6.69.2 Field Documentation . . . . .	74
6.69.2.1 a . . . . .	74
6.70 lognormal_args_s Struct Reference . . . . .	74
6.70.1 Detailed Description . . . . .	75
6.70.2 Field Documentation . . . . .	75
6.70.2.1 sigma . . . . .	75
6.70.2.2 zeta . . . . .	75
6.71 pareto_args_s Struct Reference . . . . .	75
6.71.1 Detailed Description . . . . .	75
6.71.2 Field Documentation . . . . .	75
6.71.2.1 a . . . . .	75
6.71.2.2 b . . . . .	76
6.72 poisson_args_s Struct Reference . . . . .	76
6.72.1 Detailed Description . . . . .	76
6.72.2 Field Documentation . . . . .	76
6.72.2.1 mu . . . . .	76
6.73 qtimer_s Struct Reference . . . . .	76
6.73.1 Detailed Description . . . . .	77
6.73.2 Field Documentation . . . . .	77
6.73.2.1 c . . . . .	77
6.73.2.2 callback_function . . . . .	77
6.73.2.3 conditional_end . . . . .	77
6.73.2.4 next_trigger . . . . .	77
6.73.2.5 trigger_parameters . . . . .	77
6.74 rayleigh_s Struct Reference . . . . .	77
6.74.1 Detailed Description . . . . .	78
6.74.2 Field Documentation . . . . .	78
6.74.2.1 sigma . . . . .	78
6.75 rayleigh_tail_s Struct Reference . . . . .	78
6.75.1 Detailed Description . . . . .	78
6.75.2 Field Documentation . . . . .	78
6.75.2.1 a . . . . .	78
6.75.2.2 sigma . . . . .	78

6.76	spherical_vector_2d_args_s Struct Reference	79
6.76.1	Detailed Description	79
6.76.2	Field Documentation	79
6.76.2.1	return_x	79
6.76.2.2	return_y	79
6.77	spherical_vector_3d_args_s Struct Reference	79
6.77.1	Detailed Description	79
6.77.2	Field Documentation	80
6.77.2.1	return_x	80
6.77.2.2	return_y	80
6.77.2.3	return_z	80
6.78	uniform_args_s Struct Reference	80
6.78.1	Detailed Description	80
6.78.2	Field Documentation	80
6.78.2.1	a	80
6.78.2.2	b	80
6.79	uniform_random_s Struct Reference	81
6.79.1	Detailed Description	81
6.79.2	Field Documentation	81
6.79.2.1	max_value	81
6.79.2.2	min_value	81
6.80	weibull_args_s Struct Reference	81
6.80.1	Detailed Description	81
6.80.2	Field Documentation	82
6.80.2.1	a	82
6.80.2.2	b	82
<b>7</b>	<b>File Documentation</b>	<b>83</b>
7.1	antenna.h File Reference	83
7.1.1	Detailed Description	83
7.1.2	Function Documentation	84
7.1.2.1	antenna_gain_rx	84
7.1.2.2	antenna_gain_tx	84
7.1.2.3	antenna_get_angle	84

7.1.2.4	<a href="#">antenna_get_loss</a>	85
7.1.2.5	<a href="#">antenna_rx</a>	85
7.1.2.6	<a href="#">antenna_set_angle</a>	85
7.2	<a href="#">battery.h File Reference</a>	85
7.2.1	<a href="#">Detailed Description</a>	86
7.2.2	<a href="#">Function Documentation</a>	86
7.2.2.1	<a href="#">battery_consume</a>	86
7.2.2.2	<a href="#">battery_consume_idle</a>	86
7.2.2.3	<a href="#">battery_consume_rx</a>	87
7.2.2.4	<a href="#">battery_consume_tx</a>	87
7.2.2.5	<a href="#">battery_consumed</a>	87
7.2.2.6	<a href="#">battery_remaining</a>	87
7.2.2.7	<a href="#">battery_status</a>	88
7.3	<a href="#">das.h File Reference</a>	88
7.3.1	<a href="#">Detailed Description</a>	89
7.3.2	<a href="#">Typedef Documentation</a>	89
7.3.2.1	<a href="#">das_delete_func_t</a>	89
7.3.3	<a href="#">Function Documentation</a>	89
7.3.3.1	<a href="#">das_create</a>	89
7.3.3.2	<a href="#">das_delete</a>	89
7.3.3.3	<a href="#">das_destroy</a>	90
7.3.3.4	<a href="#">das_find</a>	90
7.3.3.5	<a href="#">das_getsize</a>	90
7.3.3.6	<a href="#">das_init</a>	90
7.3.3.7	<a href="#">das_init_traverse</a>	91
7.3.3.8	<a href="#">das_insert</a>	91
7.3.3.9	<a href="#">das_pop</a>	91
7.3.3.10	<a href="#">das_pop_FIFO</a>	91
7.3.3.11	<a href="#">das_selective_delete</a>	91
7.3.3.12	<a href="#">das_traverse</a>	92
7.4	<a href="#">dbg.h File Reference</a>	92
7.4.1	<a href="#">Detailed Description</a>	93
7.4.2	<a href="#">Define Documentation</a>	93
7.4.2.1	<a href="#">DBG</a>	93

7.4.2.2	DBG_CRIT . . . . .	93
7.4.2.3	DBG_INFO . . . . .	93
7.4.2.4	DBG_NOISE . . . . .	93
7.4.2.5	DBG_VERB . . . . .	93
7.4.2.6	DBG_WARN . . . . .	93
7.4.2.7	DBG_XTRM . . . . .	94
7.4.2.8	DEBUG_MAX . . . . .	94
7.4.2.9	NDBG . . . . .	94
7.4.2.10	NDBG_CRIT . . . . .	94
7.4.2.11	NDBG_INFO . . . . .	94
7.4.2.12	NDBG_NOISE . . . . .	94
7.4.2.13	NDBG_VERB . . . . .	94
7.4.2.14	NDBG_WARN . . . . .	94
7.4.2.15	NDBG_XTRM . . . . .	94
7.4.3	Function Documentation . . . . .	95
7.4.3.1	get_debug . . . . .	95
7.5	entity.h File Reference . . . . .	95
7.5.1	Detailed Description . . . . .	96
7.5.2	Function Documentation . . . . .	96
7.5.2.1	get_antenna_entities . . . . .	96
7.5.2.2	get_application_entities . . . . .	96
7.5.2.3	get_energy_entity . . . . .	97
7.5.2.4	get_entity_bindings_down . . . . .	97
7.5.2.5	get_entity_bindings_up . . . . .	97
7.5.2.6	get_entity_links_down . . . . .	98
7.5.2.7	get_entity_links_down_nbr . . . . .	98
7.5.2.8	get_entity_links_up . . . . .	98
7.5.2.9	get_entity_links_up_nbr . . . . .	99
7.5.2.10	get_entity_name . . . . .	99
7.5.2.11	get_entity_private_data . . . . .	99
7.5.2.12	get_entity_type . . . . .	99
7.5.2.13	get_mac_entities . . . . .	100
7.5.2.14	get_mobility_entity . . . . .	100
7.5.2.15	get_node_private_data . . . . .	100

7.5.2.16	<a href="#">get_radio_entities</a>	101
7.5.2.17	<a href="#">get_routing_entities</a>	101
7.5.2.18	<a href="#">set_entity_private_data</a>	101
7.5.2.19	<a href="#">set_node_private_data</a>	101
7.6	<a href="#">hadas.h File Reference</a>	102
7.6.1	<a href="#">Detailed Description</a>	102
7.6.2	<a href="#">Typedef Documentation</a>	102
7.6.2.1	<a href="#">hash_equal_t</a>	102
7.6.2.2	<a href="#">hash_hash_t</a>	103
7.6.3	<a href="#">Function Documentation</a>	103
7.6.3.1	<a href="#">hadas_create</a>	103
7.6.3.2	<a href="#">hadas_delete</a>	103
7.6.3.3	<a href="#">hadas_destroy</a>	103
7.6.3.4	<a href="#">hadas_get</a>	103
7.6.3.5	<a href="#">hadas_init</a>	104
7.6.3.6	<a href="#">hadas_insert</a>	104
7.7	<a href="#">ioctl_message.h File Reference</a>	104
7.7.1	<a href="#">Function Documentation</a>	105
7.7.1.1	<a href="#">get_ioctl_message_body</a>	105
7.7.1.2	<a href="#">get_ioctl_message_real_size</a>	105
7.7.1.3	<a href="#">get_ioctl_message_size</a>	105
7.7.1.4	<a href="#">get_ioctl_message_type</a>	105
7.7.1.5	<a href="#">ioctl_message_body_duplicate</a>	105
7.7.1.6	<a href="#">ioctl_message_create</a>	106
7.7.1.7	<a href="#">ioctl_message_dealloc</a>	106
7.8	<a href="#">log.h File Reference</a>	106
7.8.1	<a href="#">Detailed Description</a>	107
7.8.2	<a href="#">Define Documentation</a>	107
7.8.2.1	<a href="#">PRINT_ANTENNA</a>	107
7.8.2.2	<a href="#">PRINT_APPLICATION</a>	107
7.8.2.3	<a href="#">PRINT_ENERGY</a>	107
7.8.2.4	<a href="#">PRINT_ENVIRONMENT</a>	107
7.8.2.5	<a href="#">PRINT_INTERFERENCES</a>	107
7.8.2.6	<a href="#">PRINT_MAC</a>	107

7.8.2.7	PRINT_MOBILITY	107
7.8.2.8	PRINT_MODULATION	108
7.8.2.9	PRINT_MONITOR	108
7.8.2.10	PRINT_PROPAGATION	108
7.8.2.11	PRINT_RADIO	108
7.8.2.12	PRINT_REPLAY	108
7.8.2.13	PRINT_ROUTING	108
7.8.2.14	PRINT_WORLDSENS	108
7.9	measure.h File Reference	108
7.9.1	Detailed Description	109
7.9.2	Function Documentation	109
7.9.2.1	get_measureid_by_name	109
7.9.2.2	READ_MEASURE	109
7.10	medium.h File Reference	109
7.10.1	Detailed Description	110
7.10.2	Define Documentation	110
7.10.2.1	MAX_SNR	110
7.10.2.2	MIN_DBM	110
7.10.3	Function Documentation	111
7.10.3.1	dBm2mW	111
7.10.3.2	MEDIA_GET_NOISE	111
7.10.3.3	MEDIA_TX	111
7.10.3.4	mW2dBm	111
7.11	mem_fs.h File Reference	112
7.11.1	Detailed Description	112
7.11.2	Function Documentation	112
7.11.2.1	mem_fs_alloc	112
7.11.2.2	mem_fs_clean	113
7.11.2.3	mem_fs_dealloc	113
7.11.2.4	mem_fs_slice_declare	113
7.12	models.h File Reference	113
7.12.1	Detailed Description	116
7.12.2	Define Documentation	116
7.12.2.1	MODELTYPE_ANTENNA	116

7.12.2.2	MODELTYPE_APPLICATION . . . . .	116
7.12.2.3	MODELTYPE_ENERGY . . . . .	116
7.12.2.4	MODELTYPE_ENVIRONMENT . . . . .	116
7.12.2.5	MODELTYPE_FADING . . . . .	116
7.12.2.6	MODELTYPE_INTERFERENCES . . . . .	116
7.12.2.7	MODELTYPE_MAC . . . . .	116
7.12.2.8	MODELTYPE_MOBILITY . . . . .	117
7.12.2.9	MODELTYPE_MODULATION . . . . .	117
7.12.2.10	MODELTYPE_MONITOR . . . . .	117
7.12.2.11	MODELTYPE_NOISE . . . . .	117
7.12.2.12	MODELTYPE_PROPAGATION . . . . .	117
7.12.2.13	MODELTYPE_RADIO . . . . .	117
7.12.2.14	MODELTYPE_ROUTING . . . . .	117
7.12.2.15	MODELTYPE_SHADOWING . . . . .	117
7.12.3	Typedef Documentation . . . . .	117
7.12.3.1	antenna_methods_t . . . . .	117
7.12.3.2	application_methods_t . . . . .	117
7.12.3.3	energy_methods_t . . . . .	118
7.12.3.4	environment_methods_t . . . . .	118
7.12.3.5	fading_methods_t . . . . .	118
7.12.3.6	interferences_methods_t . . . . .	118
7.12.3.7	mac_methods_t . . . . .	118
7.12.3.8	mobility_methods_t . . . . .	118
7.12.3.9	model_t . . . . .	118
7.12.3.10	modulation_methods_t . . . . .	118
7.12.3.11	monitor_methods_t . . . . .	118
7.12.3.12	noise_methods_t . . . . .	118
7.12.3.13	propagation_methods_t . . . . .	119
7.12.3.14	radio_methods_t . . . . .	119
7.12.3.15	routing_methods_t . . . . .	119
7.12.3.16	shadowing_methods_t . . . . .	119
7.13	modelutils.h File Reference . . . . .	119
7.13.1	Detailed Description . . . . .	120
7.13.2	Define Documentation . . . . .	120

7.13.2.1 BROADCAST_ADDR . . . . .	120
7.13.3 Function Documentation . . . . .	120
7.13.3.1 end_simulation . . . . .	120
7.13.3.2 GET_HEADER_REAL_SIZE . . . . .	121
7.13.3.3 GET_HEADER_SIZE . . . . .	121
7.13.3.4 get_node_count . . . . .	121
7.13.3.5 get_time . . . . .	121
7.13.3.6 get_topology_area . . . . .	121
7.13.3.7 IOCTL . . . . .	121
7.13.3.8 RX . . . . .	121
7.13.3.9 SET_HEADER . . . . .	121
7.13.3.10 TX . . . . .	121
7.14 modulation.h File Reference . . . . .	121
7.14.1 Detailed Description . . . . .	122
7.14.2 Function Documentation . . . . .	122
7.14.2.1 modulation_bit_per_symbol . . . . .	122
7.15 monitor.h File Reference . . . . .	122
7.15.1 Detailed Description . . . . .	122
7.15.2 Function Documentation . . . . .	123
7.15.2.1 monitor_register_callback . . . . .	123
7.15.2.2 monitor_simulation . . . . .	123
7.16 node.h File Reference . . . . .	123
7.16.1 Detailed Description . . . . .	124
7.16.2 Function Documentation . . . . .	124
7.16.2.1 distance . . . . .	124
7.16.2.2 get_node_position . . . . .	124
7.16.2.3 is_node_alive . . . . .	124
7.16.2.4 node_kill . . . . .	125
7.17 options.h File Reference . . . . .	125
7.17.1 Detailed Description . . . . .	125
7.17.2 Define Documentation . . . . .	126
7.17.2.1 CHANNELS_NUMBER . . . . .	126
7.17.2.2 LOG_APPLICATION . . . . .	126
7.17.2.3 LOG_MAC . . . . .	126



7.17.2.4	LOG_REPLAY	126
7.17.2.5	LOG_WORLDSENS	126
7.17.2.6	SNR_ERRORS	126
7.17.2.7	SNR_STEP	126
7.18	packet.h File Reference	126
7.18.1	Detailed Description	127
7.18.2	Function Documentation	127
7.18.2.1	packet_alloc	127
7.18.2.2	packet_clone	127
7.18.2.3	packet_create	128
7.18.2.4	packet_dealloc	128
7.19	param.h File Reference	128
7.19.1	Function Documentation	129
7.19.1.1	get_param_distance	129
7.19.1.2	get_param_double	129
7.19.1.3	get_param_double_range	130
7.19.1.4	get_param_entity	130
7.19.1.5	get_param_integer	130
7.19.1.6	get_param_nodeid	131
7.19.1.7	get_param_time	131
7.19.1.8	get_param_x_position	131
7.19.1.9	get_param_y_position	132
7.19.1.10	get_param_z_position	132
7.20	probabilistic_distribution.h File Reference	132
7.20.1	Detailed Description	135
7.20.2	Define Documentation	135
7.20.2.1	BERNOULLI	135
7.20.2.2	BETA	135
7.20.2.3	BINOMIAL	135
7.20.2.4	CAUCHY	135
7.20.2.5	CHI_SQUARED	135
7.20.2.6	EXPONENTIAL	135
7.20.2.7	EXPONENTIAL_POWER	135
7.20.2.8	GAMMA	136

7.20.2.9	GAUSSIAN	136
7.20.2.10	GAUSSIAN_TAIL	136
7.20.2.11	GEOMETRIC	136
7.20.2.12	GUMBELL_1	136
7.20.2.13	GUMBELL_2	136
7.20.2.14	HYPERGEOMETRIC	136
7.20.2.15	LANDAU	136
7.20.2.16	LAPLACE	136
7.20.2.17	LEVY_ALPHA_STABLE	136
7.20.2.18	LOG_NORMAL	137
7.20.2.19	LOGARITHMIC	137
7.20.2.20	LOGISTIC	137
7.20.2.21	PARETO	137
7.20.2.22	POISSON	137
7.20.2.23	RAYLEIGH	137
7.20.2.24	RAYLEIGH_TAIL	137
7.20.2.25	UNIFORM	137
7.20.2.26	WEIBULL	137
7.20.3	Typedef Documentation	137
7.20.3.1	bernoulli_args_t	137
7.20.3.2	beta_args_t	137
7.20.3.3	binomial_args_t	138
7.20.3.4	bivariate_gaussian_args_t	138
7.20.3.5	cauchy_args_t	138
7.20.3.6	chisq_args_t	138
7.20.3.7	distribution_function_t	138
7.20.3.8	exponential_args_t	138
7.20.3.9	exppow_args_t	138
7.20.3.10	gamma_args_t	138
7.20.3.11	gaussian_args_t	138
7.20.3.12	gaussian_tail_args_t	138
7.20.3.13	geometric_args_t	138
7.20.3.14	gumbel_t1_args_t	138
7.20.3.15	gumbel_t2_args_t	138

7.20.3.16	<a href="#">hyper_geometric_args_t</a>	138
7.20.3.17	<a href="#">laplace_args_t</a>	138
7.20.3.18	<a href="#">levy_alpha_stable_args_t</a>	138
7.20.3.19	<a href="#">logarithmic_args_t</a>	138
7.20.3.20	<a href="#">logistic_args_t</a>	138
7.20.3.21	<a href="#">lognormal_args_t</a>	138
7.20.3.22	<a href="#">pareto_args_t</a>	138
7.20.3.23	<a href="#">poisson_args_t</a>	138
7.20.3.24	<a href="#">rayleigh_args_t</a>	138
7.20.3.25	<a href="#">rayleigh_tail_args_t</a>	138
7.20.3.26	<a href="#">spherical_vector_2d_args_t</a>	139
7.20.3.27	<a href="#">spherical_vector_3d_args_t</a>	139
7.20.3.28	<a href="#">uniform_args_t</a>	139
7.20.3.29	<a href="#">weibull_args_t</a>	139
7.20.4	Function Documentation	139
7.20.4.1	<a href="#">get_distribution_function_by_type</a>	139
7.21	radio.h File Reference	139
7.21.1	Detailed Description	140
7.21.2	Function Documentation	140
7.21.2.1	<a href="#">radio_cs</a>	140
7.21.2.2	<a href="#">radio_get_channel</a>	140
7.21.2.3	<a href="#">radio_get_cs</a>	141
7.21.2.4	<a href="#">radio_get_modulation</a>	141
7.21.2.5	<a href="#">radio_get_modulation_bit_per_symbol</a>	141
7.21.2.6	<a href="#">radio_get_noise</a>	142
7.21.2.7	<a href="#">radio_get_power</a>	142
7.21.2.8	<a href="#">radio_get_sensibility</a>	142
7.21.2.9	<a href="#">radio_get_Tb</a>	142
7.21.2.10	<a href="#">radio_get_Ts</a>	143
7.21.2.11	<a href="#">radio_set_channel</a>	143
7.21.2.12	<a href="#">radio_set_modulation</a>	143
7.21.2.13	<a href="#">radio_set_power</a>	143
7.21.2.14	<a href="#">radio_set_sensibility</a>	144
7.21.2.15	<a href="#">radio_set_Ts</a>	144

7.21.2.16	radio_sleep	144
7.21.2.17	radio_wakeup	144
7.22	rng.h File Reference	144
7.22.1	Detailed Description	146
7.22.2	Define Documentation	147
7.22.2.1	DEFAULT_RNG	147
7.22.2.2	MT19937	147
7.22.2.3	RANLUX	147
7.22.2.4	RANLUX389	147
7.22.2.5	RANLXD1	147
7.22.2.6	RANLXD2	147
7.22.2.7	RANLXS0	147
7.22.2.8	RANLXS1	147
7.22.2.9	RANLXS2	147
7.22.2.10	RNG_DEFAULT_RETRY_ATTEMPTS	147
7.22.3	Function Documentation	148
7.22.3.1	create_rng	148
7.22.3.2	get_random_distance	148
7.22.3.3	get_random_distance_gsl	148
7.22.3.4	get_random_double	148
7.22.3.5	get_random_double_gsl	149
7.22.3.6	get_random_double_range	149
7.22.3.7	get_random_double_range_gsl	149
7.22.3.8	get_random_integer	150
7.22.3.9	get_random_integer_gsl	150
7.22.3.10	get_random_integer_range	150
7.22.3.11	get_random_integer_range_gsl	150
7.22.3.12	get_random_node	151
7.22.3.13	get_random_node_gsl	151
7.22.3.14	get_random_time	151
7.22.3.15	get_random_time_gsl	152
7.22.3.16	get_random_time_range	152
7.22.3.17	get_random_time_range_gsl	152
7.22.3.18	get_random_x_position	153

7.22.3.19	<a href="#">get_random_x_position_gsl</a>	153
7.22.3.20	<a href="#">get_random_y_position</a>	153
7.22.3.21	<a href="#">get_random_y_position_gsl</a>	153
7.22.3.22	<a href="#">get_random_z_position</a>	154
7.22.3.23	<a href="#">get_random_z_position_gsl</a>	154
7.22.3.24	<a href="#">get_rng_by_id</a>	154
7.23	<a href="#">scheduler.h File Reference</a>	154
7.23.1	<a href="#">Detailed Description</a>	155
7.23.2	<a href="#">Typedef Documentation</a>	155
7.23.2.1	<a href="#">event_t</a>	155
7.23.3	<a href="#">Function Documentation</a>	155
7.23.3.1	<a href="#">scheduler_add_callback</a>	155
7.23.3.2	<a href="#">scheduler_delete_callback</a>	156
7.24	<a href="#">sodas.h File Reference</a>	156
7.24.1	<a href="#">Detailed Description</a>	157
7.24.2	<a href="#">Typedef Documentation</a>	157
7.24.2.1	<a href="#">sodas_compare_t</a>	157
7.24.3	<a href="#">Function Documentation</a>	157
7.24.3.1	<a href="#">sodas_create</a>	157
7.24.3.2	<a href="#">sodas_delete</a>	157
7.24.3.3	<a href="#">sodas_destroy</a>	158
7.24.3.4	<a href="#">sodas_init</a>	158
7.24.3.5	<a href="#">sodas_insert</a>	158
7.24.3.6	<a href="#">sodas_pop</a>	158
7.24.3.7	<a href="#">sodas_see_first</a>	158
7.25	<a href="#">spadas.h File Reference</a>	159
7.25.1	<a href="#">Detailed Description</a>	159
7.25.2	<a href="#">Function Documentation</a>	160
7.25.2.1	<a href="#">spadas_create</a>	160
7.25.2.2	<a href="#">spadas_delete</a>	160
7.25.2.3	<a href="#">spadas_destroy</a>	160
7.25.2.4	<a href="#">spadas_init</a>	160
7.25.2.5	<a href="#">spadas_insert</a>	161
7.25.2.6	<a href="#">spadas_rangesearch</a>	161

7.25.2.7	spadas_update	161
7.26	timer.h File Reference	161
7.26.1	Detailed Description	162
7.26.2	Typedef Documentation	163
7.26.2.1	exponential_parameters_t	163
7.26.2.2	qtimer_t	163
7.26.2.3	uniform_random_parameters_t	163
7.26.3	Function Documentation	163
7.26.3.1	change_parameter	163
7.26.3.2	create_timer	163
7.26.3.3	destroy_timer	163
7.26.3.4	exponential_trigger	163
7.26.3.5	fetch_timer	163
7.26.3.6	never_stop	163
7.26.3.7	periodic_trigger	163
7.26.3.8	start_timer	163
7.26.3.9	timer_clean	163
7.26.3.10	timer_init	163
7.26.3.11	uniform_random_trigger	163
7.27	types.h File Reference	163
7.27.1	Detailed Description	165
7.27.2	Typedef Documentation	165
7.27.2.1	angle_t	165
7.27.2.2	array_t	165
7.27.2.3	call_t	165
7.27.2.4	callback_t	165
7.27.2.5	destination_t	165
7.27.2.6	entityid_t	165
7.27.2.7	ioctl_message_t	166
7.27.2.8	measureid_t	166
7.27.2.9	nodeid_t	166
7.27.2.10	packet_t	166
7.27.2.11	packetid_t	166
7.27.2.12	param_t	166

7.27.2.13 position_t . . . . .	166
7.28 worldsens_debug.h File Reference . . . . .	166
7.28.1 Define Documentation . . . . .	167
7.28.1.1 WSNET_S_DBG . . . . .	167
7.28.1.2 WSNET_S_DBG_DBG . . . . .	167
7.28.1.3 WSNET_S_DBG_EXC . . . . .	167
7.28.1.4 WSNET_S_DBG_OUT . . . . .	167
7.28.1.5 WSNET_S_EXC_DBG . . . . .	167
7.29 worldsens_pkt.h File Reference . . . . .	167
7.29.1 Detailed Description . . . . .	169
7.29.2 Define Documentation . . . . .	169
7.29.2.1 __PACKED__ . . . . .	169
7.29.2.2 WORLDSSENS_MAX_MODELS_SIZE . . . . .	169
7.29.2.3 WORLDSSENS_MAX_PKTLENGTH . . . . .	169
7.29.3 Typedef Documentation . . . . .	169
7.29.3.1 ws_data . . . . .	169
7.29.3.2 ws_frequency . . . . .	169
7.29.3.3 ws_id_node . . . . .	169
7.29.3.4 ws_id_resource . . . . .	169
7.29.3.5 ws_id_rp . . . . .	170
7.29.3.6 ws_id_seq . . . . .	170
7.29.3.7 ws_measure . . . . .	170
7.29.3.8 ws_pkt_type . . . . .	170
7.29.3.9 ws_power . . . . .	170
7.29.3.10 ws_sinnr . . . . .	170
7.29.3.11 ws_time . . . . .	170
7.29.4 Enumeration Type Documentation . . . . .	170
7.29.4.1 woldsens_pkt_type . . . . .	170
7.29.5 Function Documentation . . . . .	171
7.29.5.1 worldsens_packet_dump . . . . .	171
7.29.5.2 worldsens_packet_hton . . . . .	171
7.29.5.3 worldsens_packet_ntoh . . . . .	171





# Chapter 1

## Deprecated List

Global **[get\\_entity\\_links\\_down](#)** (**call\_t \*c**)

Should use [get\\_entity\\_bindings\\_down\(\)](#) instead.

Global **[get\\_entity\\_links\\_down\\_nbr](#)** (**call\_t \*c**)

Should use [get\\_entity\\_bindings\\_down\(\)](#) instead.

Global **[get\\_entity\\_links\\_up](#)** (**call\_t \*c**)

Should use [get\\_entity\\_bindings\\_up\(\)](#) instead.

Global **[get\\_entity\\_links\\_up\\_nbr](#)** (**call\_t \*c**)

Should use [get\\_entity\\_bindings\\_up\(\)](#) instead.



## Chapter 2

# Directory Hierarchy

### 2.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

include . . . . . [11](#)



## Chapter 3

# Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">_angle</a>	An angle in the 3D space . . . . .	13
<a href="#">_antenna_methods</a>	Methods that should be implemented by an antenna model . . . . .	14
<a href="#">_application_methods</a>	Methods that should be implemented by an application model . . . . .	15
<a href="#">_array</a>	An array of integers containing its size . . . . .	16
<a href="#">_call</a>	A parameter that identifies who we are calling and who has called us . . . . .	16
<a href="#">_destination</a>	A packet destination . . . . .	17
<a href="#">_energy_methods</a>	Methods that should be implemented by an energy model . . . . .	18
<a href="#">_entityid</a>	An entity identifier . . . . .	20
<a href="#">_environment_methods</a>	Methods that should be implemented by an environment model . . . . .	20
<a href="#">_event</a>	A scheduler event . . . . .	21
<a href="#">_fading_methods</a>	Methods that should be implemented by a fading model . . . . .	22
<a href="#">_interferences_methods</a>	Methods that should be implemented by a interferences model . . . . .	23
<a href="#">_io_ctl_message</a>	An ioctl message . . . . .	24
<a href="#">_ioctl_message</a>	. . . . .	24
<a href="#">_mac_methods</a>	Methods that should be implemented by a mac model . . . . .	25

<a href="#">_measureid</a>	A measure identifier . . . . .	26
<a href="#">_mobility_methods</a>	Methods that should be implemented by a mobility model . . . . .	26
<a href="#">_model</a>	Information about a model . . . . .	27
<a href="#">_modulation_methods</a>	Methods that should be implemented by a modulation model . . . . .	28
<a href="#">_monitor_methods</a>	Methods that should be implemented by a monitor model . . . . .	29
<a href="#">_nodeid</a>	A node identifier . . . . .	30
<a href="#">_noise_methods</a>	Methods that should be implemented by a noise model . . . . .	30
<a href="#">_packet</a>	A radio packet . . . . .	31
<a href="#">_packetid</a>	A packet identifier . . . . .	35
<a href="#">_param</a>	A parameter for the "init" and "setnode" entity functions . . . . .	35
<a href="#">_position</a>	A position in the 3D space . . . . .	36
<a href="#">_propagation_methods</a>	Methods that should be implemented by a propagation model . . . . .	37
<a href="#">_radio_methods</a>	Methods that should be implemented by a radio model . . . . .	38
<a href="#">_routing_methods</a>	Methods that should be implemented by a routing model . . . . .	41
<a href="#">_shadowing_methods</a>	Methods that should be implemented by a shadowing model . . . . .	42
<a href="#">_worldsens_c_byte_tx</a>	. . . . .	43
<a href="#">_worldsens_c_connect_req</a>	. . . . .	44
<a href="#">_worldsens_c_disconnect</a>	. . . . .	45
<a href="#">_worldsens_c_header</a>	. . . . .	45
<a href="#">_worldsens_c_measure_req</a>	. . . . .	46
<a href="#">_worldsens_c_sync_ack</a>	. . . . .	47
<a href="#">_worldsens_pkt</a>	. . . . .	48
<a href="#">_worldsens_s_backtrack</a>	. . . . .	50
<a href="#">_worldsens_s_byte_rx</a>	. . . . .	51
<a href="#">_worldsens_s_byte_sr_rx</a>	. . . . .	53
<a href="#">_worldsens_s_connect_rsp_nok</a>	. . . . .	54
<a href="#">_worldsens_s_connect_rsp_ok</a>	. . . . .	55
<a href="#">_worldsens_s_header</a>	. . . . .	56
<a href="#">_worldsens_s_kill</a>	. . . . .	57
<a href="#">_worldsens_s_killsim</a>	. . . . .	58
<a href="#">_worldsens_s_measure_rsp</a>	. . . . .	58
<a href="#">_worldsens_s_measure_sr_rsp</a>	. . . . .	59
<a href="#">_worldsens_s_sync_release</a>	. . . . .	60
<a href="#">_worldsens_s_sync_reminder</a>	. . . . .	61
<a href="#">bernoulli_args_s</a>	. . . . .	62

beta_args_s	63
binomial_args_s	63
bivariate_gaussian_args_s	64
cauchy_args_s	65
chisq_args_s	65
exponential_args_s	66
exponential_s	66
exppow_args_s	67
gamma_args_s	68
gaussian_args_s	69
gaussian_tail_args_s	69
geometric_args_s	70
gumbel_t1_args_s	70
gumbel_t2_args_s	71
hyper_geometric_args_s	71
laplace_args_s	72
levy_alpha_stable_s	73
logarithmic_args_s	73
logistic_args_s	74
lognormal_args_s	74
pareto_args_s	75
poisson_args_s	76
qtimer_s	
Timer structure parameters: parameters for next_trigger functions	
(eg: period for periodic timer) conditional_end: pointer to a function	
that return 1 if the timer must be destroyed and 0 otherwise callback-	
_function: function to callback when the timer triggers	76
rayleigh_s	77
rayleigh_tail_s	78
spherical_vector_2d_args_s	79
spherical_vector_3d_args_s	79
uniform_args_s	80
uniform_random_s	81
weibull_args_s	81





## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">antenna.h</a>	Antenna declarations . . . . .	83
<a href="#">battery.h</a>	Battery declarations . . . . .	85
<a href="#">das.h</a>	DAta Structure module declarations . . . . .	88
<a href="#">dbg.h</a>	Dbg declarations . . . . .	92
<a href="#">entity.h</a>	Entity declarations . . . . .	95
<a href="#">hadas.h</a>	Hashed DAta Structure module declarations . . . . .	102
<a href="#">ioctl_message.h</a>	. . . . .	104
<a href="#">log.h</a>	Log declarations . . . . .	106
<a href="#">measure.h</a>	Measure declarations . . . . .	108
<a href="#">medium.h</a>	Medium declarations . . . . .	109
<a href="#">mem_fs.h</a>	Fixed size memory management module declarations . . . . .	112
<a href="#">models.h</a>	Models declarations . . . . .	113
<a href="#">modelutils.h</a>	Utility function declarations . . . . .	119
<a href="#">modulation.h</a>	Modulation declarations . . . . .	121
<a href="#">monitor.h</a>	Monitor declarations . . . . .	122

<a href="#">node.h</a>	Node declarations . . . . .	123
<a href="#">options.h</a>	User options declarations . . . . .	125
<a href="#">packet.h</a>	Packet declarations . . . . .	126
<a href="#">param.h</a>	. . . . .	128
<a href="#">probabilistic_distribution.h</a>	Probabilistic distributions declarations . . . . .	132
<a href="#">radio.h</a>	Radio declarations . . . . .	139
<a href="#">rng.h</a>	Random number generator declarations . . . . .	144
<a href="#">scheduler.h</a>	Scheduler declarations . . . . .	154
<a href="#">sodas.h</a>	SOrted DAta Structure module declarations . . . . .	156
<a href="#">spadas.h</a>	Space PArtitioning DAta Structure module declarations . . . . .	159
<a href="#">timer.h</a>	Generic timer . . . . .	161
<a href="#">types.h</a>	Type declarations . . . . .	163
<a href="#">worldsens_debug.h</a>	. . . . .	166
<a href="#">worldsens_pkt.h</a>	Worldsens packet format . . . . .	167

## Chapter 5

# Directory Documentation

### 5.1 include/ Directory Reference

#### Files

- file [antenna.h](#)  
*Antenna declarations.*
- file [battery.h](#)  
*Battery declarations.*
- file [das.h](#)  
*DAta Structure module declarations.*
- file [dbg.h](#)  
*dbg declarations*
- file [entity.h](#)  
*Entity declarations.*
- file [hadas.h](#)  
*Hashed DAta Structure module declarations.*
- file [ioctl\\_message.h](#)
- file [log.h](#)  
*log declarations*
- file [measure.h](#)  
*Measure declarations.*
- file [medium.h](#)  
*Medium declarations.*
- file [mem\\_fs.h](#)  
*Fixed size memory management module declarations.*
- file [models.h](#)  
*Models declarations.*
- file [modelutils.h](#)  
*Utility function declarations.*

- file [modulation.h](#)  
*Modulation declarations.*
- file [monitor.h](#)  
*Monitor declarations.*
- file [node.h](#)  
*Node declarations.*
- file [options.h](#)  
*User options declarations.*
- file [packet.h](#)  
*Packet declarations.*
- file [param.h](#)
- file [probabilistic\\_distribution.h](#)  
*Probabilistic distributions declarations.*
- file [radio.h](#)  
*Radio declarations.*
- file [rng.h](#)  
*Random number generator declarations.*
- file [scheduler.h](#)  
*Scheduler declarations.*
- file [sodas.h](#)  
*SORTed DATA Structure module declarations.*
- file [spadas.h](#)  
*Space PARTitioning DATA Structure module declarations.*
- file [timer.h](#)  
*Generic timer.*
- file [types.h](#)  
*Type declarations.*
- file [worldsens\\_debug.h](#)
- file [worldsens\\_pkt.h](#)  
*Worldsens packet format.*

## Chapter 6

# Data Structure Documentation

### 6.1 `_angle` Struct Reference

An angle in the 3D space.

```
#include <types.h>
```

#### Data Fields

- double `xy`  
*angle on the xy plane*
- double `z`  
*angle between the xy plane and the z axis*

#### 6.1.1 Detailed Description

An angle in the 3D space.

Should use type `angle_t`.

Definition at line 82 of file `types.h`.

#### 6.1.2 Field Documentation

##### 6.1.2.1 `double _angle::xy`

angle on the xy plane

Definition at line 83 of file `types.h`.

### 6.1.2.2 double \_angle::z

angle between the xy plane and the z axis

Definition at line 84 of file types.h.

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

- [types.h](#)

## 6.2 \_antenna\_methods Struct Reference

Methods that should be implemented by an antenna model.

```
#include <models.h>
```

### Data Fields

- void(\* rx)(call\_t \*c, packet\_t \*packet)
- void(\* cs)(call\_t \*c, packet\_t \*packet)
- double(\* get\_loss)(call\_t \*c)
- angle\_t>(\* get\_angle)(call\_t \*c)
- void(\* set\_angle)(call\_t \*c, angle\_t \*angle)
- double(\* gain\_tx)(call\_t \*c, position\_t \*pos)
- double(\* gain\_rx)(call\_t \*c, position\_t \*pos)

### 6.2.1 Detailed Description

Methods that should be implemented by an antenna model.

Should use type antenna\_methods\_t.

Definition at line 195 of file models.h.

### 6.2.2 Field Documentation

#### 6.2.2.1 void(\* \_antenna\_methods::cs)(call\_t \*c, packet\_t \*packet)

Definition at line 197 of file models.h.

#### 6.2.2.2 double(\* \_antenna\_methods::gain\_rx)(call\_t \*c, position\_t \*pos)

Definition at line 202 of file models.h.

#### 6.2.2.3 double(\* \_antenna\_methods::gain\_tx)(call\_t \*c, position\_t \*pos)

Definition at line 201 of file models.h.

#### 6.2.2.4 `angle_t>(* _antenna_methods::get_angle)(call_t *c)`

Definition at line 199 of file `models.h`.

#### 6.2.2.5 `double(* _antenna_methods::get_loss)(call_t *c)`

Definition at line 198 of file `models.h`.

#### 6.2.2.6 `void(* _antenna_methods::rx)(call_t *c, packet_t *packet)`

Definition at line 196 of file `models.h`.

#### 6.2.2.7 `void(* _antenna_methods::set_angle)(call_t *c, angle_t *angle)`

Definition at line 200 of file `models.h`.

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

- [models.h](#)

## 6.3 `_application_methods` Struct Reference

Methods that should be implemented by an application model.

```
#include <models.h>
```

### Data Fields

- `void(* rx)(call_t *c, packet_t *packet)`

### 6.3.1 Detailed Description

Methods that should be implemented by an application model.

Should use type `application_methods_t`.

Definition at line 289 of file `models.h`.

### 6.3.2 Field Documentation

#### 6.3.2.1 `void(* _application_methods::rx)(call_t *c, packet_t *packet)`

Definition at line 290 of file `models.h`.

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

- [models.h](#)

## 6.4 `_array` Struct Reference

An array of integers containing its size.

```
#include <types.h>
```

### Data Fields

- int [size](#)  
*array size*
- int \* [elts](#)  
*array elements*

#### 6.4.1 Detailed Description

An array of integers containing its size.

Should use type `array_t`.

Definition at line 21 of file `types.h`.

#### 6.4.2 Field Documentation

##### 6.4.2.1 `int* _array::elts`

array elements

Definition at line 23 of file `types.h`.

##### 6.4.2.2 `int _array::size`

array size

Definition at line 22 of file `types.h`.

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

- [types.h](#)

## 6.5 `_call` Struct Reference

A parameter that identifies who we are calling and who has called us.

```
#include <types.h>
```



## Data Fields

- [entityid\\_t entity](#)  
*the called entity id*
- [nodeid\\_t node](#)  
*the called node id*
- [entityid\\_t from](#)  
*the entity that made the call*

### 6.5.1 Detailed Description

A parameter that identifies who we are calling and who has called us.

Kind of a self pointer. Should use type `call_t`.

Definition at line 106 of file `types.h`.

### 6.5.2 Field Documentation

#### 6.5.2.1 `entityid_t_call::entity`

the called entity id

Definition at line 107 of file `types.h`.

#### 6.5.2.2 `entityid_t_call::from`

the entity that made the call

Definition at line 109 of file `types.h`.

#### 6.5.2.3 `nodeid_t_call::node`

the called node id

Definition at line 108 of file `types.h`.

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

- [types.h](#)

## 6.6 \_destination Struct Reference

A packet destination.

```
#include <types.h>
```

## Data Fields

- [nodeid\\_t id](#)  
*the destination node id*
- [position\\_t position](#)  
*the destination position*

### 6.6.1 Detailed Description

A packet destination.

May be a node address or a geographical position. Should use type `destination_t`.

Definition at line 94 of file `types.h`.

### 6.6.2 Field Documentation

#### 6.6.2.1 `nodeid_t _destination::id`

the destination node id

Definition at line 95 of file `types.h`.

#### 6.6.2.2 `position_t _destination::position`

the destination position

Definition at line 96 of file `types.h`.

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

- [types.h](#)

## 6.7 `_energy_methods` Struct Reference

Methods that should be implemented by an energy model.

```
#include <models.h>
```

## Data Fields

- `void(* consume\_tx)(call\_t *c, uint64_t duration, double txdBm)`
- `void(* consume\_rx)(call\_t *c, uint64_t duration)`
- `void(* consume\_idle)(call\_t *c, uint64_t duration)`
- `void(* consume)(call\_t *c, double energy)`
- `double(* energy\_consumed)(call\_t *c)`
- `double(* energy\_remaining)(call\_t *c)`
- `double(* energy\_status)(call\_t *c)`

### 6.7.1 Detailed Description

Methods that should be implemented by an energy model.

Should use type `energy_methods_t`.

Definition at line 176 of file `models.h`.

### 6.7.2 Field Documentation

**6.7.2.1** `void(*_energy_methods::consume)(call_t *c, double energy)`

Definition at line 180 of file `models.h`.

**6.7.2.2** `void(*_energy_methods::consume_idle)(call_t *c, uint64_t duration)`

Definition at line 179 of file `models.h`.

**6.7.2.3** `void(*_energy_methods::consume_rx)(call_t *c, uint64_t duration)`

Definition at line 178 of file `models.h`.

**6.7.2.4** `void(*_energy_methods::consume_tx)(call_t *c, uint64_t duration, double txdBm)`

Definition at line 177 of file `models.h`.

**6.7.2.5** `double(*_energy_methods::energy_consumed)(call_t *c)`

Definition at line 181 of file `models.h`.

**6.7.2.6** `double(*_energy_methods::energy_remaining)(call_t *c)`

Definition at line 182 of file `models.h`.

**6.7.2.7** `double(*_energy_methods::energy_status)(call_t *c)`

Definition at line 183 of file `models.h`.

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

- [models.h](#)

## 6.8 `_entityid` Struct Reference

An entity identifier.

```
#include <types.h>
```

### 6.8.1 Detailed Description

An entity identifier.

Should use type `entityid_t`.

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

- [types.h](#)

## 6.9 `_environment_methods` Struct Reference

Methods that should be implemented by an environment model.

```
#include <models.h>
```

### Data Fields

- `void(* read\_measure)(call\_t *c, measureid\_t measure, double *value)`

### 6.9.1 Detailed Description

Methods that should be implemented by an environment model.

Should use type `environment_methods_t`.

Definition at line 135 of file `models.h`.

### 6.9.2 Field Documentation

6.9.2.1 `void(* \_environment\_methods::read\_measure)(call\_t *c, measureid\_t measure, double *value)`

Definition at line 136 of file `models.h`.

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

- [models.h](#)

## 6.10 `_event` Struct Reference

A scheduler event.

```
#include <scheduler.h>
```

### Data Fields

- `uint64_t clock`
- `int priority`
- `int id`
- `union {`
  - `struct {`
    - `call_t call`
    - `callback_t callback`
    - `void * arg`
  - `} cb`
  - `struct {`
    - `packet_t * packet`
    - `call_t call`
  - `} rx`
  - `nodeid_t nodeid`
- `} u`

### 6.10.1 Detailed Description

A scheduler event.

Should use type `event_t`.

Definition at line 19 of file `scheduler.h`.

### 6.10.2 Field Documentation

#### 6.10.2.1 `void* _event::arg`

Definition at line 28 of file `scheduler.h`.

#### 6.10.2.2 `call_t _event::call`

Definition at line 26 of file `scheduler.h`.

#### 6.10.2.3 `callback_t _event::callback`

Definition at line 27 of file `scheduler.h`.

6.10.2.4 `struct { ... } _event::cb`

6.10.2.5 `uint64_t _event::clock`

Definition at line 20 of file scheduler.h.

6.10.2.6 `int _event::id`

Definition at line 22 of file scheduler.h.

6.10.2.7 `nodeid_t _event::nodeid`

Definition at line 34 of file scheduler.h.

6.10.2.8 `packet_t* _event::packet`

Definition at line 31 of file scheduler.h.

6.10.2.9 `int _event::priority`

Definition at line 21 of file scheduler.h.

6.10.2.10 `struct { ... } _event::rx`

6.10.2.11 `union { ... } _event::u`

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

- [scheduler.h](#)

## 6.11 `_fading_methods` Struct Reference

Methods that should be implemented by a fading model.

```
#include <models.h>
```

### Data Fields

- `double(* fading)(call_t *c, packet_t *packet, nodeid_t src, nodeid_t dst, double rxdBm)`

### 6.11.1 Detailed Description

Methods that should be implemented by a fading model.

Should use type `fading_methods_t`.

Definition at line 69 of file `models.h`.

### 6.11.2 Field Documentation

6.11.2.1 `double(*_fading_methods::fading)(call_t *c, packet_t *packet, nodeid_t src, nodeid_t dst, double rxdBm)`

Definition at line 70 of file `models.h`.

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

- [models.h](#)

## 6.12 `_interferences_methods` Struct Reference

Methods that should be implemented by a interferences model.

```
#include <models.h>
```

### Data Fields

- `double(* interfere )(call_t *c, int channel0, int channel1)`

### 6.12.1 Detailed Description

Methods that should be implemented by a interferences model.

Should use type `interferences_methods_t`.

Definition at line 95 of file `models.h`.

### 6.12.2 Field Documentation

6.12.2.1 `double(*_interferences_methods::interfere)(call_t *c, int channel0, int channel1)`

Definition at line 96 of file `models.h`.

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

- [models.h](#)

## 6.13 `_ioctl_message` Struct Reference

An ioctl message.

```
#include <types.h>
```

### 6.13.1 Detailed Description

An ioctl message.

Should use type `ioctl_message_t`.

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

- [types.h](#)

## 6.14 `_ioctl_message` Struct Reference

```
#include <types.h>
```

### Data Fields

- int [type](#)
- int [size](#)
- int [real\\_size](#)
- void \* [body](#)

### 6.14.1 Detailed Description

Definition at line 159 of file `types.h`.

### 6.14.2 Field Documentation

#### 6.14.2.1 `void* _ioctl_message::body`

Definition at line 167 of file `types.h`.

#### 6.14.2.2 `int _ioctl_message::real_size`

Definition at line 165 of file `types.h`.

#### 6.14.2.3 `int _ioctl_message::size`

Definition at line 163 of file `types.h`.



6.14.2.4 `int _ioctl_message::type`

Definition at line 161 of file `types.h`.

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

- [types.h](#)

## 6.15 `_mac_methods` Struct Reference

Methods that should be implemented by a mac model.

```
#include <models.h>
```

### Data Fields

- `void(* rx)(call_t *c, packet_t *packet)`
- `void(* tx)(call_t *c, packet_t *packet)`
- `int(* set_header)(call_t *c, packet_t *packet, destination_t *dst)`
- `int(* get_header_size)(call_t *c)`
- `int(* get_header_real_size)(call_t *c)`

### 6.15.1 Detailed Description

Methods that should be implemented by a mac model.

Should use type `mac_methods_t`.

Definition at line 251 of file `models.h`.

### 6.15.2 Field Documentation

6.15.2.1 `int(* _mac_methods::get_header_real_size)(call_t *c)`

Definition at line 257 of file `models.h`.

6.15.2.2 `int(* _mac_methods::get_header_size)(call_t *c)`

Definition at line 255 of file `models.h`.

6.15.2.3 `void(* _mac_methods::rx)(call_t *c, packet_t *packet)`

Definition at line 252 of file `models.h`.

6.15.2.4 `int(*_mac_methods::set_header)(call_t *c, packet_t *packet, destination_t *dst)`

Definition at line 254 of file models.h.

6.15.2.5 `void(*_mac_methods::tx)(call_t *c, packet_t *packet)`

Definition at line 253 of file models.h.

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

- [models.h](#)

## 6.16 `_measureid` Struct Reference

A measure identifier.

```
#include <types.h>
```

### 6.16.1 Detailed Description

A measure identifier.

Should use type `measureid_t`.

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

- [types.h](#)

## 6.17 `_mobility_methods` Struct Reference

Methods that should be implemented by a mobility model.

```
#include <models.h>
```

### Data Fields

- `void(* update\_position )(call_t *c)`

### 6.17.1 Detailed Description

Methods that should be implemented by a mobility model.

Should use type `mobility_methods_t`.

Definition at line 163 of file models.h.

### 6.17.2 Field Documentation

#### 6.17.2.1 `void(*_mobility_methods::update_position)(call_t *c)`

Definition at line 164 of file `models.h`.

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

- [models.h](#)

## 6.18 `_model` Struct Reference

Information about a model.

```
#include <models.h>
```

### Data Fields

- `char * online`
- `char * author`
- `char * version`
- `int type`
- `struct {`
  - `char ** exported`
  - `int count``} measure`

### 6.18.1 Detailed Description

Information about a model.

Should use type `model_t`.

Definition at line 36 of file `models.h`.

### 6.18.2 Field Documentation

#### 6.18.2.1 `char* _model::author`

Definition at line 38 of file `models.h`.

#### 6.18.2.2 `int _model::count`

Definition at line 43 of file `models.h`.

### 6.18.2.3 `char** _model::exported`

Definition at line 42 of file models.h.

### 6.18.2.4 `struct { ... } _model::measure`

### 6.18.2.5 `char* _model::online`

Definition at line 37 of file models.h.

### 6.18.2.6 `int _model::type`

Definition at line 40 of file models.h.

### 6.18.2.7 `char* _model::version`

Definition at line 39 of file models.h.

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

- [models.h](#)

## 6.19 `_modulation_methods` Struct Reference

Methods that should be implemented by a modulation model.

```
#include <models.h>
```

### Data Fields

- `double(* modulate)(call\_t *c, double snr)`
- `int(* bit\_per\_symbol)(call\_t *c)`

### 6.19.1 Detailed Description

Methods that should be implemented by a modulation model.

Should use type `modulation_methods_t`.

Definition at line 121 of file models.h.

### 6.19.2 Field Documentation

### 6.19.2.1 `int(*_modulation_methods::bit_per_symbol)(call_t *c)`

Definition at line 123 of file `models.h`.

### 6.19.2.2 `double(*_modulation_methods::modulate)(call_t *c, double snr)`

Definition at line 122 of file `models.h`.

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

- [models.h](#)

## 6.20 `_monitor_methods` Struct Reference

Methods that should be implemented by a monitor model.

```
#include <models.h>
```

### Data Fields

- `void(* monitor_death )(call_t *c)`
- `void(* monitor_event )(call_t *c)`
- `void(* monitor_register_callback )(call_t *c, callback_t callback, void *arg)`

### 6.20.1 Detailed Description

Methods that should be implemented by a monitor model.

Should use type `monitor_methods_t`.

Definition at line 148 of file `models.h`.

### 6.20.2 Field Documentation

#### 6.20.2.1 `void(*_monitor_methods::monitor_death)(call_t *c)`

Definition at line 149 of file `models.h`.

#### 6.20.2.2 `void(*_monitor_methods::monitor_event)(call_t *c)`

Definition at line 150 of file `models.h`.

6.20.2.3 `void(* _monitor_methods::monitor_register_callback)(call_t *c, callback_t callback, void *arg)`

Definition at line 151 of file models.h.

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

- [models.h](#)

## 6.21 `_nodeid` Struct Reference

A node identifier.

```
#include <types.h>
```

### 6.21.1 Detailed Description

A node identifier.

Should use type `nodeid_t`.

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

- [types.h](#)

## 6.22 `_noise_methods` Struct Reference

Methods that should be implemented by a noise model.

```
#include <models.h>
```

### Data Fields

- `double(* noise )(call_t *c, nodeid\_t node, int channel)`

### 6.22.1 Detailed Description

Methods that should be implemented by a noise model.

Should use type `noise_methods_t`.

Definition at line 108 of file models.h.

### 6.22.2 Field Documentation

6.22.2.1 `double(*_noise_methods::noise)(call_t *c, nodeid_t node, int channel)`

Definition at line 109 of file `models.h`.

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

- [models.h](#)

## 6.23 `_packet` Struct Reference

A radio packet.

```
#include <types.h>
```

### Data Fields

- [packetid\\_t id](#)  
*the packet id*
- `int size`  
*size of the packet data*
- `int real_size`  
*real size of the packet data*
- `int type`  
*type of the packet (for multistandard nodes)*
- `uint64_t clock0`  
*packet rx start*
- `uint64_t clock1`  
*packet rx end*
- `uint64_t duration`  
*packet tx/rx duration*
- `nodeid_t node`  
*node that has created the packet*
- `entityid_t antenna`  
*antenna that has emitted the packet*
- `int worldsens_mod`  
*worldsens modulation id (wsim radio modulation id)*
- `double worldsens_freq`  
*worldsens radio frequency*
- `double txdBm`  
*tx power in dBm*
- `int channel`
- `entityid_t modulation`  
*modulation entity*
- `uint64_t Tb`

- radio bandwidth: time to send a bit*
- double [PER](#)  
*packet error rate*
- double [rxdBm](#)  
*rx power in dBm*
- double [rxmW](#)  
*rx power in mW*
- double \* [noise\\_mW](#)  
*packet noise in mW*
- double \* [ber](#)  
*packet ber*
- char \* [data](#)  
*packet data*

### 6.23.1 Detailed Description

A radio packet.

Should use type `packet_t`.

Definition at line 119 of file `types.h`.

### 6.23.2 Field Documentation

#### 6.23.2.1 `entityid_t_packet::antenna`

antenna that has emitted the packet

Definition at line 130 of file `types.h`.

#### 6.23.2.2 `double*_packet::ber`

packet ber

Definition at line 146 of file `types.h`.

#### 6.23.2.3 `int_packet::channel`

Definition at line 138 of file `types.h`.

#### 6.23.2.4 `uint64_t_packet::clock0`

packet rx start

Definition at line 125 of file `types.h`.



**6.23.2.5 uint64\_t \_packet::clock1**

packet rx end

Definition at line 126 of file types.h.

**6.23.2.6 char\* \_packet::data**

packet data

Definition at line 148 of file types.h.

**6.23.2.7 uint64\_t \_packet::duration**

packet tx/rx duration

Definition at line 127 of file types.h.

**6.23.2.8 packetid\_t \_packet::id**

the packet id

Definition at line 120 of file types.h.

**6.23.2.9 entityid\_t \_packet::modulation**

modulation entity

Definition at line 139 of file types.h.

**6.23.2.10 nodeid\_t \_packet::node**

node that has created the packet

Definition at line 129 of file types.h.

**6.23.2.11 double\* \_packet::noise\_mW**

packet noise in mW

Definition at line 145 of file types.h.

**6.23.2.12 double \_packet::PER**

packet error rate

Definition at line 142 of file types.h.

**6.23.2.13 int \_packet::real\_size**

real size of the packet data

Definition at line 122 of file types.h.

**6.23.2.14 double \_packet::rxdBm**

rx power in dBm

Definition at line 143 of file types.h.

**6.23.2.15 double \_packet::rxmW**

rx power in mW

Definition at line 144 of file types.h.

**6.23.2.16 int \_packet::size**

size of the packet data

Definition at line 121 of file types.h.

**6.23.2.17 uint64\_t \_packet::Tb**

radio bandwidth: time to send a bit

Definition at line 140 of file types.h.

**6.23.2.18 double \_packet::txdBm**

tx power in dBm

Definition at line 137 of file types.h.

**6.23.2.19 int \_packet::type**

type of the packet (for multistandard nodes)

Definition at line 123 of file types.h.

**6.23.2.20 double \_packet::worldsens\_freq**

worldsens radio frequency

Definition at line 134 of file types.h.

6.23.2.21 `int _packet::worldsens_mod`

worldsens modulation id (wsim radio modulation id)

Definition at line 133 of file `types.h`.

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

- [types.h](#)

## 6.24 `_packetid` Struct Reference

A packet identifier.

```
#include <types.h>
```

### 6.24.1 Detailed Description

A packet identifier.

Should use type `packetid_t`.

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

- [types.h](#)

## 6.25 `_param` Struct Reference

A parameter for the "init" and "setnode" entity functions.

```
#include <types.h>
```

### Data Fields

- `char * key`  
*the parameter key*
- `char * value`  
*the parameter value*

### 6.25.1 Detailed Description

A parameter for the "init" and "setnode" entity functions.

Should use type `param_t`.

Definition at line 178 of file `types.h`.

## 6.25.2 Field Documentation

### 6.25.2.1 `char* _param::key`

the parameter key

Definition at line 179 of file types.h.

### 6.25.2.2 `char* _param::value`

the parameter value

Definition at line 180 of file types.h.

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

- [types.h](#)

## 6.26 `_position` Struct Reference

A position in the 3D space.

```
#include <types.h>
```

### Data Fields

- double [x](#)  
*x position*
- double [y](#)  
*y position*
- double [z](#)  
*z position*

### 6.26.1 Detailed Description

A position in the 3D space.

Should use type `position_t`.

Definition at line 69 of file types.h.

### 6.26.2 Field Documentation

#### 6.26.2.1 `double _position::x`

x position

Definition at line 70 of file types.h.

### 6.26.2.2 `double _position::y`

y position

Definition at line 71 of file `types.h`.

### 6.26.2.3 `double _position::z`

z position

Definition at line 72 of file `types.h`.

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

- [types.h](#)

## 6.27 `_propagation_methods` Struct Reference

Methods that should be implemented by a propagation model.

```
#include <models.h>
```

### Data Fields

- `double(* propagation )(call\_t *c, packet\_t *packet, nodeid\_t src, nodeid\_t dst, double rxdBm)`

### 6.27.1 Detailed Description

Methods that should be implemented by a propagation model.

Should use type `propagation_methods_t`.

Definition at line 56 of file `models.h`.

### 6.27.2 Field Documentation

#### 6.27.2.1 `double(* \_propagation\_methods::propagation)(call\_t *c, packet\_t *packet, nodeid\_t src, nodeid\_t dst, double rxdBm)`

Definition at line 57 of file `models.h`.

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

- [models.h](#)

## 6.28 \_radio\_methods Struct Reference

Methods that should be implemented by a radio model.

```
#include <models.h>
```

### Data Fields

- void(\* rx)(call\_t \*c, packet\_t \*packet)
- void(\* tx)(call\_t \*c, packet\_t \*packet)
- int(\* set\_header)(call\_t \*c, packet\_t \*packet, destination\_t \*dst)
- int(\* get\_header\_size)(call\_t \*c)
- int(\* get\_header\_real\_size)(call\_t \*c)
- void(\* tx\_end)(call\_t \*c, packet\_t \*packet)
- void(\* cs)(call\_t \*c, packet\_t \*packet)
- double(\* get\_noise)(call\_t \*c)
- double(\* get\_cs)(call\_t \*c)
- double(\* get\_power)(call\_t \*c)
- void(\* set\_power)(call\_t \*c, double power)
- int(\* get\_channel)(call\_t \*c)
- void(\* set\_channel)(call\_t \*c, int channel)
- entityid\_t(\* get\_modulation)(call\_t \*c)
- void(\* set\_modulation)(call\_t \*c, entityid\_t modulation)
- uint64\_t(\* get\_Tb)(call\_t \*c)
- uint64\_t(\* get\_Ts)(call\_t \*c)
- void(\* set\_Ts)(call\_t \*c, uint64\_t Ts)
- double(\* get\_sensibility)(call\_t \*c)
- void(\* set\_sensibility)(call\_t \*c, double sensibility)
- void(\* sleep)(call\_t \*c)
- void(\* wakeup)(call\_t \*c)
- int(\* get\_modulation\_bit\_per\_symbol)(call\_t \*c)

### 6.28.1 Detailed Description

Methods that should be implemented by a radio model.

Should use type radio\_methods\_t.

Definition at line 214 of file models.h.

### 6.28.2 Field Documentation

#### 6.28.2.1 void(\* \_radio\_methods::cs)(call\_t \*c, packet\_t \*packet)

Definition at line 223 of file models.h.

**6.28.2.2** `int(*_radio_methods::get_channel)(call_t *c)`

Definition at line 228 of file models.h.

**6.28.2.3** `double(*_radio_methods::get_cs)(call_t *c)`

Definition at line 225 of file models.h.

**6.28.2.4** `int(*_radio_methods::get_header_real_size)(call_t *c)`

Definition at line 220 of file models.h.

**6.28.2.5** `int(*_radio_methods::get_header_size)(call_t *c)`

Definition at line 219 of file models.h.

**6.28.2.6** `entityid_t(*_radio_methods::get_modulation)(call_t *c)`

Definition at line 230 of file models.h.

**6.28.2.7** `int(*_radio_methods::get_modulation_bit_per_symbol)(call_t *c)`

Definition at line 239 of file models.h.

**6.28.2.8** `double(*_radio_methods::get_noise)(call_t *c)`

Definition at line 224 of file models.h.

**6.28.2.9** `double(*_radio_methods::get_power)(call_t *c)`

Definition at line 226 of file models.h.

**6.28.2.10** `double(*_radio_methods::get_sensibility)(call_t *c)`

Definition at line 235 of file models.h.

**6.28.2.11** `uint64_t(*_radio_methods::get_Tb)(call_t *c)`

Definition at line 232 of file models.h.

**6.28.2.12** `uint64_t(*_radio_methods::get_Ts)(call_t *c)`

Definition at line 233 of file models.h.

**6.28.2.13** `void(*_radio_methods::rx)(call_t *c, packet_t *packet)`

Definition at line 215 of file models.h.

**6.28.2.14** `void(*_radio_methods::set_channel)(call_t *c, int channel)`

Definition at line 229 of file models.h.

**6.28.2.15** `int(*_radio_methods::set_header)(call_t *c, packet_t *packet,  
destination_t *dst)`

Definition at line 218 of file models.h.

**6.28.2.16** `void(*_radio_methods::set_modulation)(call_t *c, entityid_t modulation)`

Definition at line 231 of file models.h.

**6.28.2.17** `void(*_radio_methods::set_power)(call_t *c, double power)`

Definition at line 227 of file models.h.

**6.28.2.18** `void(*_radio_methods::set_sensibility)(call_t *c, double sensibility)`

Definition at line 236 of file models.h.

**6.28.2.19** `void(*_radio_methods::set_Ts)(call_t *c, uint64_t Ts)`

Definition at line 234 of file models.h.

**6.28.2.20** `void(*_radio_methods::sleep)(call_t *c)`

Definition at line 237 of file models.h.

**6.28.2.21** `void(*_radio_methods::tx)(call_t *c, packet_t *packet)`

Definition at line 216 of file models.h.



6.28.2.22 `void(*_radio_methods::tx_end)(call_t *c, packet_t *packet)`

Definition at line 222 of file `models.h`.

6.28.2.23 `void(*_radio_methods::wakeup)(call_t *c)`

Definition at line 238 of file `models.h`.

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

- [models.h](#)

## 6.29 `_routing_methods` Struct Reference

Methods that should be implemented by a routing model.

```
#include <models.h>
```

### Data Fields

- `void(* rx)(call_t *c, packet_t *packet)`
- `void(* tx)(call_t *c, packet_t *packet)`
- `int(* set_header)(call_t *c, packet_t *packet, destination_t *dst)`
- `int(* get_header_size)(call_t *c)`
- `int(* get_header_real_size)(call_t *c)`

### 6.29.1 Detailed Description

Methods that should be implemented by a routing model.

Should use type `routing_methods_t`.

Definition at line 270 of file `models.h`.

### 6.29.2 Field Documentation

6.29.2.1 `int(*_routing_methods::get_header_real_size)(call_t *c)`

Definition at line 276 of file `models.h`.

6.29.2.2 `int(*_routing_methods::get_header_size)(call_t *c)`

Definition at line 274 of file `models.h`.

6.29.2.3 `void(*_routing_methods::rx)(call_t *c, packet_t *packet)`

Definition at line 271 of file models.h.

6.29.2.4 `int(*_routing_methods::set_header)(call_t *c, packet_t *packet, destination_t *dst)`

Definition at line 273 of file models.h.

6.29.2.5 `void(*_routing_methods::tx)(call_t *c, packet_t *packet)`

Definition at line 272 of file models.h.

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

- [models.h](#)

## 6.30 `_shadowing_methods` Struct Reference

Methods that should be implemented by a shadowing model.

```
#include <models.h>
```

### Data Fields

- `double(*_shadowing)(call_t *c, packet_t *packet, nodeid_t src, nodeid_t dst, double rxdBm)`

#### 6.30.1 Detailed Description

Methods that should be implemented by a shadowing model.

Should use type `shadowing_methods_t`.

Definition at line 82 of file models.h.

#### 6.30.2 Field Documentation

6.30.2.1 `double(*_shadowing_methods::shadowing)(call_t *c, packet_t *packet, nodeid_t src, nodeid_t dst, double rxdBm)`

Definition at line 83 of file models.h.

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

- [models.h](#)

## 6.31 `_worldsens_c_byte_tx` Struct Reference

```
#include <worldsens_pkt.h>
```

### Data Fields

- `ws_pkt_type` type
- `ws_id_node` node\_id
- `ws_id_resource` antenna\_id
- `ws_id_resource` wsnet\_mod\_id
- `ws_id_resource` wsim\_mod\_id
- `ws_frequency` freq
- `ws_power` power\_dbm
- `ws_time` duration
- `ws_time` period
- `ws_data` data

#### 6.31.1 Detailed Description

Definition at line 95 of file `worldsens_pkt.h`.

#### 6.31.2 Field Documentation

##### 6.31.2.1 `ws_id_resource _worldsens_c_byte_tx::antenna_id`

Definition at line 98 of file `worldsens_pkt.h`.

##### 6.31.2.2 `ws_data _worldsens_c_byte_tx::data`

Definition at line 105 of file `worldsens_pkt.h`.

##### 6.31.2.3 `ws_time _worldsens_c_byte_tx::duration`

Definition at line 103 of file `worldsens_pkt.h`.

##### 6.31.2.4 `ws_frequency _worldsens_c_byte_tx::freq`

Definition at line 101 of file `worldsens_pkt.h`.

##### 6.31.2.5 `ws_id_node _worldsens_c_byte_tx::node_id`

Definition at line 97 of file `worldsens_pkt.h`.

#### 6.31.2.6 `ws_time_worldsens_c_byte_tx::period`

Definition at line 104 of file `worldsens_pkt.h`.

#### 6.31.2.7 `ws_power_worldsens_c_byte_tx::power_dbm`

Definition at line 102 of file `worldsens_pkt.h`.

#### 6.31.2.8 `ws_pkt_type_worldsens_c_byte_tx::type`

Definition at line 96 of file `worldsens_pkt.h`.

#### 6.31.2.9 `ws_id_resource_worldsens_c_byte_tx::wsim_mod_id`

Definition at line 100 of file `worldsens_pkt.h`.

#### 6.31.2.10 `ws_id_resource_worldsens_c_byte_tx::wsnet_mod_id`

Definition at line 99 of file `worldsens_pkt.h`.

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

- [worldsens\\_pkt.h](#)

## 6.32 `_worldsens_c_connect_req` Struct Reference

```
#include <worldsens_pkt.h>
```

### Data Fields

- [ws\\_pkt\\_type](#) type
- [ws\\_id\\_node](#) node\_id

#### 6.32.1 Detailed Description

Definition at line 84 of file `worldsens_pkt.h`.

#### 6.32.2 Field Documentation

##### 6.32.2.1 `ws_id_node_worldsens_c_connect_req::node_id`

Definition at line 86 of file `worldsens_pkt.h`.

#### 6.32.2.2 `ws_pkt_type _worldsens_c_connect_req::type`

Definition at line 85 of file `worldsens_pkt.h`.

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

- [worldsens\\_pkt.h](#)

### 6.33 `_worldsens_c_disconnect` Struct Reference

```
#include <worldsens_pkt.h>
```

#### Data Fields

- [ws\\_pkt\\_type](#) type
- [ws\\_id\\_node](#) node\_id

#### 6.33.1 Detailed Description

Definition at line 115 of file `worldsens_pkt.h`.

#### 6.33.2 Field Documentation

##### 6.33.2.1 `ws_id_node _worldsens_c_disconnect::node_id`

Definition at line 117 of file `worldsens_pkt.h`.

##### 6.33.2.2 `ws_pkt_type _worldsens_c_disconnect::type`

Definition at line 116 of file `worldsens_pkt.h`.

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

- [worldsens\\_pkt.h](#)

### 6.34 `_worldsens_c_header` Struct Reference

```
#include <worldsens_pkt.h>
```

#### Data Fields

- [ws\\_pkt\\_type](#) type
- [ws\\_id\\_node](#) id

### 6.34.1 Detailed Description

Definition at line 79 of file worldsens\_pkt.h.

### 6.34.2 Field Documentation

#### 6.34.2.1 ws\_id\_node\_worldsens\_c\_header::id

Definition at line 81 of file worldsens\_pkt.h.

#### 6.34.2.2 ws\_pkt\_type\_worldsens\_c\_header::type

Definition at line 80 of file worldsens\_pkt.h.

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

- [worldsens\\_pkt.h](#)

## 6.35 \_worldsens\_c\_measure\_req Struct Reference

```
#include <worldsens_pkt.h>
```

### Data Fields

- [ws\\_pkt\\_type](#) type
- [ws\\_id\\_node](#) node\_id
- [ws\\_id\\_resource](#) measure\_id
- [ws\\_time](#) period

### 6.35.1 Detailed Description

Definition at line 108 of file worldsens\_pkt.h.

### 6.35.2 Field Documentation

#### 6.35.2.1 ws\_id\_resource\_worldsens\_c\_measure\_req::measure\_id

Definition at line 111 of file worldsens\_pkt.h.

#### 6.35.2.2 ws\_id\_node\_worldsens\_c\_measure\_req::node\_id

Definition at line 110 of file worldsens\_pkt.h.

### 6.35.2.3 ws\_time\_worldsens\_c\_measure\_req::period

Definition at line 112 of file worldsens\_pkt.h.

### 6.35.2.4 ws\_pkt\_type\_worldsens\_c\_measure\_req::type

Definition at line 109 of file worldsens\_pkt.h.

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

- [worldsens\\_pkt.h](#)

## 6.36 \_worldsens\_c\_sync\_ack Struct Reference

```
#include <worldsens_pkt.h>
```

### Data Fields

- [ws\\_pkt\\_type](#) type
- [ws\\_id\\_node](#) node\_id
- [ws\\_id\\_rp](#) rp\_id

### 6.36.1 Detailed Description

Definition at line 89 of file worldsens\_pkt.h.

### 6.36.2 Field Documentation

#### 6.36.2.1 ws\_id\_node\_worldsens\_c\_sync\_ack::node\_id

Definition at line 91 of file worldsens\_pkt.h.

#### 6.36.2.2 ws\_id\_rp\_worldsens\_c\_sync\_ack::rp\_id

Definition at line 92 of file worldsens\_pkt.h.

#### 6.36.2.3 ws\_pkt\_type\_worldsens\_c\_sync\_ack::type

Definition at line 90 of file worldsens\_pkt.h.

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

- [worldsens\\_pkt.h](#)

## 6.37 `_worldsens_pkt` Union Reference

```
#include <worldsens_pkt.h>
```

### Data Fields

- struct [\\_worldsens\\_c\\_header](#) `c_header`
- struct [\\_worldsens\\_s\\_header](#) `s_header`
- struct [\\_worldsens\\_c\\_connect\\_req](#) `cnx_req`
- struct [\\_worldsens\\_s\\_connect\\_rsp\\_ok](#) `cnx_rsp_ok`
- struct [\\_worldsens\\_s\\_connect\\_rsp\\_nok](#) `cnx_rsp_nok`
- struct [\\_worldsens\\_c\\_byte\\_tx](#) `byte_tx`
- struct [\\_worldsens\\_s\\_byte\\_rx](#) `byte_rx`
- struct [\\_worldsens\\_s\\_byte\\_sr\\_rx](#) `byte_sr_rx`
- struct [\\_worldsens\\_c\\_sync\\_ack](#) `sync_ack`
- struct [\\_worldsens\\_s\\_sync\\_release](#) `sync_release`
- struct [\\_worldsens\\_s\\_sync\\_reminder](#) `sync_reminder`
- struct [\\_worldsens\\_s\\_backtrack](#) `bktrk`
- struct [\\_worldsens\\_c\\_measure\\_req](#) `measure_req`
- struct [\\_worldsens\\_s\\_measure\\_rsp](#) `measure_rsp`
- struct [\\_worldsens\\_s\\_measure\\_sr\\_rsp](#) `measure_sr_rsp`
- struct [\\_worldsens\\_c\\_disconnect](#) `disconnect`
- struct [\\_worldsens\\_s\\_killsim](#) `killsim`
- struct [\\_worldsens\\_s\\_kill](#) `kill`

### 6.37.1 Detailed Description

Definition at line 231 of file `worldsens_pkt.h`.

### 6.37.2 Field Documentation

#### 6.37.2.1 `struct _worldsens_s_backtrack _worldsens_pkt::bktrk`

Definition at line 247 of file `worldsens_pkt.h`.

#### 6.37.2.2 `struct _worldsens_s_byte_rx _worldsens_pkt::byte_rx`

Definition at line 240 of file `worldsens_pkt.h`.

#### 6.37.2.3 `struct _worldsens_s_byte_sr_rx _worldsens_pkt::byte_sr_rx`

Definition at line 241 of file `worldsens_pkt.h`.



**6.37.2.4 struct \_worldsens\_c\_byte\_tx \_worldsens\_pkt::byte\_tx**

Definition at line 239 of file worldsens\_pkt.h.

**6.37.2.5 struct \_worldsens\_c\_header \_worldsens\_pkt::c\_header**

Definition at line 232 of file worldsens\_pkt.h.

**6.37.2.6 struct \_worldsens\_c\_connect\_req \_worldsens\_pkt::cnx\_req**

Definition at line 235 of file worldsens\_pkt.h.

**6.37.2.7 struct \_worldsens\_s\_connect\_rsp\_nok \_worldsens\_pkt::cnx\_rsp\_nok**

Definition at line 237 of file worldsens\_pkt.h.

**6.37.2.8 struct \_worldsens\_s\_connect\_rsp\_ok \_worldsens\_pkt::cnx\_rsp\_ok**

Definition at line 236 of file worldsens\_pkt.h.

**6.37.2.9 struct \_worldsens\_c\_disconnect \_worldsens\_pkt::disconnect**

Definition at line 253 of file worldsens\_pkt.h.

**6.37.2.10 struct \_worldsens\_s\_kill \_worldsens\_pkt::kill**

Definition at line 255 of file worldsens\_pkt.h.

**6.37.2.11 struct \_worldsens\_s\_killsim \_worldsens\_pkt::killsim**

Definition at line 254 of file worldsens\_pkt.h.

**6.37.2.12 struct \_worldsens\_c\_measure\_req \_worldsens\_pkt::measure\_req**

Definition at line 249 of file worldsens\_pkt.h.

**6.37.2.13 struct \_worldsens\_s\_measure\_rsp \_worldsens\_pkt::measure\_rsp**

Definition at line 250 of file worldsens\_pkt.h.

#### 6.37.2.14 `struct _worldsens_s_measure_sr_rsp _worldsens_pkt::measure_sr_rsp`

Definition at line 251 of file `worldsens_pkt.h`.

#### 6.37.2.15 `struct _worldsens_s_header _worldsens_pkt::s_header`

Definition at line 233 of file `worldsens_pkt.h`.

#### 6.37.2.16 `struct _worldsens_c_sync_ack _worldsens_pkt::sync_ack`

Definition at line 243 of file `worldsens_pkt.h`.

#### 6.37.2.17 `struct _worldsens_s_sync_release _worldsens_pkt::sync_release`

Definition at line 244 of file `worldsens_pkt.h`.

#### 6.37.2.18 `struct _worldsens_s_sync_reminder _worldsens_pkt::sync_reminder`

Definition at line 245 of file `worldsens_pkt.h`.

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

- [worldsens\\_pkt.h](#)

## 6.38 `_worldsens_s_backtrack` Struct Reference

```
#include <worldsens_pkt.h>
```

### Data Fields

- [ws\\_pkt\\_type](#) type
- [ws\\_id\\_seq](#) seq
- [ws\\_id\\_rp](#) rp\_next
- [ws\\_time](#) rp\_duration

### 6.38.1 Detailed Description

Definition at line 159 of file `worldsens_pkt.h`.

### 6.38.2 Field Documentation

#### 6.38.2.1 ws\_time\_worldsens\_s\_backtrack::rp\_duration

Definition at line 163 of file worldsens\_pkt.h.

#### 6.38.2.2 ws\_id\_rp\_worldsens\_s\_backtrack::rp\_next

Definition at line 162 of file worldsens\_pkt.h.

#### 6.38.2.3 ws\_id\_seq\_worldsens\_s\_backtrack::seq

Definition at line 161 of file worldsens\_pkt.h.

#### 6.38.2.4 ws\_pkt\_type\_worldsens\_s\_backtrack::type

Definition at line 160 of file worldsens\_pkt.h.

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

- [worldsens\\_pkt.h](#)

## 6.39 \_worldsens\_s\_byte\_rx Struct Reference

```
#include <worldsens_pkt.h>
```

### Data Fields

- [ws\\_pkt\\_type](#) type
- [ws\\_id\\_seq](#) seq
- [ws\\_id\\_node](#) node\_id
- [ws\\_id\\_resource](#) antenna\_id
- [ws\\_id\\_resource](#) wsim\_mod\_id
- [ws\\_frequency](#) freq
- [ws\\_power](#) power\_dbm
- [ws\\_sinr](#) sinr
- [ws\\_data](#) data

### 6.39.1 Detailed Description

Definition at line 172 of file worldsens\_pkt.h.

### 6.39.2 Field Documentation

#### 6.39.2.1 `ws_id_resource_worldsens_s_byte_rx::antenna_id`

Definition at line 176 of file `worldsens_pkt.h`.

#### 6.39.2.2 `ws_data_worldsens_s_byte_rx::data`

Definition at line 181 of file `worldsens_pkt.h`.

#### 6.39.2.3 `ws_frequency_worldsens_s_byte_rx::freq`

Definition at line 178 of file `worldsens_pkt.h`.

#### 6.39.2.4 `ws_id_node_worldsens_s_byte_rx::node_id`

Definition at line 175 of file `worldsens_pkt.h`.

#### 6.39.2.5 `ws_power_worldsens_s_byte_rx::power_dbm`

Definition at line 179 of file `worldsens_pkt.h`.

#### 6.39.2.6 `ws_id_seq_worldsens_s_byte_rx::seq`

Definition at line 174 of file `worldsens_pkt.h`.

#### 6.39.2.7 `ws_sinr_worldsens_s_byte_rx::sinr`

Definition at line 180 of file `worldsens_pkt.h`.

#### 6.39.2.8 `ws_pkt_type_worldsens_s_byte_rx::type`

Definition at line 173 of file `worldsens_pkt.h`.

#### 6.39.2.9 `ws_id_resource_worldsens_s_byte_rx::wsim_mod_id`

Definition at line 177 of file `worldsens_pkt.h`.

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

- [worldsens\\_pkt.h](#)

## 6.40 `_worldsens_s_byte_sr_rx` Struct Reference

```
#include <worldsens_pkt.h>
```

### Data Fields

- [ws\\_pkt\\_type](#) type
- [ws\\_id\\_seq](#) seq
- [ws\\_id\\_rp\\_rp\\_next](#) rp\_next
- [ws\\_time\\_rp\\_duration](#) rp\_duration
- [ws\\_id\\_node](#) node\_id
- [ws\\_id\\_resource](#) antenna\_id
- [ws\\_id\\_resource](#) wsim\_mod\_id
- [ws\\_frequency](#) freq
- [ws\\_power](#) power\_dbm
- [ws\\_sinr](#) sinr
- [ws\\_data](#) data

### 6.40.1 Detailed Description

Definition at line 184 of file `worldsens_pkt.h`.

### 6.40.2 Field Documentation

#### 6.40.2.1 `ws_id_resource_worldsens_s_byte_sr_rx::antenna_id`

Definition at line 190 of file `worldsens_pkt.h`.

#### 6.40.2.2 `ws_data_worldsens_s_byte_sr_rx::data`

Definition at line 195 of file `worldsens_pkt.h`.

#### 6.40.2.3 `ws_frequency_worldsens_s_byte_sr_rx::freq`

Definition at line 192 of file `worldsens_pkt.h`.

#### 6.40.2.4 `ws_id_node_worldsens_s_byte_sr_rx::node_id`

Definition at line 189 of file `worldsens_pkt.h`.

#### 6.40.2.5 `ws_power_worldsens_s_byte_sr_rx::power_dbm`

Definition at line 193 of file `worldsens_pkt.h`.

#### 6.40.2.6 `ws_time_worldsens_s_byte_sr_rx::rp_duration`

Definition at line 188 of file `worldsens_pkt.h`.

#### 6.40.2.7 `ws_id_rp_worldsens_s_byte_sr_rx::rp_next`

Definition at line 187 of file `worldsens_pkt.h`.

#### 6.40.2.8 `ws_id_seq_worldsens_s_byte_sr_rx::seq`

Definition at line 186 of file `worldsens_pkt.h`.

#### 6.40.2.9 `ws_sinr_worldsens_s_byte_sr_rx::sinr`

Definition at line 194 of file `worldsens_pkt.h`.

#### 6.40.2.10 `ws_pkt_type_worldsens_s_byte_sr_rx::type`

Definition at line 185 of file `worldsens_pkt.h`.

#### 6.40.2.11 `ws_id_resource_worldsens_s_byte_sr_rx::wsim_mod_id`

Definition at line 191 of file `worldsens_pkt.h`.

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

- [worldsens\\_pkt.h](#)

### 6.41 `_worldsens_s_connect_rsp_nok` Struct Reference

```
#include <worldsens_pkt.h>
```

#### Data Fields

- [ws\\_pkt\\_type](#) type
- [ws\\_id\\_seq](#) seq

#### 6.41.1 Detailed Description

Definition at line 147 of file `worldsens_pkt.h`.

### 6.41.2 Field Documentation

#### 6.41.2.1 `ws_id_seq_worldsens_s_connect_rsp_nok::seq`

Definition at line 149 of file `worldsens_pkt.h`.

#### 6.41.2.2 `ws_pkt_type_worldsens_s_connect_rsp_nok::type`

Definition at line 148 of file `worldsens_pkt.h`.

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

- [worldsens\\_pkt.h](#)

## 6.42 `_worldsens_s_connect_rsp_ok` Struct Reference

```
#include <worldsens_pkt.h>
```

### Data Fields

- [ws\\_pkt\\_type](#) type
- [ws\\_id\\_seq](#) seq
- [ws\\_id\\_rp](#) rp\_next
- [ws\\_time](#) rp\_duration
- [uint8\\_t n\\_antenna\\_id](#)
- [uint8\\_t n\\_modulation\\_id](#)
- [uint8\\_t n\\_measure\\_id](#)
- [char names\\_and\\_ids](#) [WORLDSENS\_MAX\_MODELS\_SIZE]

### 6.42.1 Detailed Description

Definition at line 134 of file `worldsens_pkt.h`.

### 6.42.2 Field Documentation

#### 6.42.2.1 `uint8_t_worldsens_s_connect_rsp_ok::n_antenna_id`

Definition at line 140 of file `worldsens_pkt.h`.

#### 6.42.2.2 `uint8_t_worldsens_s_connect_rsp_ok::n_measure_id`

Definition at line 142 of file `worldsens_pkt.h`.

#### 6.42.2.3 `uint8_t _worldsens_s_connect_rsp_ok::n_modulation_id`

Definition at line 141 of file `worldsens_pkt.h`.

#### 6.42.2.4 `char _worldsens_s_connect_rsp_ok::names_and_ids[WORLDSSENS_MAX_MODELS_SIZE]`

Definition at line 144 of file `worldsens_pkt.h`.

#### 6.42.2.5 `ws_time _worldsens_s_connect_rsp_ok::rp_duration`

Definition at line 138 of file `worldsens_pkt.h`.

#### 6.42.2.6 `ws_id_rp _worldsens_s_connect_rsp_ok::rp_next`

Definition at line 137 of file `worldsens_pkt.h`.

#### 6.42.2.7 `ws_id_seq _worldsens_s_connect_rsp_ok::seq`

Definition at line 136 of file `worldsens_pkt.h`.

#### 6.42.2.8 `ws_pkt_type _worldsens_s_connect_rsp_ok::type`

Definition at line 135 of file `worldsens_pkt.h`.

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

- [worldsens\\_pkt.h](#)

### 6.43 `_worldsens_s_header` Struct Reference

```
#include <worldsens_pkt.h>
```

#### Data Fields

- [ws\\_pkt\\_type](#) type
- [ws\\_id\\_seq](#) seq

#### 6.43.1 Detailed Description

Definition at line 129 of file `worldsens_pkt.h`.



### 6.43.2 Field Documentation

#### 6.43.2.1 `ws_id_seq_worldsens_s_header::seq`

Definition at line 131 of file `worldsens_pkt.h`.

#### 6.43.2.2 `ws_pkt_type_worldsens_s_header::type`

Definition at line 130 of file `worldsens_pkt.h`.

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

- [worldsens\\_pkt.h](#)

## 6.44 `_worldsens_s_kill` Struct Reference

```
#include <worldsens_pkt.h>
```

### Data Fields

- [ws\\_pkt\\_type](#) type
- [ws\\_id\\_seq](#) seq
- [ws\\_id\\_node](#) node\_id

### 6.44.1 Detailed Description

Definition at line 221 of file `worldsens_pkt.h`.

### 6.44.2 Field Documentation

#### 6.44.2.1 `ws_id_node_worldsens_s_kill::node_id`

Definition at line 224 of file `worldsens_pkt.h`.

#### 6.44.2.2 `ws_id_seq_worldsens_s_kill::seq`

Definition at line 223 of file `worldsens_pkt.h`.

#### 6.44.2.3 `ws_pkt_type_worldsens_s_kill::type`

Definition at line 222 of file `worldsens_pkt.h`.

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

- [worldsens\\_pkt.h](#)

## 6.45 `_worldsens_s_killsim` Struct Reference

```
#include <worldsens_pkt.h>
```

### Data Fields

- [ws\\_pkt\\_type](#) type
- [ws\\_id\\_seq](#) seq

### 6.45.1 Detailed Description

Definition at line 216 of file `worldsens_pkt.h`.

### 6.45.2 Field Documentation

#### 6.45.2.1 `ws_id_seq_worldsens_s_killsim::seq`

Definition at line 218 of file `worldsens_pkt.h`.

#### 6.45.2.2 `ws_pkt_type_worldsens_s_killsim::type`

Definition at line 217 of file `worldsens_pkt.h`.

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

- [worldsens\\_pkt.h](#)

## 6.46 `_worldsens_s_measure_rsp` Struct Reference

```
#include <worldsens_pkt.h>
```

### Data Fields

- [ws\\_pkt\\_type](#) type
- [ws\\_id\\_seq](#) seq
- [ws\\_id\\_node](#) node\_id
- [ws\\_id\\_resource](#) measure\_id
- [ws\\_measure](#) measure\_val

### 6.46.1 Detailed Description

Definition at line 198 of file `worldsens_pkt.h`.

### 6.46.2 Field Documentation

#### 6.46.2.1 `ws_id_resource_worldsens_s_measure_rsp::measure_id`

Definition at line 202 of file `worldsens_pkt.h`.

#### 6.46.2.2 `ws_measure_worldsens_s_measure_rsp::measure_val`

Definition at line 203 of file `worldsens_pkt.h`.

#### 6.46.2.3 `ws_id_node_worldsens_s_measure_rsp::node_id`

Definition at line 201 of file `worldsens_pkt.h`.

#### 6.46.2.4 `ws_id_seq_worldsens_s_measure_rsp::seq`

Definition at line 200 of file `worldsens_pkt.h`.

#### 6.46.2.5 `ws_pkt_type_worldsens_s_measure_rsp::type`

Definition at line 199 of file `worldsens_pkt.h`.

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

- [worldsens\\_pkt.h](#)

## 6.47 `_worldsens_s_measure_sr_rsp` Struct Reference

```
#include <worldsens_pkt.h>
```

### Data Fields

- [ws\\_pkt\\_type](#) type
- [ws\\_id\\_seq](#) seq
- [ws\\_id\\_node](#) node\_id
- [ws\\_id\\_resource](#) measure\_id
- [ws\\_measure](#) measure\_val
- [ws\\_id\\_rp](#) rp\_next
- [ws\\_time](#) rp\_duration

### 6.47.1 Detailed Description

Definition at line 206 of file worldsens\_pkt.h.

### 6.47.2 Field Documentation

#### 6.47.2.1 `ws_id_resource_worldsens_s_measure_sr_rsp::measure_id`

Definition at line 210 of file worldsens\_pkt.h.

#### 6.47.2.2 `ws_measure_worldsens_s_measure_sr_rsp::measure_val`

Definition at line 211 of file worldsens\_pkt.h.

#### 6.47.2.3 `ws_id_node_worldsens_s_measure_sr_rsp::node_id`

Definition at line 209 of file worldsens\_pkt.h.

#### 6.47.2.4 `ws_time_worldsens_s_measure_sr_rsp::rp_duration`

Definition at line 213 of file worldsens\_pkt.h.

#### 6.47.2.5 `ws_id_rp_worldsens_s_measure_sr_rsp::rp_next`

Definition at line 212 of file worldsens\_pkt.h.

#### 6.47.2.6 `ws_id_seq_worldsens_s_measure_sr_rsp::seq`

Definition at line 208 of file worldsens\_pkt.h.

#### 6.47.2.7 `ws_pkt_type_worldsens_s_measure_sr_rsp::type`

Definition at line 207 of file worldsens\_pkt.h.

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

- [worldsens\\_pkt.h](#)

## 6.48 `_worldsens_s_sync_release` Struct Reference

```
#include <worldsens_pkt.h>
```

### Data Fields

- [ws\\_pkt\\_type](#) type
- [ws\\_id\\_seq](#) seq
- [ws\\_id\\_rp](#) rp\_next
- [ws\\_time](#) rp\_duration

#### 6.48.1 Detailed Description

Definition at line 152 of file `worldsens_pkt.h`.

#### 6.48.2 Field Documentation

##### 6.48.2.1 `ws_time_worldsens_s_sync_release::rp_duration`

Definition at line 156 of file `worldsens_pkt.h`.

##### 6.48.2.2 `ws_id_rp_worldsens_s_sync_release::rp_next`

Definition at line 155 of file `worldsens_pkt.h`.

##### 6.48.2.3 `ws_id_seq_worldsens_s_sync_release::seq`

Definition at line 154 of file `worldsens_pkt.h`.

##### 6.48.2.4 `ws_pkt_type_worldsens_s_sync_release::type`

Definition at line 153 of file `worldsens_pkt.h`.

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

- [worldsens\\_pkt.h](#)

## 6.49 `_worldsens_s_sync_reminder` Struct Reference

```
#include <worldsens_pkt.h>
```

### Data Fields

- [ws\\_pkt\\_type](#) type
- [ws\\_id\\_seq](#) seq
- [ws\\_id\\_rp](#) rp\_next

### 6.49.1 Detailed Description

Definition at line 166 of file worldsens\_pkt.h.

### 6.49.2 Field Documentation

#### 6.49.2.1 `ws_id_rp_worldsens_s_sync_reminder::rp_next`

Definition at line 169 of file worldsens\_pkt.h.

#### 6.49.2.2 `ws_id_seq_worldsens_s_sync_reminder::seq`

Definition at line 168 of file worldsens\_pkt.h.

#### 6.49.2.3 `ws_pkt_type_worldsens_s_sync_reminder::type`

Definition at line 167 of file worldsens\_pkt.h.

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

- [worldsens\\_pkt.h](#)

## 6.50 `bernoulli_args_s` Struct Reference

```
#include <probabilistic_distribution.h>
```

### Data Fields

- double `p`

### 6.50.1 Detailed Description

Definition at line 150 of file probabilistic\_distribution.h.

### 6.50.2 Field Documentation

#### 6.50.2.1 `double bernoulli_args_s::p`

Definition at line 151 of file probabilistic\_distribution.h.

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

- [probabilistic\\_distribution.h](#)

## 6.51 `beta_args_s` Struct Reference

```
#include <probabilistic_distribution.h>
```

### Data Fields

- double `a`
- double `b`

### 6.51.1 Detailed Description

Definition at line 106 of file `probabilistic_distribution.h`.

### 6.51.2 Field Documentation

#### 6.51.2.1 double `beta_args_s::a`

Definition at line 107 of file `probabilistic_distribution.h`.

#### 6.51.2.2 double `beta_args_s::b`

Definition at line 108 of file `probabilistic_distribution.h`.

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

- [probabilistic\\_distribution.h](#)

## 6.52 `binomial_args_s` Struct Reference

```
#include <probabilistic_distribution.h>
```

### Data Fields

- double `p`
- unsigned int `n`

### 6.52.1 Detailed Description

Definition at line 154 of file `probabilistic_distribution.h`.

### 6.52.2 Field Documentation

#### 6.52.2.1 unsigned int `binomial_args_s::n`

Definition at line 156 of file `probabilistic_distribution.h`.

#### 6.52.2.2 double `binomial_args_s::p`

Definition at line 155 of file `probabilistic_distribution.h`.

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

- [probabilistic\\_distribution.h](#)

## 6.53 `bivariate_gaussian_args_s` Struct Reference

```
#include <probabilistic_distribution.h>
```

### Data Fields

- double `sigma_x`
- double `sigma_y`
- double `rho`
- double \* `return_x`
- double \* `return_y`

#### 6.53.1 Detailed Description

Definition at line 48 of file `probabilistic_distribution.h`.

### 6.53.2 Field Documentation

#### 6.53.2.1 double\* `bivariate_gaussian_args_s::return_x`

Definition at line 52 of file `probabilistic_distribution.h`.

#### 6.53.2.2 double\* `bivariate_gaussian_args_s::return_y`

Definition at line 53 of file `probabilistic_distribution.h`.

#### 6.53.2.3 double `bivariate_gaussian_args_s::rho`

Definition at line 51 of file `probabilistic_distribution.h`.



#### 6.53.2.4 `double bivariate_gaussian_args_s::sigma_x`

Definition at line 49 of file `probabilistic_distribution.h`.

#### 6.53.2.5 `double bivariate_gaussian_args_s::sigma_y`

Definition at line 50 of file `probabilistic_distribution.h`.

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

- [probabilistic\\_distribution.h](#)

## 6.54 `cauchy_args_s` Struct Reference

```
#include <probabilistic_distribution.h>
```

### Data Fields

- `double a`

### 6.54.1 Detailed Description

Definition at line 69 of file `probabilistic_distribution.h`.

### 6.54.2 Field Documentation

#### 6.54.2.1 `double cauchy_args_s::a`

Definition at line 70 of file `probabilistic_distribution.h`.

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

- [probabilistic\\_distribution.h](#)

## 6.55 `chisq_args_s` Struct Reference

```
#include <probabilistic_distribution.h>
```

### Data Fields

- `double nu`

### 6.55.1 Detailed Description

Definition at line 102 of file probabilistic\_distribution.h.

### 6.55.2 Field Documentation

#### 6.55.2.1 double chisq\_args\_s::nu

Definition at line 103 of file probabilistic\_distribution.h.

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

- [probabilistic\\_distribution.h](#)

## 6.56 exponential\_args\_s Struct Reference

```
#include <probabilistic_distribution.h>
```

### Data Fields

- double [mu](#)

### 6.56.1 Detailed Description

Definition at line 56 of file probabilistic\_distribution.h.

### 6.56.2 Field Documentation

#### 6.56.2.1 double exponential\_args\_s::mu

Definition at line 57 of file probabilistic\_distribution.h.

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

- [probabilistic\\_distribution.h](#)

## 6.57 exponential\_s Struct Reference

```
#include <timer.h>
```

## Data Fields

- uint64\_t [initial\\_value](#)
- uint64\_t [ratio](#)
- uint64\_t [offset](#)
- uint64\_t [rank](#)

### 6.57.1 Detailed Description

Definition at line 38 of file timer.h.

### 6.57.2 Field Documentation

#### 6.57.2.1 uint64\_t exponential\_s::initial\_value

Definition at line 39 of file timer.h.

#### 6.57.2.2 uint64\_t exponential\_s::offset

Definition at line 41 of file timer.h.

#### 6.57.2.3 uint64\_t exponential\_s::rank

Definition at line 42 of file timer.h.

#### 6.57.2.4 uint64\_t exponential\_s::ratio

Definition at line 40 of file timer.h.

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

- [timer.h](#)

## 6.58 exppow\_args\_s Struct Reference

```
#include <probabilistic_distribution.h>
```

## Data Fields

- double [a](#)
- double [b](#)

### 6.58.1 Detailed Description

Definition at line 64 of file probabilistic\_distribution.h.

### 6.58.2 Field Documentation

#### 6.58.2.1 double exppow\_args\_s::a

Definition at line 65 of file probabilistic\_distribution.h.

#### 6.58.2.2 double exppow\_args\_s::b

Definition at line 66 of file probabilistic\_distribution.h.

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

- [probabilistic\\_distribution.h](#)

## 6.59 gamma\_args\_s Struct Reference

```
#include <probabilistic_distribution.h>
```

### Data Fields

- double [a](#)
- double [b](#)

### 6.59.1 Detailed Description

Definition at line 87 of file probabilistic\_distribution.h.

### 6.59.2 Field Documentation

#### 6.59.2.1 double gamma\_args\_s::a

Definition at line 88 of file probabilistic\_distribution.h.

#### 6.59.2.2 double gamma\_args\_s::b

Definition at line 89 of file probabilistic\_distribution.h.

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

- [probabilistic\\_distribution.h](#)

## 6.60 gaussian\_args\_s Struct Reference

```
#include <probabilistic_distribution.h>
```

### Data Fields

- double [sigma](#)

#### 6.60.1 Detailed Description

Definition at line 39 of file probabilistic\_distribution.h.

#### 6.60.2 Field Documentation

##### 6.60.2.1 double gaussian\_args\_s::sigma

Definition at line 40 of file probabilistic\_distribution.h.

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

- [probabilistic\\_distribution.h](#)

## 6.61 gaussian\_tail\_args\_s Struct Reference

```
#include <probabilistic_distribution.h>
```

### Data Fields

- double [a](#)
- double [sigma](#)

#### 6.61.1 Detailed Description

Definition at line 43 of file probabilistic\_distribution.h.

#### 6.61.2 Field Documentation

##### 6.61.2.1 double gaussian\_tail\_args\_s::a

Definition at line 44 of file probabilistic\_distribution.h.

### 6.61.2.2 double gaussian\_tail\_args\_s::sigma

Definition at line 45 of file probabilistic\_distribution.h.

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

- [probabilistic\\_distribution.h](#)

## 6.62 geometric\_args\_s Struct Reference

```
#include <probabilistic_distribution.h>
```

### Data Fields

- double [p](#)

### 6.62.1 Detailed Description

Definition at line 159 of file probabilistic\_distribution.h.

### 6.62.2 Field Documentation

#### 6.62.2.1 double geometric\_args\_s::p

Definition at line 160 of file probabilistic\_distribution.h.

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

- [probabilistic\\_distribution.h](#)

## 6.63 gumbel\_t1\_args\_s Struct Reference

```
#include <probabilistic_distribution.h>
```

### Data Fields

- double [a](#)
- double [b](#)

### 6.63.1 Detailed Description

Definition at line 136 of file probabilistic\_distribution.h.

### 6.63.2 Field Documentation

#### 6.63.2.1 double gumbel\_t1\_args\_s::a

Definition at line 137 of file probabilistic\_distribution.h.

#### 6.63.2.2 double gumbel\_t1\_args\_s::b

Definition at line 138 of file probabilistic\_distribution.h.

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

- [probabilistic\\_distribution.h](#)

## 6.64 gumbel\_t2\_args\_s Struct Reference

```
#include <probabilistic_distribution.h>
```

### Data Fields

- double [a](#)
- double [b](#)

### 6.64.1 Detailed Description

Definition at line 141 of file probabilistic\_distribution.h.

### 6.64.2 Field Documentation

#### 6.64.2.1 double gumbel\_t2\_args\_s::a

Definition at line 142 of file probabilistic\_distribution.h.

#### 6.64.2.2 double gumbel\_t2\_args\_s::b

Definition at line 143 of file probabilistic\_distribution.h.

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

- [probabilistic\\_distribution.h](#)

## 6.65 hyper\_geometric\_args\_s Struct Reference

```
#include <probabilistic_distribution.h>
```

## Data Fields

- unsigned int [n1](#)
- unsigned int [n2](#)
- unsigned int [t](#)

### 6.65.1 Detailed Description

Definition at line 163 of file probabilistic\_distribution.h.

### 6.65.2 Field Documentation

#### 6.65.2.1 unsigned int `hyper_geometric_args_s::n1`

Definition at line 164 of file probabilistic\_distribution.h.

#### 6.65.2.2 unsigned int `hyper_geometric_args_s::n2`

Definition at line 165 of file probabilistic\_distribution.h.

#### 6.65.2.3 unsigned int `hyper_geometric_args_s::t`

Definition at line 166 of file probabilistic\_distribution.h.

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

- [probabilistic\\_distribution.h](#)

## 6.66 `laplace_args_s` Struct Reference

```
#include <probabilistic_distribution.h>
```

## Data Fields

- double [a](#)

### 6.66.1 Detailed Description

Definition at line 60 of file probabilistic\_distribution.h.



### 6.66.2 Field Documentation

#### 6.66.2.1 double laplace\_args\_s::a

Definition at line 61 of file probabilistic\_distribution.h.

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

- [probabilistic\\_distribution.h](#)

## 6.67 levy\_alpha\_stable\_s Struct Reference

```
#include <probabilistic_distribution.h>
```

### Data Fields

- double [c](#)
- double [alpha](#)

### 6.67.1 Detailed Description

Definition at line 82 of file probabilistic\_distribution.h.

### 6.67.2 Field Documentation

#### 6.67.2.1 double levy\_alpha\_stable\_s::alpha

Definition at line 84 of file probabilistic\_distribution.h.

#### 6.67.2.2 double levy\_alpha\_stable\_s::c

Definition at line 83 of file probabilistic\_distribution.h.

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

- [probabilistic\\_distribution.h](#)

## 6.68 logarithmic\_args\_s Struct Reference

```
#include <probabilistic_distribution.h>
```

## Data Fields

- double [p](#)

### 6.68.1 Detailed Description

Definition at line 169 of file probabilistic\_distribution.h.

### 6.68.2 Field Documentation

#### 6.68.2.1 double logarithmic\_args\_s::p

Definition at line 170 of file probabilistic\_distribution.h.

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

- [probabilistic\\_distribution.h](#)

## 6.69 logistic\_args\_s Struct Reference

```
#include <probabilistic_distribution.h>
```

## Data Fields

- double [a](#)

### 6.69.1 Detailed Description

Definition at line 111 of file probabilistic\_distribution.h.

### 6.69.2 Field Documentation

#### 6.69.2.1 double logistic\_args\_s::a

Definition at line 112 of file probabilistic\_distribution.h.

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

- [probabilistic\\_distribution.h](#)

## 6.70 lognormal\_args\_s Struct Reference

```
#include <probabilistic_distribution.h>
```

### Data Fields

- double [zeta](#)
- double [sigma](#)

#### 6.70.1 Detailed Description

Definition at line 97 of file `probabilistic_distribution.h`.

#### 6.70.2 Field Documentation

##### 6.70.2.1 double `lognormal_args_s::sigma`

Definition at line 99 of file `probabilistic_distribution.h`.

##### 6.70.2.2 double `lognormal_args_s::zeta`

Definition at line 98 of file `probabilistic_distribution.h`.

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

- [probabilistic\\_distribution.h](#)

### 6.71 `pareto_args_s` Struct Reference

```
#include <probabilistic_distribution.h>
```

### Data Fields

- double [a](#)
- double [b](#)

#### 6.71.1 Detailed Description

Definition at line 115 of file `probabilistic_distribution.h`.

#### 6.71.2 Field Documentation

##### 6.71.2.1 double `pareto_args_s::a`

Definition at line 116 of file `probabilistic_distribution.h`.

### 6.71.2.2 double `pareto_args_s::b`

Definition at line 117 of file `probabilistic_distribution.h`.

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

- [probabilistic\\_distribution.h](#)

## 6.72 `poisson_args_s` Struct Reference

```
#include <probabilistic_distribution.h>
```

### Data Fields

- double [mu](#)

### 6.72.1 Detailed Description

Definition at line 146 of file `probabilistic_distribution.h`.

### 6.72.2 Field Documentation

#### 6.72.2.1 double `poisson_args_s::mu`

Definition at line 147 of file `probabilistic_distribution.h`.

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

- [probabilistic\\_distribution.h](#)

## 6.73 `qtimer_s` Struct Reference

timer structure parameters: parameters for `next_trigger` functions (eg: period for periodic timer) `conditional_end`: pointer to a function that return 1 if the timer must be destroyed and 0 otherwise `callback_function`: function to callback when the timer triggers.

```
#include <timer.h>
```

### Data Fields

- void \* [trigger\\_parameters](#)
- int(\* [conditional\\_end](#))([call\\_t](#) \*c, void \*timer\_id)
- void(\* [callback\\_function](#))([call\\_t](#) \*c, void \*timer\_id)

- `uint64_t(* next_trigger)(call_t *c, void *timer_id)`
- `call_t * c`

### 6.73.1 Detailed Description

timer structure parameters: parameters for next\_trigger functions (eg: period for periodic timer) conditional\_end: pointer to a function that return 1 if the timer must be destroyed and 0 otherwise callback\_function: function to callback when the timer triggers.

Definition at line 18 of file timer.h.

### 6.73.2 Field Documentation

#### 6.73.2.1 `call_t* qtimer_s::c`

Definition at line 23 of file timer.h.

#### 6.73.2.2 `void(* qtimer_s::callback_function)(call_t *c, void *timer_id)`

Definition at line 21 of file timer.h.

#### 6.73.2.3 `int(* qtimer_s::conditional_end)(call_t *c, void *timer_id)`

Definition at line 20 of file timer.h.

#### 6.73.2.4 `uint64_t(* qtimer_s::next_trigger)(call_t *c, void *timer_id)`

Definition at line 22 of file timer.h.

#### 6.73.2.5 `void* qtimer_s::trigger_parameters`

Definition at line 19 of file timer.h.

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

- [timer.h](#)

## 6.74 rayleigh\_s Struct Reference

```
#include <probabilistic_distribution.h>
```

## Data Fields

- double [sigma](#)

### 6.74.1 Detailed Description

Definition at line 73 of file probabilistic\_distribution.h.

### 6.74.2 Field Documentation

#### 6.74.2.1 double rayleigh\_s::sigma

Definition at line 74 of file probabilistic\_distribution.h.

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

- [probabilistic\\_distribution.h](#)

## 6.75 rayleigh\_tail\_s Struct Reference

```
#include <probabilistic_distribution.h>
```

## Data Fields

- double [a](#)
- double [sigma](#)

### 6.75.1 Detailed Description

Definition at line 77 of file probabilistic\_distribution.h.

### 6.75.2 Field Documentation

#### 6.75.2.1 double rayleigh\_tail\_s::a

Definition at line 78 of file probabilistic\_distribution.h.

#### 6.75.2.2 double rayleigh\_tail\_s::sigma

Definition at line 79 of file probabilistic\_distribution.h.

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

- [probabilistic\\_distribution.h](#)

## 6.76 `spherical_vector_2d_args_s` Struct Reference

```
#include <probabilistic_distribution.h>
```

### Data Fields

- double \* [return\\_x](#)
- double \* [return\\_y](#)

### 6.76.1 Detailed Description

Definition at line 120 of file `probabilistic_distribution.h`.

### 6.76.2 Field Documentation

#### 6.76.2.1 `double* spherical_vector_2d_args_s::return_x`

Definition at line 121 of file `probabilistic_distribution.h`.

#### 6.76.2.2 `double* spherical_vector_2d_args_s::return_y`

Definition at line 122 of file `probabilistic_distribution.h`.

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

- [probabilistic\\_distribution.h](#)

## 6.77 `spherical_vector_3d_args_s` Struct Reference

```
#include <probabilistic_distribution.h>
```

### Data Fields

- double \* [return\\_x](#)
- double \* [return\\_y](#)
- double \* [return\\_z](#)

### 6.77.1 Detailed Description

Definition at line 125 of file `probabilistic_distribution.h`.

## 6.77.2 Field Documentation

### 6.77.2.1 `double* spherical_vector_3d_args_s::return_x`

Definition at line 126 of file `probabilistic_distribution.h`.

### 6.77.2.2 `double* spherical_vector_3d_args_s::return_y`

Definition at line 127 of file `probabilistic_distribution.h`.

### 6.77.2.3 `double* spherical_vector_3d_args_s::return_z`

Definition at line 128 of file `probabilistic_distribution.h`.

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

- [probabilistic\\_distribution.h](#)

## 6.78 `uniform_args_s` Struct Reference

```
#include <probabilistic_distribution.h>
```

### Data Fields

- double [a](#)
- double [b](#)

### 6.78.1 Detailed Description

Definition at line 92 of file `probabilistic_distribution.h`.

### 6.78.2 Field Documentation

#### 6.78.2.1 `double uniform_args_s::a`

Definition at line 93 of file `probabilistic_distribution.h`.

#### 6.78.2.2 `double uniform_args_s::b`

Definition at line 94 of file `probabilistic_distribution.h`.

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

- [probabilistic\\_distribution.h](#)



## 6.79 `uniform_random_s` Struct Reference

```
#include <timer.h>
```

### Data Fields

- `uint64_t` [min\\_value](#)
- `uint64_t` [max\\_value](#)

### 6.79.1 Detailed Description

Definition at line 45 of file `timer.h`.

### 6.79.2 Field Documentation

#### 6.79.2.1 `uint64_t uniform_random_s::max_value`

Definition at line 47 of file `timer.h`.

#### 6.79.2.2 `uint64_t uniform_random_s::min_value`

Definition at line 46 of file `timer.h`.

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

- [timer.h](#)

## 6.80 `weibull_args_s` Struct Reference

```
#include <probabilistic_distribution.h>
```

### Data Fields

- `double` [a](#)
- `double` [b](#)

### 6.80.1 Detailed Description

Definition at line 131 of file `probabilistic_distribution.h`.

## 6.80.2 Field Documentation

### 6.80.2.1 `double weibull_args_s::a`

Definition at line 132 of file probabilistic\_distribution.h.

### 6.80.2.2 `double weibull_args_s::b`

Definition at line 133 of file probabilistic\_distribution.h.

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

- [probabilistic\\_distribution.h](#)

## Chapter 7

# File Documentation

### 7.1 antenna.h File Reference

Antenna declarations.

```
#include <include/types.h>
```

#### Functions

- double [antenna\\_get\\_loss](#) ([call\\_t](#) \*c)  
*Return the signal loss induced by an antenna circuit.*
- [angle\\_t](#) \* [antenna\\_get\\_angle](#) ([call\\_t](#) \*c)  
*Return the antenna orientation.*
- void [antenna\\_set\\_angle](#) ([call\\_t](#) \*c, [angle\\_t](#) \*angle)  
*Set the antenna orientation.*
- void [antenna\\_rx](#) ([call\\_t](#) \*c, [packet\\_t](#) \*packet)  
*Forward a received packet to the antenna.*
- double [antenna\\_gain\\_tx](#) ([call\\_t](#) \*c, [position\\_t](#) \*position)  
*Return the tx antenna gain towards the destination direction.*
- double [antenna\\_gain\\_rx](#) ([call\\_t](#) \*c, [position\\_t](#) \*position)  
*Return the rx antenna gain towards the source direction.*

#### 7.1.1 Detailed Description

Antenna declarations.

#### Author

Guillaume Chelius & Elyes Ben Hamida

## Date

2007

Definition in file [antenna.h](#).**7.1.2 Function Documentation****7.1.2.1 double antenna\_gain\_rx ( call\_t \* c, position\_t \* position )**

Return the rx antenna gain towards the source direction.

**Parameters**

<i>c</i>	should be {antenna id, node id, -1}.
<i>position</i>	the source position.

**Returns**

The antenna gain in dB.

**7.1.2.2 double antenna\_gain\_tx ( call\_t \* c, position\_t \* position )**

Return the tx antenna gain towards the destination direction.

**Parameters**

<i>c</i>	should be {antenna id, node id, -1}.
<i>position</i>	the destination position.

**Returns**

The antenna gain in dB.

**7.1.2.3 angle\_t\* antenna\_get\_angle ( call\_t \* c )**

Return the antenna orientation.

**Parameters**

<i>c</i>	should be {antenna id, node id, -1}.
----------	--------------------------------------

**Returns**

Current antenna orientation.

## 7.1.2.4 double antenna\_get\_loss ( call\_t \* c )

Return the signal loss induced by an antenna circuit.

## Parameters

<i>c</i>	should be {antenna id, node id, -1}.
----------	--------------------------------------

## Returns

Signal loss in dB.

## 7.1.2.5 void antenna\_rx ( call\_t \* c, packet\_t \* packet )

Forward a received packet to the antenna.

## Parameters

<i>c</i>	should be {antenna id, node id, -1}.
<i>packet</i>	the received packet.

## 7.1.2.6 void antenna\_set\_angle ( call\_t \* c, angle\_t \* angle )

Set the antenna orientation.

## Parameters

<i>c</i>	should be {antenna id, node id, -1}.
<i>angle</i>	new antenna orientation.

## 7.2 battery.h File Reference

Battery declarations.

```
#include <include/types.h>
```

## Functions

- void [battery\\_consume\\_tx](#) (call\_t \*c, uint64\_t duration, double txdBm)  
*Consume energy associated to a transmission.*
- void [battery\\_consume\\_rx](#) (call\_t \*c, uint64\_t duration)  
*Consume energy associated to a reception.*
- void [battery\\_consume\\_idle](#) (call\_t \*c, uint64\_t duration)  
*Consume energy associated to idle time.*

- void `battery_consume` (`call_t *c`, double *energy*)  
*Consume energy.*
- double `battery_consumed` (`call_t *c`)  
*Return the consumed energy.*
- double `battery_remaining` (`call_t *c`)  
*Return the remaining energy.*
- double `battery_status` (`call_t *c`)  
*Returns the percentage of energy charge.*

### 7.2.1 Detailed Description

Battery declarations.

#### Author

Guillaume Chelius & Elyes Ben Hamida

#### Date

2007

Definition in file `battery.h`.

### 7.2.2 Function Documentation

#### 7.2.2.1 void `battery_consume` ( `call_t * c`, double *energy* )

Consume energy.

##### Parameters

<i>c</i>	should be {-1, node id, -1}.
<i>the</i>	amount of consumed energy.

#### 7.2.2.2 void `battery_consume_idle` ( `call_t * c`, uint64\_t *duration* )

Consume energy associated to idle time.

##### Parameters

<i>c</i>	should be {-1, node id, -1}.
<i>duration</i>	the idle duration.

### 7.2.2.3 void battery\_consume\_rx ( call\_t \* c, uint64\_t duration )

Consume energy associated to a reception.

#### Parameters

<i>c</i>	should be {-1, node id, -1}.
<i>duration</i>	the reception duration.

### 7.2.2.4 void battery\_consume\_tx ( call\_t \* c, uint64\_t duration, double txdBm )

Consume energy associated to a transmission.

#### Parameters

<i>c</i>	should be {-1, node id, -1}.
<i>duration</i>	the transmission duration.
<i>txdBm</i>	the transmission power in dBm.

### 7.2.2.5 double battery\_consumed ( call\_t \* c )

Return the consumed energy.

#### Parameters

<i>c</i>	should be {-1, node id, -1}.
----------	------------------------------

#### Returns

Volume of consumed energy.

### 7.2.2.6 double battery\_remaining ( call\_t \* c )

Return the remaining energy.

#### Parameters

<i>c</i>	should be {-1, node id, -1}.
----------	------------------------------

#### Returns

Volume of remaining energy.

### 7.2.2.7 double battery\_status ( call\_t \* c )

Returns the percentage of energy charge.

#### Parameters

c	should be {-1, node id, -1}.
---	------------------------------

#### Returns

Percentage of energy charge.

## 7.3 das.h File Reference

Data Structure module declarations.

### Typedefs

- typedef int(\* [das\\_delete\\_func\\_t](#))(void \*, void \*)  
*The prototype of a delete function.*

### Functions

- int [das\\_init](#) (void)  
*Initialize the das module.*
- void \* [das\\_create](#) (void)  
*Create an empty data structure.*
- void [das\\_destroy](#) (void \*das)  
*Destroy a data structure.*
- int [das\\_getsize](#) (void \*das)  
*Return the number of elements in the data structure.*
- void [das\\_insert](#) (void \*das, void \*data)  
*Insert an object in the data structure.*
- void \* [das\\_pop](#) (void \*das)  
*Remove an arbitrary object from the data structure.*
- void \* [das\\_pop\\_FIFO](#) (void \*das)  
*Remove the oldest object from the data structure.*
- int [das\\_find](#) (void \*das, void \*data)  
*Tell if an object belong to the data structure.*
- void [das\\_delete](#) (void \*das, void \*data)  
*Remove a particular object from the data structure.*
- void [das\\_selective\\_delete](#) (void \*d, [das\\_delete\\_func\\_t](#) delete, void \*arg)  
*Remove objects selected by a delete function from the data structure.*



- void [das\\_init\\_traverse](#) (void \*das)  
*Initialize a data structure traversal.*
- void \* [das\\_traverse](#) (void \*das)  
*Traverse the data structure.*

### 7.3.1 Detailed Description

DAta Structure module declarations.

#### Author

Guillaume Chelius & Elyes Ben Hamida

#### Date

2007

Definition in file [das.h](#).

### 7.3.2 Typedef Documentation

#### 7.3.2.1 [das\\_delete\\_func\\_t](#)

The prototype of a delete function.

Definition at line 14 of file das.h.

### 7.3.3 Function Documentation

#### 7.3.3.1 [void\\* das\\_create \( void \)](#)

Create an empty data structure.

#### Returns

An opaque pointer to the data structure.

#### 7.3.3.2 [void das\\_delete \( void \\* das, void \\* data \)](#)

Remove a particular object from the data structure.

#### Parameters

<i>das</i>	the opaque pointer to the data structure.
<i>data</i>	the object to remove.

### 7.3.3.3 void **das\_destroy** ( void \* *das* )

Destroy a data structure.

Objects in the structure are not deallocated.

#### Parameters

<i>das</i>	the opaque pointer to the data structure.
------------	---

### 7.3.3.4 int **das\_find** ( void \* *das*, void \* *data* )

Tell if an object belong to the data structure.

#### Parameters

<i>das</i>	the opaque pointer to the data structure.
<i>data</i>	the object to find.

#### Returns

1 if the object is in the data structure, 0 otherwise.

### 7.3.3.5 int **das\_getsize** ( void \* *das* )

Return the number of elements in the data structure.

#### Parameters

<i>das</i>	the opaque pointer to the data structure.
------------	---

#### Returns

The number of elements in the data structure.

### 7.3.3.6 int **das\_init** ( void )

Initialize the das module.

Done by the wsnet core.

#### Returns

0 if success, -1 otherwise.

**7.3.3.7 void das\_init\_traverse ( void \* *das* )**

Initialize a data structure traversal.

**Parameters**

<i>das</i>	the opaque pointer to the data structure.
------------	---

**7.3.3.8 void das\_insert ( void \* *das*, void \* *data* )**

Insert an object in the data structure.

**Parameters**

<i>das</i>	the opaque pointer to the data structure.
<i>data</i>	the object to insert.

**7.3.3.9 void\* das\_pop ( void \* *das* )**

Remove an arbitrary object from the data structure.

**Parameters**

<i>das</i>	the opaque pointer to the data structure.
------------	---

**Returns**

An arbitrary object.

**7.3.3.10 void\* das\_pop\_FIFO ( void \* *das* )**

Remove the oldest object from the data structure.

**Parameters**

<i>das</i>	the opaque pointer to the data structure.
------------	---

**Returns**

The oldest object.

**7.3.3.11 void das\_selective\_delete ( void \* *d*, das\_delete\_func\_t *delete*, void \* *arg* )**

Remove objects selected by a delete function from the data structure.

## Parameters

<i>das</i>	the opaque pointer to the data structure.
<i>delete</i>	the function that selects objects to be removed.
<i>arg</i>	an argument passed to the delete function.

## 7.3.3.12 void\* das\_traverse ( void \* das )

Traverse the data structure.

## Parameters

<i>das</i>	the opaque pointer to the data structure.
------------	---

## Returns

The next object in the traversal.

## 7.4 dbg.h File Reference

dbg declarations

## Defines

- #define [DEBUG\\_MAX](#) 9
- #define [DBG](#)(lvl, msg,...)
- #define [NDBG](#)(lvl, msg, c,...)
- #define [DBG\\_CRIT](#)(msg,...) [DBG](#)(0, msg, ## \_\_VA\_ARGS\_\_)
- #define [NDBG\\_CRIT](#)(msg, c,...) [NDBG](#)(0, msg, c, ## \_\_VA\_ARGS\_\_)
- #define [DBG\\_WARN](#)(msg,...) [DBG](#)(1, msg, ## \_\_VA\_ARGS\_\_)
- #define [NDBG\\_WARN](#)(msg,...) [NDBG](#)(1, msg, c, ## \_\_VA\_ARGS\_\_)
- #define [DBG\\_INFO](#)(msg,...) [DBG](#)(2, msg, ## \_\_VA\_ARGS\_\_)
- #define [NDBG\\_INFO](#)(msg, c,...) [NDBG](#)(2, msg, c, ## \_\_VA\_ARGS\_\_)
- #define [DBG\\_VERB](#)(msg,...) [DBG](#)(3, msg, ## \_\_VA\_ARGS\_\_)
- #define [NDBG\\_VERB](#)(msg, c,...) [NDBG](#)(3, msg, c, ## \_\_VA\_ARGS\_\_)
- #define [DBG\\_NOISE](#)(msg,...) [DBG](#)(4, msg, ## \_\_VA\_ARGS\_\_)
- #define [NDBG\\_NOISE](#)(msg, c,...) [NDBG](#)(4, msg, c, ## \_\_VA\_ARGS\_\_)
- #define [DBG\\_XTRM](#)(msg,...) [DBG](#)(5, msg, ## \_\_VA\_ARGS\_\_)
- #define [NDBG\\_XTRM](#)(msg, c,...) [NDBG](#)(5, msg, c, ## \_\_VA\_ARGS\_\_)

## Functions

- static void [get\\_debug](#) ([call\\_t](#) \*c, char \*\*entity\_name, int \*debug, int \*log)

### 7.4.1 Detailed Description

dbg declarations

**Author**

Stephane D'Alu

**Date**

2008

Definition in file [dbg.h](#).

### 7.4.2 Define Documentation

#### 7.4.2.1 #define DBG( *lvl*, *msg*, ... )

**Value:**

```
if (lvl <= DBG_VAR) {
    fprintf(stderr, "%s: ", DBG_NAME);
    fprintf(stderr, msg "\n", ## __VA_ARGS__);
}
```

Definition at line 16 of file dbg.h.

#### 7.4.2.2 #define DBG\_CRIT( *msg*, ... ) DBG(0, msg, ## \_\_VA\_ARGS\_\_)

Definition at line 32 of file dbg.h.

#### 7.4.2.3 #define DBG\_INFO( *msg*, ... ) DBG(2, msg, ## \_\_VA\_ARGS\_\_)

Definition at line 48 of file dbg.h.

#### 7.4.2.4 #define DBG\_NOISE( *msg*, ... ) DBG(4, msg, ## \_\_VA\_ARGS\_\_)

Definition at line 64 of file dbg.h.

#### 7.4.2.5 #define DBG\_VERB( *msg*, ... ) DBG(3, msg, ## \_\_VA\_ARGS\_\_)

Definition at line 56 of file dbg.h.

#### 7.4.2.6 #define DBG\_WARN( *msg*, ... ) DBG(1, msg, ## \_\_VA\_ARGS\_\_)

Definition at line 40 of file dbg.h.

7.4.2.7 **#define DBG\_XTRM( *msg*, ... )** DBG(5, *msg*, ## \_\_VA\_ARGS\_\_)

Definition at line 72 of file dbg.h.

7.4.2.8 **#define DEBUG\_MAX** 9

Definition at line 12 of file dbg.h.

7.4.2.9 **#define NDBG( *lvl*, *msg*, *c*, ... )**

**Value:**

```
if (lvl <= DBG_VAR) {
    fprintf(stderr, "%s[%3d]: ", DBG_NAME, (c)->node);
    fprintf(stderr, msg "\n", ## __VA_ARGS__);
}
```

Definition at line 21 of file dbg.h.

7.4.2.10 **#define NDBG\_CRIT( *msg*, *c*, ... )** NDBG(0, *msg*, *c*, ## \_\_VA\_ARGS\_\_)

Definition at line 33 of file dbg.h.

7.4.2.11 **#define NDBG\_INFO( *msg*, *c*, ... )** NDBG(2, *msg*, *c*, ## \_\_VA\_ARGS\_\_)

Definition at line 49 of file dbg.h.

7.4.2.12 **#define NDBG\_NOISE( *msg*, *c*, ... )** NDBG(4, *msg*, *c*, ## \_\_VA\_ARGS\_\_)

Definition at line 65 of file dbg.h.

7.4.2.13 **#define NDBG\_VERB( *msg*, *c*, ... )** NDBG(3, *msg*, *c*, ## \_\_VA\_ARGS\_\_)

Definition at line 57 of file dbg.h.

7.4.2.14 **#define NDBG\_WARN( *msg*, ... )** NDBG(1, *msg*, *c*, ## \_\_VA\_ARGS\_\_)

Definition at line 41 of file dbg.h.

7.4.2.15 **#define NDBG\_XTRM( *msg*, *c*, ... )** NDBG(5, *msg*, *c*, ## \_\_VA\_ARGS\_\_)

Definition at line 73 of file dbg.h.

### 7.4.3 Function Documentation

7.4.3.1 static void `get_debug` ( `call_t * c`, `char ** entity_name`, `int * debug`, `int * log` )  
`[inline, static]`

Definition at line 81 of file `dbg.h`.

References `get_entity_name()`.

## 7.5 entity.h File Reference

Entity declarations.

```
#include <include/types.h>
```

### Functions

- `char * get_entity_name (call_t *c)`  
*Return the name of an entity.*
- `void * get_entity_private_data (call_t *c)`  
*Return the private data associated to an entity.*
- `void set_entity_private_data (call_t *c, void *data)`  
*Associate a private data memory space with an entity.*
- `void * get_node_private_data (call_t *c)`  
*Return the private data associated to an (entity, node).*
- `void set_node_private_data (call_t *c, void *data)`  
*Associate a private data memory space with an (entity, node).*
- `int get_entity_type (call_t *c)`  
*Return the type of an entity.*
- `array_t * get_entity_bindings_up (call_t *c)`  
*Return an array containing the entities that are up "c->entity" in "c->node".*
- `array_t * get_entity_bindings_down (call_t *c)`  
*Return an array containing the entities that are down "c->entity" in "c->node".*
- `array_t * get_application_entities (call_t *c)`  
*Return an array containing the application entities in "c->node".*
- `array_t * get_routing_entities (call_t *c)`  
*Return an array containing the routing entities in "c->node".*
- `array_t * get_mac_entities (call_t *c)`  
*Return an array containing the mac entities in "c->node".*
- `array_t * get_radio_entities (call_t *c)`  
*Return an array containing the radio entities in "c->node".*
- `array_t * get_antenna_entities (call_t *c)`  
*Return an array containing the antenna entities in "c->node".*
- `entityid_t get_energy_entity (call_t *c)`

*Return an array containing the energy entity in "c->node".*

- [entityid\\_t get\\_mobility\\_entity \(call\\_t \\*c\)](#)

*Return an array containing the mobility entity in "c->node".*

- [int get\\_entity\\_links\\_up\\_nbr \(call\\_t \\*c\)](#)

*Return the number of entities that are linked up with "c->entity" in "c->node".*

- [entityid\\_t \\* get\\_entity\\_links\\_up \(call\\_t \\*c\)](#)

*Return an array containing the entity ids of the entities linked up with "c->entity" in "c->node".*

- [int get\\_entity\\_links\\_down\\_nbr \(call\\_t \\*c\)](#)

*Return the number of entities that are linked down with "c->entity" in "c->node".*

- [entityid\\_t \\* get\\_entity\\_links\\_down \(call\\_t \\*c\)](#)

*Return an array containing the entity ids of the entities linked down with "c->entity" in "c->node".*

### 7.5.1 Detailed Description

Entity declarations. Parameter parsing declarations.

#### Author

Guillaume Chelius & Elyes Ben Hamida

#### Date

2007

Definition in file [entity.h](#).

### 7.5.2 Function Documentation

#### 7.5.2.1 [array\\_t\\* get\\_antenna\\_entities \( call\\_t \\* c \)](#)

Return an array containing the antenna entities in "c->node".

#### Parameters

<a href="#">c</a>	could be {-1, node id, -1}.
-------------------	-----------------------------

#### Returns

An array of entity ids.

#### 7.5.2.2 [array\\_t\\* get\\_application\\_entities \( call\\_t \\* c \)](#)

Return an array containing the application entities in "c->node".



## Parameters

<i>c</i>	could be {-1, node id, -1}.
----------	-----------------------------

## Returns

An array of entity ids.

**7.5.2.3 entityid\_t get\_energy\_entity ( call\_t \* *c* )**

Return an array containing the energy entity in "c->node".

## Parameters

<i>c</i>	could be {-1, node id, -1}.
----------	-----------------------------

## Returns

An entity id.

**7.5.2.4 array\_t\* get\_entity\_bindings\_down ( call\_t \* *c* )**

Return an array containing the entities that are down "c->entity" in "c->node".

## Parameters

<i>c</i>	could be {entity id, node id, -1}.
----------	------------------------------------

## Returns

An array of entity ids.

**7.5.2.5 array\_t\* get\_entity\_bindings\_up ( call\_t \* *c* )**

Return an array containing the entities that are up "c->entity" in "c->node".

## Parameters

<i>c</i>	could be {entity id, node id, -1}.
----------	------------------------------------

## Returns

An array of entity ids.

### 7.5.2.6 `entityid_t* get_entity_links_down ( call_t * c )`

Return an array containing the entity ids of the entities linked down with "c->entity" in "c->node".

#### Parameters

<code>c</code>	could be {entity id, node id, -1}.
----------------	------------------------------------

#### Returns

Array of entity ids.

**Deprecated** Should use [get\\_entity\\_bindings\\_down\(\)](#) instead.

### 7.5.2.7 `int get_entity_links_down_nbr ( call_t * c )`

Return the number of entities that are linked down with "c->entity" in "c->node".

#### Parameters

<code>c</code>	could be {entity id, node id, -1}.
----------------	------------------------------------

#### Returns

The number of linked down entities.

**Deprecated** Should use [get\\_entity\\_bindings\\_down\(\)](#) instead.

### 7.5.2.8 `entityid_t* get_entity_links_up ( call_t * c )`

Return an array containing the entity ids of the entities linked up with "c->entity" in "c->node".

#### Parameters

<code>c</code>	could be {entity id, node id, -1}.
----------------	------------------------------------

#### Returns

Array of entity ids.

**Deprecated** Should use [get\\_entity\\_bindings\\_up\(\)](#) instead.

### 7.5.2.9 int get\_entity\_links\_up\_nbr ( call\_t \* c )

Return the number of entities that are linked up with "c->entity" in "c->node".

#### Parameters

<i>c</i>	could be {entity id, node id, -1}.
----------	------------------------------------

#### Returns

The number of linked up entities.

**Deprecated** Should use [get\\_entity\\_bindings\\_up\(\)](#) instead.

### 7.5.2.10 char\* get\_entity\_name ( call\_t \* c )

Return the name of an entity.

#### Parameters

<i>c</i>	could be {entity id, -1, -1}.
----------	-------------------------------

#### Returns

The name of the entity.

Referenced by [get\\_debug\(\)](#).

### 7.5.2.11 void\* get\_entity\_private\_data ( call\_t \* c )

Return the private data associated to an entity.

#### Parameters

<i>c</i>	could be {entity id, -1, -1}.
----------	-------------------------------

#### Returns

A (void \*) pointer to the entity private data.

### 7.5.2.12 int get\_entity\_type ( call\_t \* c )

Return the type of an entity.

**Parameters**

<b>c</b>	could be {entity id, -1, -1}.
----------	-------------------------------

**Returns**

The entity type.

**7.5.2.13 array\_t\* get\_mac\_entities ( call\_t \* c )**

Return an array containing the mac entities in "c->node".

**Parameters**

<b>c</b>	could be {-1, node id, -1}.
----------	-----------------------------

**Returns**

An array of entity ids.

**7.5.2.14 entityid\_t get\_mobility\_entity ( call\_t \* c )**

Return an array containing the mobility entity in "c->node".

**Parameters**

<b>c</b>	could be {-1, node id, -1}.
----------	-----------------------------

**Returns**

An entity id.

**7.5.2.15 void\* get\_node\_private\_data ( call\_t \* c )**

Return the private data associated to an (entity, node).

**Parameters**

<b>c</b>	could be {entity id, node id, -1}.
----------	------------------------------------

**Returns**

A (void \*) pointer to the (entity, node) private data.

**7.5.2.16 array\_t\* get\_radio\_entities ( call\_t \* c )**

Return an array containing the radio entities in "c->node".

**Parameters**

<i>c</i>	could be {-1, node id, -1}.
----------	-----------------------------

**Returns**

An array of entity ids.

**7.5.2.17 array\_t\* get\_routing\_entities ( call\_t \* c )**

Return an array containing the routing entities in "c->node".

**Parameters**

<i>c</i>	could be {-1, node id, -1}.
----------	-----------------------------

**Returns**

An array of entity ids.

**7.5.2.18 void set\_entity\_private\_data ( call\_t \* c, void \* data )**

Associate a private data memory space with an entity.

**Parameters**

<i>c</i>	could be {entity id, -1, -1}.
<i>data</i>	a (void *) pointer to the private memory space.

**7.5.2.19 void set\_node\_private\_data ( call\_t \* c, void \* data )**

Associate a private data memory space with an (entity, node).

**Parameters**

<i>c</i>	could be {entity id, node id, -1}.
<i>data</i>	a (void *) pointer to the private memory space.

## 7.6 hadas.h File Reference

Hashed DAta Structure module declarations.

### Typedefs

- typedef unsigned long(\* [hash\\_hash\\_t](#))(void \*key)  
*The prototype of a hash function.*
- typedef int(\* [hash\\_equal\\_t](#))(void \*key0, void \*key1)  
*The prototype of an equal function.*

### Functions

- int [hadas\\_init](#) (void)  
*Initialize the hadas module.*
- void \* [hadas\\_create](#) ([hash\\_hash\\_t](#) hash, [hash\\_equal\\_t](#) equal)  
*Create an empty hadas structure.*
- void [hadas\\_destroy](#) (void \*hadas)  
*Destroy a hadas structure.*
- void [hadas\\_insert](#) (void \*hadas, void \*key, void \*data)  
*Insert a (key,object) in the hadas structure.*
- void \* [hadas\\_get](#) (void \*hadas, void \*key)  
*Return the object associated to a key in the hadas structure.*
- void [hadas\\_delete](#) (void \*hadas, void \*key)  
*Remove the object associated to a key in the hadas structure.*

### 7.6.1 Detailed Description

Hashed DAta Structure module declarations.

#### Author

Guillaume Chelius & Elyes Ben Hamida

#### Date

2007

Definition in file [hadas.h](#).

### 7.6.2 Typedef Documentation

#### 7.6.2.1 hash\_equal\_t

The prototype of an equal function.

Definition at line 20 of file [hadas.h](#).

### 7.6.2.2 hash\_hash\_t

The prototype of a hash function.

Definition at line 14 of file hadas.h.

## 7.6.3 Function Documentation

### 7.6.3.1 void\* hadas\_create ( hash\_hash\_t hash, hash\_equal\_t equal )

Create an empty hadas structure.

#### Parameters

<i>hash</i>	the function used to hash object keys.
<i>equal</i>	the function used to decide equality between keys.

#### Returns

An opaque pointer to the hadas structure.

### 7.6.3.2 void hadas\_delete ( void \* hadas, void \* key )

Remove the object associated to a key in the hadas structure.

#### Parameters

<i>hadas</i>	the opaque pointer to the data structure.
<i>key</i>	the key of the object to be removed.

### 7.6.3.3 void hadas\_destroy ( void \* hadas )

Destroy a hadas structure.

Objects and keys in the structure are not deallocated.

#### Parameters

<i>hadas</i>	the opaque pointer to the hadas structure.
--------------	--

### 7.6.3.4 void\* hadas\_get ( void \* hadas, void \* key )

Return the object associated to a key in the hadas structure.

## Parameters

<i>hadas</i>	the opaque pointer to the data structure.
<i>key</i>	the key of the object to get.

## Returns

The object.

7.6.3.5 `int hadas_init ( void )`

Initialize the hadas module.

Done by the wsnet core.

## Returns

0 if success, -1 otherwise.

7.6.3.6 `void hadas_insert ( void * hadas, void * key, void * data )`

Insert a (key,object) in the hadas structure.

## Parameters

<i>hadas</i>	the opaque pointer to the data structure.
<i>key</i>	the key of the object to insert.
<i>data</i>	the object to insert.

7.7 `ioctl_message.h` File Reference

## Functions

- `ioctl_message_t * ioctl_message_create` (int type, int size, int real\_size, void \*body)  
*Create an ioctl message.*
- `void ioctl_message_dealloc` (void \*message)  
*Deallocate an ioctl message.*
- `void ioctl_message_body_duplicate` (void \*message, void \*pointer)  
*Duplicate an ioctl message.*
- `void * get_ioctl_message_body` (void \*message)  
*Return the message body.*
- `int get_ioctl_message_type` (void \*message)  
*Return the message type.*
- `int get_ioctl_message_size` (void \*message)



*Return the size of the message body.*

- int [get\\_ioctl\\_message\\_real\\_size](#) (void \*message)

*Return the real size of the message body.*

### 7.7.1 Function Documentation

#### 7.7.1.1 void\* get\_ioctl\_message\_body ( void \* message )

Return the message body.

##### Parameters

<i>message</i>	the message.
----------------	--------------

#### 7.7.1.2 int get\_ioctl\_message\_real\_size ( void \* message )

Return the real size of the message body.

##### Parameters

<i>message</i>	the message.
----------------	--------------

#### 7.7.1.3 int get\_ioctl\_message\_size ( void \* message )

Return the size of the message body.

##### Parameters

<i>message</i>	the message.
----------------	--------------

#### 7.7.1.4 int get\_ioctl\_message\_type ( void \* message )

Return the message type.

##### Parameters

<i>message</i>	the message.
----------------	--------------

#### 7.7.1.5 void ioctl\_message\_body\_duplicate ( void \* message, void \* pointer )

Duplicate an ioctl message.

## Parameters

<i>message</i>	the message to duplicate.
----------------	---------------------------

### 7.7.1.6 `ioctl_message_t* ioctl_message_create ( int type, int size, int real_size, void * body )`

Create an ioctl message.

## Parameters

<i>type</i>	type of the message.
<i>size</i>	size of the message.
<i>real_size</i>	real size of the message.
<i>body</i>	body of the message.

### 7.7.1.7 `void ioctl_message_dealloc ( void * message )`

Deallocate an ioctl message.

## Parameters

<i>message</i>	the message to deallocate.
----------------	----------------------------

## 7.8 log.h File Reference

log declarations

```
#include <include/options.h>
```

## Defines

- `#define PRINT_REPLAY(x...) do { } while (0)`
- `#define PRINT_APPLICATION(x...) do { } while (0)`
- `#define PRINT_ROUTING(x...) do { } while (0)`
- `#define PRINT_MAC(x...) do { } while (0)`
- `#define PRINT_RADIO(x...) do { } while (0)`
- `#define PRINT_ANTENNA(x...) do { } while (0)`
- `#define PRINT_MOBILITY(x...) do { } while (0)`
- `#define PRINT_ENERGY(x...) do { } while (0)`
- `#define PRINT_ENVIRONMENT(x...) do { } while (0)`
- `#define PRINT_MONITOR(x...) do { } while (0)`
- `#define PRINT_MODULATION(x...) do { } while (0)`
- `#define PRINT_INTERFERENCES(x...) do { } while (0)`
- `#define PRINT_PROPAGATION(x...) do { } while (0)`
- `#define PRINT_WORLDSSENS(x...) do { } while (0)`

### 7.8.1 Detailed Description

log declarations

**Author**

Guillaume Chelius & Elyes Ben Hamida

**Date**

2007

Definition in file [log.h](#).

### 7.8.2 Define Documentation

**7.8.2.1** `#define PRINT_ANTENNA( x... ) do { } while (0)`

Definition at line 45 of file log.h.

**7.8.2.2** `#define PRINT_APPLICATION( x... ) do { } while (0)`

Definition at line 21 of file log.h.

**7.8.2.3** `#define PRINT_ENERGY( x... ) do { } while (0)`

Definition at line 57 of file log.h.

**7.8.2.4** `#define PRINT_ENVIRONMENT( x... ) do { } while (0)`

Definition at line 63 of file log.h.

**7.8.2.5** `#define PRINT_INTERFERENCES( x... ) do { } while (0)`

Definition at line 81 of file log.h.

**7.8.2.6** `#define PRINT_MAC( x... ) do { } while (0)`

Definition at line 33 of file log.h.

**7.8.2.7** `#define PRINT_MOBILITY( x... ) do { } while (0)`

Definition at line 51 of file log.h.

7.8.2.8 **#define PRINT\_MODULATION( x... ) do { } while (0)**

Definition at line 75 of file log.h.

7.8.2.9 **#define PRINT\_MONITOR( x... ) do { } while (0)**

Definition at line 69 of file log.h.

7.8.2.10 **#define PRINT\_PROPAGATION( x... ) do { } while (0)**

Definition at line 87 of file log.h.

7.8.2.11 **#define PRINT\_RADIO( x... ) do { } while (0)**

Definition at line 39 of file log.h.

7.8.2.12 **#define PRINT\_REPLAY( x... ) do { } while (0)**

Definition at line 15 of file log.h.

7.8.2.13 **#define PRINT\_ROUTING( x... ) do { } while (0)**

Definition at line 27 of file log.h.

7.8.2.14 **#define PRINT\_WORLDSENS( x... ) do { } while (0)**

Definition at line 94 of file log.h.

## 7.9 measure.h File Reference

Measure declarations.

```
#include <include/types.h>
```

### Functions

- [measureid\\_t get\\_measureid\\_by\\_name](#) (char \*name)  
*Return the measure id associated to a measure name.*
- void [READ\\_MEASURE](#) ([call\\_t](#) \*c, [measureid\\_t](#) measure, double \*value)  
*Return the measure id associated to a measure name.*

### 7.9.1 Detailed Description

Measure declarations.

**Author**

Guillaume Chelius & Elyes Ben Hamida

**Date**

2007

Definition in file [measure.h](#).

### 7.9.2 Function Documentation

#### 7.9.2.1 `measureid_t get_measureid_by_name ( char * name )`

Return the measure id associated to a measure name.

**Parameters**

<i>name</i>	the measure name.
-------------	-------------------

**Returns**

The measure id.

#### 7.9.2.2 `void READ_MEASURE ( call_t * c, measureid_t measure, double * value )`

Return the measure id associated to a measure name.

**Parameters**

<i>c</i>	should be {caller entity id, caller node id, -1}.
<i>measure</i>	the measure id.
<i>value</i>	a pointer where to put the read value.

## 7.10 medium.h File Reference

Medium declarations.

```
#include <include/types.h> #include <math.h>
```

## Defines

- `#define MIN_DBM -DBL_MAX`  
*Signal strength of a null signal in dBm.*
- `#define MAX_SNR DBL_MAX`  
*Maximum SNR value, when there is no interference nor noise.*

## Functions

- static double `dBm2mW` (double dBm)  
*Convert a dBm value into a mW value.*
- static double `mW2dBm` (double mW)  
*Convert a mW value into a dBm value.*
- void `MEDIA_TX` (`call_t` \*c, `packet_t` \*packet)  
*Transmit a packet in the radio medium.*
- double `MEDIA_GET_NOISE` (`call_t` \*c, int channel)  
*Return the radio medium noise on a given channel.*

### 7.10.1 Detailed Description

Medium declarations.

#### Author

Guillaume Chelius & Elyes Ben Hamida

#### Date

2007

Definition in file [medium.h](#).

### 7.10.2 Define Documentation

#### 7.10.2.1 `#define MAX_SNR DBL_MAX`

Maximum SNR value, when there is no interference nor noise.

Definition at line 26 of file `medium.h`.

#### 7.10.2.2 `#define MIN_DBM -DBL_MAX`

Signal strength of a null signal in dBm.

Definition at line 19 of file `medium.h`.

Referenced by `dBm2mW()`, and `mW2dBm()`.

### 7.10.3 Function Documentation

7.10.3.1 `static double dBm2mW ( double dBm ) [inline, static]`

Convert a dBm value into a mW value.

#### Parameters

<i>dBm</i>	the dBm value to convert.
------------	---------------------------

#### Returns

The converted mW value.

Definition at line 34 of file medium.h.

References MIN\_DBM.

7.10.3.2 `double MEDIA_GET_NOISE ( call_t * c, int channel )`

Return the radio medium noise on a given channel.

#### Parameters

<i>c</i>	should be {radio id, node id, antenna id}.
<i>channel</i>	the radio channel we are listening to.

#### Returns

the noise strength in dBm.

7.10.3.3 `void MEDIA_TX ( call_t * c, packet_t * packet )`

Transmit a packet in the radio medium.

#### Parameters

<i>c</i>	should be {radio id, node id, antenna id}.
<i>packet</i>	the transmitted packet.

7.10.3.4 `static double mW2dBm ( double mW ) [inline, static]`

Convert a mW value into a dBm value.

#### Parameters

<i>mW</i>	the mW value to convert.
-----------	--------------------------

**Returns**

The converted dBm value.

Definition at line 44 of file medium.h.

References MIN\_DBM.

## 7.11 mem\_fs.h File Reference

Fixed size memory management module declarations.

**Functions**

- void [mem\\_fs\\_clean](#) (void)  
*Clean the mem\_fs module.*
- void \* [mem\\_fs\\_slice\\_declare](#) (int size)  
*Declare a memory allocation slice.*
- void \* [mem\\_fs\\_alloc](#) (void \*slice)  
*Allocate a memory block from a slice.*
- void [mem\\_fs\\_dealloc](#) (void \*slice, void \*pointer)  
*Deallocate a memory block.*

### 7.11.1 Detailed Description

Fixed size memory management module declarations.

**Author**

Guillaume Chelius & Elyes Ben Hamida

**Date**

2007

Definition in file [mem\\_fs.h](#).

### 7.11.2 Function Documentation

#### 7.11.2.1 void\* mem\_fs\_alloc ( void \* slice )

Allocate a memory block from a slice.

**Parameters**

<i>slice</i>	the opaque pointer to the memory slice.
--------------	---



**Returns**

The allocated memory block.

**7.11.2.2 void mem\_fs\_clean ( void )**

Clean the mem\_fs module.

Done by the wsnet core.

**7.11.2.3 void mem\_fs\_dealloc ( void \* slice, void \* pointer )**

Deallocate a memory block.

**Parameters**

<i>slice</i>	the opaque pointer to the memory slice.
--------------	---

**Returns**

pointer the memory block to deallocate.

**7.11.2.4 void\* mem\_fs\_slice\_declare ( int size )**

Declare a memory allocation slice.

**Parameters**

<i>size</i>	the size of the memory blocs that will be allocated using this slice.
-------------	---

**Returns**

An opaque pointer to the memory slice.

**7.12 models.h File Reference**

Models declarations.

```
#include <include/types.h>
```

**Data Structures**

- struct [\\_model](#)  
*Information about a model.*
- struct [\\_propagation\\_methods](#)

*Methods that should be implemented by a propagation model.*

- struct [\\_fading\\_methods](#)

*Methods that should be implemented by a fading model.*

- struct [\\_shadowing\\_methods](#)

*Methods that should be implemented by a shadowing model.*

- struct [\\_interferences\\_methods](#)

*Methods that should be implemented by a interferences model.*

- struct [\\_noise\\_methods](#)

*Methods that should be implemented by a noise model.*

- struct [\\_modulation\\_methods](#)

*Methods that should be implemented by a modulation model.*

- struct [\\_environment\\_methods](#)

*Methods that should be implemented by an environment model.*

- struct [\\_monitor\\_methods](#)

*Methods that should be implemented by a monitor model.*

- struct [\\_mobility\\_methods](#)

*Methods that should be implemented by a mobility model.*

- struct [\\_energy\\_methods](#)

*Methods that should be implemented by an energy model.*

- struct [\\_antenna\\_methods](#)

*Methods that should be implemented by an antenna model.*

- struct [\\_radio\\_methods](#)

*Methods that should be implemented by a radio model.*

- struct [\\_mac\\_methods](#)

*Methods that should be implemented by a mac model.*

- struct [\\_routing\\_methods](#)

*Methods that should be implemented by a routing model.*

- struct [\\_application\\_methods](#)

*Methods that should be implemented by an application model.*

## Defines

- #define [MODELTYPE\\_PROPAGATION](#) 0
- #define [MODELTYPE\\_SHADOWING](#) 1
- #define [MODELTYPE\\_FADING](#) 2
- #define [MODELTYPE\\_INTERFERENCES](#) 3
- #define [MODELTYPE\\_NOISE](#) 4
- #define [MODELTYPE\\_MODULATION](#) 5
- #define [MODELTYPE\\_ENVIRONMENT](#) 6
- #define [MODELTYPE\\_MOBILITY](#) 7
- #define [MODELTYPE\\_ANTENNA](#) 8
- #define [MODELTYPE\\_RADIO](#) 9
- #define [MODELTYPE\\_MAC](#) 10
- #define [MODELTYPE\\_ROUTING](#) 11

- #define [MODELTYPE\\_APPLICATION](#) 12
- #define [MODELTYPE\\_ENERGY](#) 13
- #define [MODELTYPE\\_MONITOR](#) 14

## Typedefs

- typedef struct [\\_model](#) [model\\_t](#)  
*Information about a model.*
- typedef struct [\\_propagation\\_methods](#) [propagation\\_methods\\_t](#)  
*Methods that should be implemented by a propagation model.*
- typedef struct [\\_fading\\_methods](#) [fading\\_methods\\_t](#)  
*Methods that should be implemented by a fading model.*
- typedef struct [\\_shadowing\\_methods](#) [shadowing\\_methods\\_t](#)  
*Methods that should be implemented by a shadowing model.*
- typedef struct [\\_interferences\\_methods](#) [interferences\\_methods\\_t](#)  
*Methods that should be implemented by a interferences model.*
- typedef struct [\\_noise\\_methods](#) [noise\\_methods\\_t](#)  
*Methods that should be implemented by a noise model.*
- typedef struct [\\_modulation\\_methods](#) [modulation\\_methods\\_t](#)  
*Methods that should be implemented by a modulation model.*
- typedef struct [\\_environment\\_methods](#) [environment\\_methods\\_t](#)  
*Methods that should be implemented by an environment model.*
- typedef struct [\\_monitor\\_methods](#) [monitor\\_methods\\_t](#)  
*Methods that should be implemented by a monitor model.*
- typedef struct [\\_mobility\\_methods](#) [mobility\\_methods\\_t](#)  
*Methods that should be implemented by a mobility model.*
- typedef struct [\\_energy\\_methods](#) [energy\\_methods\\_t](#)  
*Methods that should be implemented by an energy model.*
- typedef struct [\\_antenna\\_methods](#) [antenna\\_methods\\_t](#)  
*Methods that should be implemented by an antenna model.*
- typedef struct [\\_radio\\_methods](#) [radio\\_methods\\_t](#)  
*Methods that should be implemented by a radio model.*
- typedef struct [\\_mac\\_methods](#) [mac\\_methods\\_t](#)  
*Methods that should be implemented by a mac model.*
- typedef struct [\\_routing\\_methods](#) [routing\\_methods\\_t](#)  
*Methods that should be implemented by a routing model.*
- typedef struct [\\_application\\_methods](#) [application\\_methods\\_t](#)  
*Methods that should be implemented by an application model.*

### 7.12.1 Detailed Description

Models declarations.

#### Author

Guillaume Chelius & Elyes Ben Hamida & Quentin Lampin

#### Date

2007

Definition in file [models.h](#).

### 7.12.2 Define Documentation

#### 7.12.2.1 `#define MODELTYPE_ANTENNA 8`

Definition at line 21 of file models.h.

#### 7.12.2.2 `#define MODELTYPE_APPLICATION 12`

Definition at line 25 of file models.h.

#### 7.12.2.3 `#define MODELTYPE_ENERGY 13`

Definition at line 26 of file models.h.

#### 7.12.2.4 `#define MODELTYPE_ENVIRONMENT 6`

Definition at line 19 of file models.h.

#### 7.12.2.5 `#define MODELTYPE_FADING 2`

Definition at line 15 of file models.h.

#### 7.12.2.6 `#define MODELTYPE_INTERFERENCES 3`

Definition at line 16 of file models.h.

#### 7.12.2.7 `#define MODELTYPE_MAC 10`

Definition at line 23 of file models.h.

#### 7.12.2.8 `#define MODELTYPE_MOBILITY 7`

Definition at line 20 of file models.h.

#### 7.12.2.9 `#define MODELTYPE_MODULATION 5`

Definition at line 18 of file models.h.

#### 7.12.2.10 `#define MODELTYPE_MONITOR 14`

Definition at line 27 of file models.h.

#### 7.12.2.11 `#define MODELTYPE_NOISE 4`

Definition at line 17 of file models.h.

#### 7.12.2.12 `#define MODELTYPE_PROPAGATION 0`

Definition at line 13 of file models.h.

#### 7.12.2.13 `#define MODELTYPE_RADIO 9`

Definition at line 22 of file models.h.

#### 7.12.2.14 `#define MODELTYPE_ROUTING 11`

Definition at line 24 of file models.h.

#### 7.12.2.15 `#define MODELTYPE_SHADOWING 1`

Definition at line 14 of file models.h.

### 7.12.3 Typedef Documentation

#### 7.12.3.1 `antenna_methods_t`

Methods that should be implemented by an antenna model.

#### 7.12.3.2 `application_methods_t`

Methods that should be implemented by an application model.

**7.12.3.3 energy\_methods\_t**

Methods that should be implemented by an energy model.

**7.12.3.4 environment\_methods\_t**

Methods that should be implemented by an environment model.

**7.12.3.5 fading\_methods\_t**

Methods that should be implemented by a fading model.

**7.12.3.6 interferences\_methods\_t**

Methods that should be implemented by a interferences model.

**7.12.3.7 mac\_methods\_t**

Methods that should be implemented by a mac model.

**7.12.3.8 mobility\_methods\_t**

Methods that should be implemented by a mobility model.

**7.12.3.9 model\_t**

Information about a model.

**7.12.3.10 modulation\_methods\_t**

Methods that should be implemented by a modulation model.

**7.12.3.11 monitor\_methods\_t**

Methods that should be implemented by a monitor model.

**7.12.3.12 noise\_methods\_t**

Methods that should be implemented by a noise model.

#### 7.12.3.13 propagation\_methods\_t

Methods that should be implemented by a propagation model.

#### 7.12.3.14 radio\_methods\_t

Methods that should be implemented by a radio model.

#### 7.12.3.15 routing\_methods\_t

Methods that should be implemented by a routing model.

#### 7.12.3.16 shadowing\_methods\_t

Methods that should be implemented by a shadowing model.

## 7.13 modelutils.h File Reference

Utility function declarations.

```
#include <stdlib.h> #include <stdio.h> #include <string.-
h> #include <float.h> #include <limits.h> #include <include/options.-
h> #include <include/log.h> #include <include/mem_fs.-
h> #include <include/das.h> #include <include/sodas.h> ×
#include <include/spadas.h> #include <include/hadas.h>
#include <include/timer.h> #include <include/types.h> ×
#include <include/models.h> #include <include/rng.h> ×
#include <include/probabilistic_distribution.h> #include
<include/medium.h> #include <include/radio.h> #include
<include/antenna.h> #include <include/battery.h> #include
<include/entity.h> #include <include/packet.h> #include
<include/node.h> #include <include/param.h> #include
<include/measure.h> #include <include/scheduler.h> ×
#include <include/monitor.h> #include <include/ioctl_-
message.h> #include <include/modulation.h>
```

### Defines

- #define [BROADCAST\\_ADDR](#) -1  
*Broadcast address.*

### Functions

- void [end\\_simulation](#) (void)

*Stop the simulation.*

- uint64\_t [get\\_time](#) (void)

*Return the current simulation time.*

- int [get\\_node\\_count](#) (void)

*Return the number of simulated nodes.*

- position\_t \* [get\\_topology\\_area](#) (void)

*Return the size of the network area.*

- void TX ([call\\_t](#) \*c, [packet\\_t](#) \*packet)
- void RX ([call\\_t](#) \*c, [packet\\_t](#) \*packet)
- int IOCTL ([call\\_t](#) \*c, int option, void \*in, void \*\*out)
- int SET\_HEADER ([call\\_t](#) \*c, [packet\\_t](#) \*packet, [destination\\_t](#) \*dst)
- int GET\_HEADER\_SIZE ([call\\_t](#) \*c)
- int GET\_HEADER\_REAL\_SIZE ([call\\_t](#) \*c)

### 7.13.1 Detailed Description

Utility function declarations.

#### Author

Guillaume Chelius & Elyes Ben Hamida & Quentin Lampin

#### Date

2007

Definition in file [modelutils.h](#).

### 7.13.2 Define Documentation

#### 7.13.2.1 #define BROADCAST\_ADDR -1

Broadcast address.

Definition at line 45 of file [modelutils.h](#).

### 7.13.3 Function Documentation

#### 7.13.3.1 void end\_simulation ( void )

Stop the simulation.



7.13.3.2 int GET\_HEADER\_REAL\_SIZE ( call\_t \* c )

7.13.3.3 int GET\_HEADER\_SIZE ( call\_t \* c )

7.13.3.4 int get\_node\_count ( void )

Return the number of simulated nodes.

#### Returns

The number of nodes.

7.13.3.5 uint64\_t get\_time ( void )

Return the current simulation time.

#### Returns

The simulation time.

7.13.3.6 position\_t\* get\_topology\_area ( void )

Return the size of the network area.

#### Returns

The network area.

7.13.3.7 int IOCTL ( call\_t \* c, int option, void \* in, void \*\* out )

7.13.3.8 void RX ( call\_t \* c, packet\_t \* packet )

7.13.3.9 int SET\_HEADER ( call\_t \* c, packet\_t \* packet, destination\_t \* dst )

7.13.3.10 void TX ( call\_t \* c, packet\_t \* packet )

## 7.14 modulation.h File Reference

Modulation declarations.

```
#include <include/types.h>
```

#### Functions

- int modulation\_bit\_per\_symbol (entityid\_t modulation)  
*Returns the bit per symbol rate for a given modulation scheme.*

### 7.14.1 Detailed Description

Modulation declarations.

#### Author

Guillaume Chelius & Elyes Ben Hamida & Christophe Savigny

#### Date

2009

Definition in file [modulation.h](#).

### 7.14.2 Function Documentation

#### 7.14.2.1 `int modulation_bit_per_symbol ( entityid_t modulation )`

Returns the bit per symbol rate for a given modulation scheme.

#### Parameters

<code>modulation</code>	: modulation id.
-------------------------	------------------

#### Returns

The bit per symbol.

## 7.15 monitor.h File Reference

Monitor declarations.

```
#include <include/types.h>
```

### Functions

- void [monitor\\_simulation](#) (void)  
*Call the "monitor\_event()" function of the monitoring entity.*
- void [monitor\\_register\\_callback](#) ([callback\\_t](#) callback, [call\\_t](#) \*c, void \*arg)  
*Register a function that is called whenever the network is monitored.*

### 7.15.1 Detailed Description

Monitor declarations.

**Author**

Guillaume Chelius & Elyes Ben Hamida

**Date**

2007

Definition in file [monitor.h](#).

**7.15.2 Function Documentation****7.15.2.1 void monitor\_register\_callback ( callback\_t callback, call\_t \* c, void \* arg )**

Register a function that is called whenever the network is monitored.

**Parameters**

<i>callback</i>	the function that is called back.
<i>c</i>	the call parameter given to the callback function.
<i>arg</i>	a paramater that given to the callback function.

**7.15.2.2 void monitor\_simulation ( void )**

Call the "monitor\_event()" function of the monitoring entity.

**7.16 node.h File Reference**

Node declarations.

```
#include <include/types.h>
```

**Functions**

- double [distance](#) ([position\\_t](#) \*position0, [position\\_t](#) \*position1)  
*Return the distance between two points.*
- [position\\_t](#) \* [get\\_node\\_position](#) ([nodeid\\_t](#) node)  
*Return a node's position.*
- void [node\\_kill](#) ([nodeid\\_t](#) id)  
*Kill a node during the simulation.*
- int [is\\_node\\_alive](#) ([nodeid\\_t](#) id)  
*Check wether a node is alive.*

### 7.16.1 Detailed Description

Node declarations.

#### Author

Guillaume Chelius & Elyes Ben Hamida

#### Date

2007

Definition in file [node.h](#).

### 7.16.2 Function Documentation

#### 7.16.2.1 `double distance ( position_t * position0, position_t * position1 )`

Return the distance between two points.

##### Parameters

<i>position0</i>	first point.
<i>position1</i>	second point.

##### Returns

The distance between point0 and point1.

#### 7.16.2.2 `position_t* get_node_position ( nodeid_t node )`

Return a node's position.

For read-only, the position should not be modified.

##### Parameters

<i>node</i>	the node id.
-------------	--------------

##### Returns

The node's position.

#### 7.16.2.3 `int is_node_alive ( nodeid_t id )`

Check whether a node is alive.

## Parameters

<i>id</i>	the node id.
-----------	--------------

## Returns

1 if the node is alive, 0 otherwise.

## 7.16.2.4 void node\_kill ( nodeid\_t id )

Kill a node during the simulation.

## Parameters

<i>id</i>	the node id.
-----------	--------------

## 7.17 options.h File Reference

User options declarations.

## Defines

- #define CHANNELS\_NUMBER 1  
*Define the number of simulated radio channels.*
- #define SNR\_STEP 1  
*Define the interference support policy.*
- #define SNR\_ERRORS 0  
*Define the packet error policy.*
- #define LOG\_REPLAY
- #define LOG\_APPLICATION
- #define LOG\_MAC
- #define LOG\_WORLDSSENS

### 7.17.1 Detailed Description

User options declarations.

## Author

Guillaume Chelius & Elyes Ben Hamida

## Date

2007

Definition in file [options.h](#).

## 7.17.2 Define Documentation

### 7.17.2.1 `#define CHANNELS_NUMBER 1`

Define the number of simulated radio channels.

Definition at line 14 of file options.h.

### 7.17.2.2 `#define LOG_APPLICATION`

Definition at line 32 of file options.h.

### 7.17.2.3 `#define LOG_MAC`

Definition at line 34 of file options.h.

### 7.17.2.4 `#define LOG_REPLAY`

Definition at line 31 of file options.h.

### 7.17.2.5 `#define LOG_WORLDSENS`

Definition at line 44 of file options.h.

### 7.17.2.6 `#define SNR_ERRORS 0`

Define the packet error policy.

0 = no error introduction in packet data, 1 = error introduction in the packet data according to the computed BER.

Definition at line 25 of file options.h.

### 7.17.2.7 `#define SNR_STEP 1`

Define the interference support policy.

0 = no interference, -1 = SINR computation for each packet byte, x = packet divided in x slices for SINR computation.

Definition at line 20 of file options.h.

## 7.18 packet.h File Reference

Packet declarations.

```
#include <include/types.h>
```

## Functions

- void [packet\\_dealloc](#) ([packet\\_t](#) \*packet)  
*Deallocate a packet.*
- [packet\\_t](#) \* [packet\\_clone](#) ([packet\\_t](#) \*packet)  
*Duplicate a packet.*
- [packet\\_t](#) \* [packet\\_create](#) ([call\\_t](#) \*c, int size, int real\_size)  
*Allocate a packet with real size argument.*
- [packet\\_t](#) \* [packet\\_alloc](#) ([call\\_t](#) \*c, int size)  
*Allocate a packet - DEPRECATED, use packet\_create instead.*

### 7.18.1 Detailed Description

Packet declarations.

#### Author

Guillaume Chelius & Elyes Ben Hamida & Quentin Lampin

#### Date

2007

Definition in file [packet.h](#).

### 7.18.2 Function Documentation

#### 7.18.2.1 [packet\\_t](#)\* [packet\\_alloc](#) ( [call\\_t](#) \* c, int size )

Allocate a packet - DEPRECATED, use [packet\\_create](#) instead.

#### Parameters

<i>c</i>	should be {-1, node id, -1}.
<i>size</i>	the data size of the packet (in bytes).

#### Returns

The newly allocated packet.

#### 7.18.2.2 [packet\\_t](#)\* [packet\\_clone](#) ( [packet\\_t](#) \* packet )

Duplicate a packet.

Data and other information are copied.

#### Parameters

<i>packet</i>	the packet to clone.
---------------	----------------------

#### Returns

The cloned packet.

#### 7.18.2.3 `packet_t* packet_create ( call_t * c, int size, int real_size )`

Allocate a packet with real size argument.

#### Parameters

<i>c</i>	should be {-1, node id, -1}.
<i>size</i>	the data size of the packet (in bytes).
<i>real_size</i>	real size in bits of the packet.

#### Returns

The newly allocated packet.

#### 7.18.2.4 `void packet_dealloc ( packet_t * packet )`

Deallocate a packet.

#### Parameters

<i>packet</i>	the packet to dealloc.
---------------	------------------------

## 7.19 param.h File Reference

```
#include <include/types.h>
```

#### Functions

- int [get\\_param\\_x\\_position](#) (char \*value, double \*position)  
*Parse a x axis position parameter.*
- int [get\\_param\\_y\\_position](#) (char \*value, double \*position)  
*Parse a y axis position parameter.*
- int [get\\_param\\_z\\_position](#) (char \*value, double \*position)  
*Parse a z axis position parameter.*



- int [get\\_param\\_distance](#) (char \*value, double \*distance)  
*Parse a distance parameter.*
- int [get\\_param\\_double](#) (char \*value, double \*res)  
*Parse a double parameter.*
- int [get\\_param\\_double\\_range](#) (char \*value, double \*res, double min, double max)  
*Parse a double parameter.*
- int [get\\_param\\_time](#) (char \*value, uint64\_t \*time)  
*Parse a time parameter.*
- int [get\\_param\\_integer](#) (char \*value, int \*integer)  
*Parse an integer parameter.*
- int [get\\_param\\_nodeid](#) (char \*value, [nodeid\\_t](#) \*node, [nodeid\\_t](#) myself)  
*Parse a node id parameter.*
- int [get\\_param\\_entity](#) (char \*value, [entityid\\_t](#) \*entity)  
*Parse an entity parameter.*

### 7.19.1 Function Documentation

#### 7.19.1.1 int [get\\_param\\_distance](#) ( char \* value, double \* distance )

Parse a distance parameter.

Accept "random" parameter value.

##### Parameters

<i>value</i>	the parameter value.
<i>distance</i>	a pointer to a double that will be filled.

##### Returns

0 upon success, -1 otherwise.

#### 7.19.1.2 int [get\\_param\\_double](#) ( char \* value, double \* res )

Parse a double parameter.

Accept "random" parameter value.

##### Parameters

<i>value</i>	the parameter value.
<i>res</i>	a pointer to a double that will be filled.

**Returns**

0 upon success, -1 otherwise.

**7.19.1.3** `int get_param_double_range ( char * value, double * res, double min, double max )`

Parse a double parameter.

Accept "random" parameter value. The parsed value is checked to be in [min,max].

**Parameters**

<i>value</i>	the parameter value.
<i>res</i>	a pointer to a double that will be filled.
<i>min</i>	the minimum accepted value.
<i>max</i>	the maximum accepted value.

**Returns**

0 upon success, -1 otherwise.

**7.19.1.4** `int get_param_entity ( char * value, entityid_t * entity )`

Parse an entity parameter.

**Parameters**

<i>value</i>	the parameter value.
<i>entity</i>	a pointer to an entity id that will be filled.

**Returns**

0 upon success, -1 otherwise.

**7.19.1.5** `int get_param_integer ( char * value, int * integer )`

Parse an integer parameter.

Accept "random" parameter value.

**Parameters**

<i>value</i>	the parameter value.
<i>integer</i>	a pointer to an integer that will be filled.

**Returns**

0 upon success, -1 otherwise.

**7.19.1.6 int get\_param\_nodeid ( char \* value, nodeid\_t \* node, nodeid\_t myself )**

Parse a node id parameter.

Accept "random" parameter value. The returned param value will be different than "myself".

**Parameters**

<i>value</i>	the parameter value.
<i>node</i>	a pointer to a node id that will be filled.
<i>myself</i>	a node id to exclude.

**Returns**

0 upon success, -1 otherwise.

**7.19.1.7 int get\_param\_time ( char \* value, uint64\_t \* time )**

Parse a time parameter.

Accept "random" parameter value.

**Parameters**

<i>value</i>	the parameter value.
<i>time</i>	a pointer to a uin64 that will be filled.

**Returns**

0 upon success, -1 otherwise.

**7.19.1.8 int get\_param\_x\_position ( char \* value, double \* position )**

Parse a x axis position parameter.

Accept "random" parameter value.

**Parameters**

<i>value</i>	the parameter value.
<i>position</i>	a pointer to a double that will be filled.

**Returns**

0 upon success, -1 otherwise.

**7.19.1.9 int get\_param\_y\_position ( char \* value, double \* position )**

Parse a y axis position parameter.

Accept "random" parameter value.

**Parameters**

<i>value</i>	the parameter value.
<i>position</i>	a pointer to a double that will be filled.

**Returns**

0 upon success, -1 otherwise.

**7.19.1.10 int get\_param\_z\_position ( char \* value, double \* position )**

Parse a z axis position parameter.

Accept "random" parameter value.

**Parameters**

<i>value</i>	the parameter value.
<i>position</i>	a pointer to a double that will be filled.

**Returns**

0 upon success, -1 otherwise.

## 7.20 probabilistic\_distribution.h File Reference

Probabilistic distributions declarations.

**Data Structures**

- struct [gaussian\\_args\\_s](#)
- struct [gaussian\\_tail\\_args\\_s](#)
- struct [bivariate\\_gaussian\\_args\\_s](#)
- struct [exponential\\_args\\_s](#)
- struct [laplace\\_args\\_s](#)
- struct [exppow\\_args\\_s](#)

- struct [cauchy\\_args\\_s](#)
- struct [rayleigh\\_s](#)
- struct [rayleigh\\_tail\\_s](#)
- struct [levy\\_alpha\\_stable\\_s](#)
- struct [gamma\\_args\\_s](#)
- struct [uniform\\_args\\_s](#)
- struct [lognormal\\_args\\_s](#)
- struct [chisq\\_args\\_s](#)
- struct [beta\\_args\\_s](#)
- struct [logistic\\_args\\_s](#)
- struct [pareto\\_args\\_s](#)
- struct [spherical\\_vector\\_2d\\_args\\_s](#)
- struct [spherical\\_vector\\_3d\\_args\\_s](#)
- struct [weibull\\_args\\_s](#)
- struct [gumbel\\_t1\\_args\\_s](#)
- struct [gumbel\\_t2\\_args\\_s](#)
- struct [poisson\\_args\\_s](#)
- struct [bernoulli\\_args\\_s](#)
- struct [binomial\\_args\\_s](#)
- struct [geometric\\_args\\_s](#)
- struct [hyper\\_geometric\\_args\\_s](#)
- struct [logarithmic\\_args\\_s](#)

## Defines

- #define [GAUSSIAN](#) 0
- #define [GAUSSIAN\\_TAIL](#) 1
- #define [EXPONENTIAL](#) 2
- #define [LAPLACE](#) 3
- #define [EXPONENTIAL\\_POWER](#) 4
- #define [CAUCHY](#) 5
- #define [RAYLEIGH](#) 6
- #define [RAYLEIGH\\_TAIL](#) 7
- #define [LANDAU](#) 8
- #define [LEVY\\_ALPHA\\_STABLE](#) 9
- #define [GAMMA](#) 10
- #define [UNIFORM](#) 11
- #define [LOG\\_NORMAL](#) 12
- #define [CHI\\_SQUARED](#) 13
- #define [BETA](#) 14
- #define [LOGISTIC](#) 15
- #define [PARETO](#) 16
- #define [WEIBULL](#) 17
- #define [GUMBELL\\_1](#) 18
- #define [GUMBELL\\_2](#) 19
- #define [POISSON](#) 20

- #define [BERNOULLI](#) 21
- #define [BINOMIAL](#) 22
- #define [GEOMETRIC](#) 23
- #define [HYPERGEOMETRIC](#) 24
- #define [LOGARITHMIC](#) 25

## Typedefs

- typedef struct [gaussian\\_args\\_s](#) [gaussian\\_args\\_t](#)
- typedef struct [gaussian\\_tail\\_args\\_s](#) [gaussian\\_tail\\_args\\_t](#)
- typedef struct [bivariate\\_gaussian\\_args\\_s](#) [bivariate\\_gaussian\\_args\\_t](#)
- typedef struct [exponential\\_args\\_s](#) [exponential\\_args\\_t](#)
- typedef struct [laplace\\_args\\_s](#) [laplace\\_args\\_t](#)
- typedef struct [exppow\\_args\\_s](#) [exppow\\_args\\_t](#)
- typedef struct [cauchy\\_args\\_s](#) [cauchy\\_args\\_t](#)
- typedef struct [rayleigh\\_s](#) [rayleigh\\_args\\_t](#)
- typedef struct [rayleigh\\_tail\\_s](#) [rayleigh\\_tail\\_args\\_t](#)
- typedef struct [levy\\_alpha\\_stable\\_s](#) [levy\\_alpha\\_stable\\_args\\_t](#)
- typedef struct [gamma\\_args\\_s](#) [gamma\\_args\\_t](#)
- typedef struct [uniform\\_args\\_s](#) [uniform\\_args\\_t](#)
- typedef struct [lognormal\\_args\\_s](#) [lognormal\\_args\\_t](#)
- typedef struct [chisq\\_args\\_s](#) [chisq\\_args\\_t](#)
- typedef struct [beta\\_args\\_s](#) [beta\\_args\\_t](#)
- typedef struct [logistic\\_args\\_s](#) [logistic\\_args\\_t](#)
- typedef struct [pareto\\_args\\_s](#) [pareto\\_args\\_t](#)
- typedef struct [spherical\\_vector\\_2d\\_args\\_s](#) [spherical\\_vector\\_2d\\_args\\_t](#)
- typedef struct [spherical\\_vector\\_3d\\_args\\_s](#) [spherical\\_vector\\_3d\\_args\\_t](#)
- typedef struct [weibull\\_args\\_s](#) [weibull\\_args\\_t](#)
- typedef struct [gumbel\\_t1\\_args\\_s](#) [gumbel\\_t1\\_args\\_t](#)
- typedef struct [gumbel\\_t2\\_args\\_s](#) [gumbel\\_t2\\_args\\_t](#)
- typedef struct [poisson\\_args\\_s](#) [poisson\\_args\\_t](#)
- typedef struct [bernoulli\\_args\\_s](#) [bernoulli\\_args\\_t](#)
- typedef struct [binomial\\_args\\_s](#) [binomial\\_args\\_t](#)
- typedef struct [geometric\\_args\\_s](#) [geometric\\_args\\_t](#)
- typedef struct [hyper\\_geometric\\_args\\_s](#) [hyper\\_geometric\\_args\\_t](#)
- typedef struct [logarithmic\\_args\\_s](#) [logarithmic\\_args\\_t](#)
- typedef double(\* [distribution\\_function\\_t](#))(void \*rng\_id, void \*parameters)

## Functions

- [distribution\\_function\\_t](#) [get\\_distribution\\_function\\_by\\_type](#) (int type)  
*Return a pointer to the distribution function according to type type of the distribution.*

### 7.20.1 Detailed Description

Probabilistic distributions declarations.

**Author**

Quentin Lampin

**Date**

2009

Definition in file [probabilistic\\_distribution.h](#).

### 7.20.2 Define Documentation

#### 7.20.2.1 `#define` **BERNOULLI** 21

Definition at line 33 of file probabilistic\_distribution.h.

#### 7.20.2.2 `#define` **BETA** 14

Definition at line 26 of file probabilistic\_distribution.h.

#### 7.20.2.3 `#define` **BINOMIAL** 22

Definition at line 34 of file probabilistic\_distribution.h.

#### 7.20.2.4 `#define` **CAUCHY** 5

Definition at line 17 of file probabilistic\_distribution.h.

#### 7.20.2.5 `#define` **CHI\_SQUARED** 13

Definition at line 25 of file probabilistic\_distribution.h.

#### 7.20.2.6 `#define` **EXPONENTIAL** 2

Definition at line 14 of file probabilistic\_distribution.h.

#### 7.20.2.7 `#define` **EXPONENTIAL\_POWER** 4

Definition at line 16 of file probabilistic\_distribution.h.

**7.20.2.8 #define GAMMA 10**

Definition at line 22 of file probabilistic\_distribution.h.

**7.20.2.9 #define GAUSSIAN 0**

Definition at line 12 of file probabilistic\_distribution.h.

**7.20.2.10 #define GAUSSIAN\_TAIL 1**

Definition at line 13 of file probabilistic\_distribution.h.

**7.20.2.11 #define GEOMETRIC 23**

Definition at line 35 of file probabilistic\_distribution.h.

**7.20.2.12 #define GUMBELL\_1 18**

Definition at line 30 of file probabilistic\_distribution.h.

**7.20.2.13 #define GUMBELL\_2 19**

Definition at line 31 of file probabilistic\_distribution.h.

**7.20.2.14 #define HYPERGEOMETRIC 24**

Definition at line 36 of file probabilistic\_distribution.h.

**7.20.2.15 #define LANDAU 8**

Definition at line 20 of file probabilistic\_distribution.h.

**7.20.2.16 #define LAPLACE 3**

Definition at line 15 of file probabilistic\_distribution.h.

**7.20.2.17 #define LEVY\_ALPHA\_STABLE 9**

Definition at line 21 of file probabilistic\_distribution.h.



**7.20.2.18 #define LOG\_NORMAL 12**

Definition at line 24 of file probabilistic\_distribution.h.

**7.20.2.19 #define LOGARITHMIC 25**

Definition at line 37 of file probabilistic\_distribution.h.

**7.20.2.20 #define LOGISTIC 15**

Definition at line 27 of file probabilistic\_distribution.h.

**7.20.2.21 #define PARETO 16**

Definition at line 28 of file probabilistic\_distribution.h.

**7.20.2.22 #define POISSON 20**

Definition at line 32 of file probabilistic\_distribution.h.

**7.20.2.23 #define RAYLEIGH 6**

Definition at line 18 of file probabilistic\_distribution.h.

**7.20.2.24 #define RAYLEIGH\_TAIL 7**

Definition at line 19 of file probabilistic\_distribution.h.

**7.20.2.25 #define UNIFORM 11**

Definition at line 23 of file probabilistic\_distribution.h.

**7.20.2.26 #define WEIBULL 17**

Definition at line 29 of file probabilistic\_distribution.h.

**7.20.3 Typedef Documentation****7.20.3.1 typedef struct bernoulli\_args\_s bernoulli\_args\_t****7.20.3.2 typedef struct beta\_args\_s beta\_args\_t**

7.20.3.3 `typedef struct binomial_args_s binomial_args_t`

7.20.3.4 `typedef struct bivariate_gaussian_args_s bivariate_gaussian_args_t`

7.20.3.5 `typedef struct cauchy_args_s cauchy_args_t`

7.20.3.6 `typedef struct chisq_args_s chisq_args_t`

7.20.3.7 `typedef double(* distribution_function_t)(void *rng_id, void *parameters)`

Definition at line 174 of file probabilistic\_distribution.h.

7.20.3.8 `typedef struct exponential_args_s exponential_args_t`

7.20.3.9 `typedef struct exppow_args_s exppow_args_t`

7.20.3.10 `typedef struct gamma_args_s gamma_args_t`

7.20.3.11 `typedef struct gaussian_args_s gaussian_args_t`

7.20.3.12 `typedef struct gaussian_tail_args_s gaussian_tail_args_t`

7.20.3.13 `typedef struct geometric_args_s geometric_args_t`

7.20.3.14 `typedef struct gumbel_t1_args_s gumbel_t1_args_t`

7.20.3.15 `typedef struct gumbel_t2_args_s gumbel_t2_args_t`

7.20.3.16 `typedef struct hyper_geometric_args_s hyper_geometric_args_t`

7.20.3.17 `typedef struct laplace_args_s laplace_args_t`

7.20.3.18 `typedef struct levy_alpha_stable_s levy_alpha_stable_args_t`

7.20.3.19 `typedef struct logarithmic_args_s logarithmic_args_t`

7.20.3.20 `typedef struct logistic_args_s logistic_args_t`

7.20.3.21 `typedef struct lognormal_args_s lognormal_args_t`

7.20.3.22 `typedef struct pareto_args_s pareto_args_t`

7.20.3.23 `typedef struct poisson_args_s poisson_args_t`

7.20.3.24 `typedef struct rayleigh_s rayleigh_args_t`

7.20.3.25 `typedef struct rayleigh_tail_s rayleigh_tail_args_t`

7.20.3.26 `typedef struct spherical_vector_2d_args_s spherical_vector_2d_args_t`

7.20.3.27 `typedef struct spherical_vector_3d_args_s spherical_vector_3d_args_t`

7.20.3.28 `typedef struct uniform_args_s uniform_args_t`

7.20.3.29 `typedef struct weibull_args_s weibull_args_t`

## 7.20.4 Function Documentation

7.20.4.1 `distribution_function_t get_distribution_function_by_type ( int type )`

Return a pointer to the distribution function according to type type of the distribution.

### Returns

a function pointer

## 7.21 radio.h File Reference

Radio declarations.

```
#include <include/types.h>
```

### Functions

- double `radio_get_cs` (`call_t *c`)  
*Carrier Sense mechanism.*
- double `radio_get_noise` (`call_t *c`)  
*Clear Channel Assesment mechanism.*
- double `radio_get_power` (`call_t *c`)  
*Get the radio tx power.*
- void `radio_set_power` (`call_t *c`, double power)  
*Set the radio tx power.*
- int `radio_get_channel` (`call_t *c`)  
*Get the radio channel.*
- void `radio_set_channel` (`call_t *c`, int channel)  
*Set the radio channel.*
- `entityid_t` `radio_get_modulation` (`call_t *c`)  
*Get the radio modulation.*
- void `radio_set_modulation` (`call_t *c`, `entityid_t` modulation)  
*Set the radio modulation.*
- `uint64_t` `radio_get_Tb` (`call_t *c`)  
*Get the radio bandwidth.*
- void `radio_cs` (`call_t *c`, `packet_t *packet`)

*Notifies the radio with a new signal.*

- void [radio\\_set\\_sensibility](#) ([call\\_t](#) \*c, double sensibility)

*Set the radio sensibility.*

- double [radio\\_get\\_sensibility](#) ([call\\_t](#) \*c)

*Get the radio sensibility.*

- void [radio\\_sleep](#) ([call\\_t](#) \*c)

*Set the radio in sleep mode.*

- void [radio\\_wakeup](#) ([call\\_t](#) \*c)

*Set the radio in active mode.*

- int [radio\\_get\\_modulation\\_bit\\_per\\_symbol](#) ([call\\_t](#) \*c)

*get the number of bit per symbol for modulation associated*

- uint64\_t [radio\\_get\\_Ts](#) ([call\\_t](#) \*c)

*Get the radio bandwidth.*

- void [radio\\_set\\_Ts](#) ([call\\_t](#) \*c, uint64\_t Ts)

*Set the radio bandwidth.*

### 7.21.1 Detailed Description

Radio declarations.

#### Author

Guillaume Chelius & Elyes Ben Hamida

#### Date

2007

Definition in file [radio.h](#).

### 7.21.2 Function Documentation

#### 7.21.2.1 void [radio\\_cs](#) ( [call\\_t](#) \* c, [packet\\_t](#) \* packet )

Notifies the radio with a new signal.

#### Parameters

<i>c</i>	should be {radio id, node id, antenna id}.
<i>packet</i>	the new signal.

#### 7.21.2.2 int [radio\\_get\\_channel](#) ( [call\\_t](#) \* c )

Get the radio channel.

## Parameters

<code>c</code>	should be {radio id, node id, -1}.
----------------	------------------------------------

## Returns

The current radio channel.

7.21.2.3 `double radio_get_cs ( call_t * c )`

Carrier Sense mechanism.

## Parameters

<code>c</code>	should be {radio id, node id, -1}.
----------------	------------------------------------

## Returns

The signal strength of the currently received signal. MIN\_DBM if no current signal.

7.21.2.4 `entityid_t radio_get_modulation ( call_t * c )`

Get the radio modulation.

## Parameters

<code>c</code>	should be {radio id, node id, -1}.
----------------	------------------------------------

## Returns

The current modulation entity id.

7.21.2.5 `int radio_get_modulation_bit_per_symbol ( call_t * c )`

get the number of bit per symbol for modulation associated

## Parameters

<code>c</code>	should be {radio id, node id, -1}.
----------------	------------------------------------

## Returns

the current number of bit per symbol

### 7.21.2.6 double radio\_get\_noise ( call\_t \* c )

Clear Channel Assesment mechanism.

#### Parameters

<i>c</i>	should be {radio id, node id, -1}.
----------	------------------------------------

#### Returns

The noise strength on the radio interface. MIN\_DBM if no noise.

### 7.21.2.7 double radio\_get\_power ( call\_t \* c )

Get the radio tx power.

#### Parameters

<i>c</i>	should be {radio id, node id, -1}.
----------	------------------------------------

#### Returns

The current transmission power in dBm.

### 7.21.2.8 double radio\_get\_sensibility ( call\_t \* c )

Get the radio sensibility.

#### Parameters

<i>c</i>	should be {radio id, node id, -1}.
----------	------------------------------------

#### Returns

The current radio sensibility in dBm.

### 7.21.2.9 uint64\_t radio\_get\_Tb ( call\_t \* c )

Get the radio bandwidth.

#### Parameters

<i>c</i>	should be {radio id, node id, -1}.
----------	------------------------------------

**Returns**

The current bandwidth, i.e. the time to transmit a bit in ns.

**7.21.2.10 uint64\_t radio\_get\_Ts ( call\_t \* c )**

Get the radio bandwidth.

**Parameters**

<i>c</i>	should be {radio id, node id, -1}.
----------	------------------------------------

**Returns**

The current bandwidth, i.e. the time to transmit a symbol in ns.

**7.21.2.11 void radio\_set\_channel ( call\_t \* c, int channel )**

Set the radio channel.

**Parameters**

<i>c</i>	should be {radio id, node id, -1}.
<i>channel</i>	the new radio channel.

**7.21.2.12 void radio\_set\_modulation ( call\_t \* c, entityid\_t modulation )**

Set the radio modulation.

**Parameters**

<i>c</i>	should be {radio id, node id, -1}.
<i>modulation</i>	the new modulation entity id.

**7.21.2.13 void radio\_set\_power ( call\_t \* c, double power )**

Set the radio tx power.

**Parameters**

<i>c</i>	should be {radio id, node id, -1}.
<i>power</i>	the new transmission power in dBm.

#### 7.21.2.14 void radio\_set\_sensibility ( call\_t \* c, double sensibility )

Set the radio sensibility.

##### Parameters

<i>c</i>	should be {radio id, node id, -1}.
<i>sensibility</i>	the new radio sensibility in dBm.

#### 7.21.2.15 void radio\_set\_Ts ( call\_t \* c, uint64\_t Ts )

Set the radio bandwidth.

Ts must be a bit\_per\_symbol's multiple

##### Parameters

<i>c</i>	should be {radio id, node id, -1}.
<i>Ts</i>	the new bandwidth, i.e. the time to transmit a symbol in ns.

#### 7.21.2.16 void radio\_sleep ( call\_t \* c )

Set the radio in sleep mode.

##### Parameters

<i>c</i>	should be {radio id, node id, -1}.
----------	------------------------------------

#### 7.21.2.17 void radio\_wakeup ( call\_t \* c )

Set the radio in active mode.

##### Parameters

<i>c</i>	should be {radio id, node id, -1}.
----------	------------------------------------

## 7.22 rng.h File Reference

Random number generator declarations.

```
#include <include/types.h> #include <gsl/gsl_rng.h>
```

### Defines

- #define [DEFAULT\\_RNG](#) 1



- #define [MT19937](#) 1
- #define [RANLXS0](#) 2
- #define [RANLXS1](#) 3
- #define [RANLXS2](#) 4
- #define [RANLXD1](#) 5
- #define [RANLXD2](#) 6
- #define [RANLUX](#) 7
- #define [RANLUX389](#) 8
- #define [RNG\\_DEFAULT\\_RETRY\\_ATTEMPTS](#) -1

## Functions

- `gsl_rng * get\_rng\_by\_id (void *rng_id)`  
*retrieve RNG by ID.*
- `void * create\_rng (int rng_type, unsigned long int seed)`  
*create RNG*
- `double get\_random\_distance\_gsl (void *rng_id, int distribution_type, void *parameters)`  
*Return a random distance.*
- `double get\_random\_x\_position\_gsl (void *rng_id, int distribution_type, void *parameters)`  
*Return a random position on the x dimension.*
- `double get\_random\_y\_position\_gsl (void *rng_id, int distribution_type, void *parameters)`  
*Return a random position on the y dimension.*
- `double get\_random\_z\_position\_gsl (void *rng_id, int distribution_type, void *parameters)`  
*Return a random position on the z dimension.*
- `double get\_random\_double\_gsl (void *rng_id, int distribution_type, void *parameters)`  
*Return a random double value in [0,1].*
- `double get\_random\_double\_range\_gsl (void *rng_id, int distribution_type, void *parameters, double min, double max)`  
*Return a random double value in [min,max].*
- `int get\_random\_integer\_gsl (void *rng_id, int distribution_type, void *parameters)`  
*Return a random integer value.*
- `int get\_random\_integer\_range\_gsl (void *rng_id, int distribution_type, void *parameters, int min, int max)`  
*Return a random integer value in [min,max].*
- `uint64_t get\_random\_time\_gsl (void *rng_id, int distribution_type, void *parameters)`  
*Return a random time.*
- `uint64_t get\_random\_time\_range\_gsl (void *rng_id, int distribution_type, void *parameters, uint64_t min, uint64_t max)`

*Return a random time in [min,max].*

- `nodeid_t get_random_node_gsl` (void \*rng\_id, int distribution\_type, void \*parameters, `nodeid_t` exclusion)

*Return a random node id different from a specified one.*

- double `get_random_distance` (void)

*ALL THE FOLLOWING FUNCTION ARE USED BY THE WSNET KERNEL WITH THE DEFAULT RNG.*

- double `get_random_x_position` (void)

*Return a random position on the x dimension.*

- double `get_random_y_position` (void)

*Return a random position on the y dimension.*

- double `get_random_z_position` (void)

*Return a random position on the z dimension.*

- double `get_random_double` (void)

*Return a random double value in [0,1[.*

- double `get_random_double_range` (double min, double max)

*Return a random double value in [min,max[.*

- int `get_random_integer` (void)

*Return a random integer value.*

- int `get_random_integer_range` (int min, int max)

*Return a random integer value in [min,max[.*

- `uint64_t get_random_time` (void)

*Return a random time in [0,simulation\_end].*

- `uint64_t get_random_time_range` (`uint64_t` min, `uint64_t` max)

*Return a random time in [min,max[.*

- `nodeid_t get_random_node` (`nodeid_t` exclusion)

*Return a random node id different from a specified one.*

### 7.22.1 Detailed Description

Random number generator declarations.

Author

Quentin Lampin

Date

2009

Definition in file `rng.h`.

### 7.22.2 Define Documentation

#### 7.22.2.1 #define **DEFAULT\_RNG** 1

Definition at line 15 of file rng.h.

#### 7.22.2.2 #define **MT19937** 1

Definition at line 18 of file rng.h.

#### 7.22.2.3 #define **RANLUX** 7

Definition at line 24 of file rng.h.

#### 7.22.2.4 #define **RANLUX389** 8

Definition at line 25 of file rng.h.

#### 7.22.2.5 #define **RANLXD1** 5

Definition at line 22 of file rng.h.

#### 7.22.2.6 #define **RANLXD2** 6

Definition at line 23 of file rng.h.

#### 7.22.2.7 #define **RANLXS0** 2

Definition at line 19 of file rng.h.

#### 7.22.2.8 #define **RANLXS1** 3

Definition at line 20 of file rng.h.

#### 7.22.2.9 #define **RANLXS2** 4

Definition at line 21 of file rng.h.

#### 7.22.2.10 #define **RNG\_DEFAULT\_RETRY\_ATTEMPTS** -1

Definition at line 28 of file rng.h.

### 7.22.3 Function Documentation

#### 7.22.3.1 void\* create\_rng ( int rng\_type, unsigned long int seed )

create RNG

##### Parameters

<i>rng_type</i>	the type of the RNG to instanciate
<i>seed</i>	the seed of the RNG

##### Returns

id of the newly created RNG

#### 7.22.3.2 double get\_random\_distance ( void )

ALL THE FOLLOWING FUNCTION ARE USED BY THE WSNET KERNEL WITH THE DEFAULT RNG.

Return a random distance.

##### Returns

A random distance.

#### 7.22.3.3 double get\_random\_distance\_gsl ( void \* rng\_id, int distribution\_type, void \* parameters )

Return a random distance.

##### Parameters

<i>rng_id</i>	ID of the RNG to use.
<i>distribution_type</i>	type of the probability distribution
<i>parameters</i>	the probability distribution parameters

##### Returns

A random distance.

#### 7.22.3.4 double get\_random\_double ( void )

Return a random double value in [0,1[.

**Returns**

A random double value in [0,1[.

**7.22.3.5** `double get_random_double_gsl ( void * rng_id, int distribution_type, void * parameters )`

Return a random double value in [0,1[.

**Parameters**

<i>rng_id</i>	ID of the RNG to use.
<i>distribution_type</i>	type of the probability distribution
<i>parameters</i>	the probability distribution parameters

**Returns**

A random double value in [0,1[.

**7.22.3.6** `double get_random_double_range ( double min, double max )`

Return a random double value in [min,max[.

**Parameters**

<i>min</i>	the min value that can be drawn.
<i>max</i>	the max value that can be drawn.

**Returns**

A random double value in [min,max[.

**7.22.3.7** `double get_random_double_range_gsl ( void * rng_id, int distribution_type, void * parameters, double min, double max )`

Return a random double value in [min,max[.

**Parameters**

<i>rng_id</i>	ID of the RNG to use.
<i>distribution_type</i>	type of the probability distribution
<i>parameters</i>	the probability distribution parameters
<i>min</i>	the min value that can be drawn.
<i>max</i>	the max value that can be drawn.

**Returns**

A random double value in [min,max[.

**7.22.3.8 int get\_random\_integer ( void )**

Return a random integer value.

**Returns**

A random integer value.

**7.22.3.9 int get\_random\_integer\_gsl ( void \* rng\_id, int distribution\_type, void \* parameters )**

Return a random integer value.

**Parameters**

<i>rng_id</i>	ID of the RNG to use.
<i>distribution_type</i>	type of the probability distribution
<i>parameters</i>	the probability distribution parameters

**Returns**

A random integer value.

**7.22.3.10 int get\_random\_integer\_range ( int min, int max )**

Return a random integer value in [min,max].

**Parameters**

<i>min</i>	the min value that can be drawn.
<i>max</i>	the max value that can be drawn.

**Returns**

A random integer value.

**7.22.3.11 int get\_random\_integer\_range\_gsl ( void \* rng\_id, int distribution\_type, void \* parameters, int min, int max )**

Return a random integer value in [min,max].

## Parameters

<i>rng_id</i>	ID of the RNG to use.
<i>distribution_ - type</i>	type of the probability distribution
<i>parameters</i>	the probability distribution parameters
<i>min</i>	the min value that can be drawn.
<i>max</i>	the max value that can be drawn.

## Returns

A random integer value.

7.22.3.12 `nodeid_t get_random_node ( nodeid_t exclusion )`

Return a random node id different from a specified one.

## Parameters

<i>exclusion</i>	the node identifier we do not want to be returned.
------------------	--

## Returns

A random node identifier.

7.22.3.13 `nodeid_t get_random_node_gsl ( void * rng_id, int distribution_type, void * parameters, nodeid_t exclusion )`

Return a random node id different from a specified one.

## Parameters

<i>rng_id</i>	ID of the RNG to use.
<i>distribution_ - type</i>	type of the probability distribution
<i>parameters</i>	the probability distribution parameters
<i>exclusion</i>	the node identifier we do not want to be returned.

## Returns

A random node identifier.

7.22.3.14 `uint64_t get_random_time ( void )`

Return a random time in [0,simulation\_end].

**Returns**

A random time [0,simulation\_end].

**7.22.3.15** `uint64_t get_random_time_gsl ( void * rng_id, int distribution_type, void * parameters )`

Return a random time.

**Parameters**

<i>rng_id</i>	ID of the RNG to use.
<i>distribution_ - type</i>	type of the probability distribution
<i>parameters</i>	the probability distribution parameters

**Returns**

A random time in ns.

**7.22.3.16** `uint64_t get_random_time_range ( uint64_t min, uint64_t max )`

Return a random time in [min,max].

**Parameters**

<i>min</i>	the min value that can be drawn.
<i>max</i>	the max value that can be drawn.

**Returns**

A random time in [min,max].

**7.22.3.17** `uint64_t get_random_time_range_gsl ( void * rng_id, int distribution_type, void * parameters, uint64_t min, uint64_t max )`

Return a random time in [min,max].

**Parameters**

<i>rng_id</i>	ID of the RNG to use.
<i>distribution_ - type</i>	type of the probability distribution
<i>parameters</i>	the probability distribution parameters
<i>min</i>	the min value that can be drawn.
<i>max</i>	the max value that can be drawn.



**Returns**

A random time in [min,max].

**7.22.3.18 double get\_random\_x\_position ( void )**

Return a random position on the x dimension.

**Returns**

A random position on the x dimension.

**7.22.3.19 double get\_random\_x\_position\_gsl ( void \* rng\_id, int distribution\_type, void \* parameters )**

Return a random position on the x dimension.

**Parameters**

<i>rng_id</i>	ID of the RNG to use.
<i>distribution_ - type</i>	type of the probability distribution
<i>parameters</i>	the probability distribution parameters

**Returns**

A random position on the x dimension.

**7.22.3.20 double get\_random\_y\_position ( void )**

Return a random position on the y dimension.

**Returns**

A random position on the y dimension.

**7.22.3.21 double get\_random\_y\_position\_gsl ( void \* rng\_id, int distribution\_type, void \* parameters )**

Return a random position on the y dimension.

**Parameters**

<i>rng_id</i>	ID of the RNG to use.
<i>distribution_ - type</i>	type of the probability distribution
<i>parameters</i>	the probability distribution parameters

**Returns**

A random position on the y dimension.

**7.22.3.22 double get\_random\_z\_position ( void )**

Return a random position on the z dimension.

**Returns**

A random position on the z dimension.

**7.22.3.23 double get\_random\_z\_position\_gsl ( void \* rng\_id, int distribution\_type, void \* parameters )**

Return a random position on the z dimension.

**Parameters**

<i>rng_id</i>	ID of the RNG to use.
<i>distribution_type</i>	type of the probability distribution
<i>parameters</i>	the probability distribution parameters

**Returns**

A random position on the z dimension.

**7.22.3.24 gsl\_rng\* get\_rng\_by\_id ( void \* rng\_id )**

retrieve RNG by ID.

**Parameters**

<i>rng_id</i>	ID of the rng to retrieve.
---------------	----------------------------

**Returns**

pointer to the RNG.

**7.23 scheduler.h File Reference**

Scheduler declarations.

```
#include <include/types.h>
```

## Data Structures

- struct [\\_event](#)  
*A scheduler event.*

## Typedefs

- typedef struct [\\_event](#) [event\\_t](#)  
*A scheduler event.*

## Functions

- [event\\_t](#) \* [scheduler\\_add\\_callback](#) (uint64\_t clock, [call\\_t](#) \*c, [callback\\_t](#) callback, void \*arg)  
*Schedule the callback of a function at a given time.*
- void [scheduler\\_delete\\_callback](#) ([call\\_t](#) \*c, [event\\_t](#) \*event)  
*Delete an event from the Scheduler.*

### 7.23.1 Detailed Description

Scheduler declarations.

#### Author

Guillaume Chelius & Elyes Ben Hamida

#### Date

2007

Definition in file [scheduler.h](#).

### 7.23.2 Typedef Documentation

#### 7.23.2.1 [event\\_t](#)

A scheduler event.

### 7.23.3 Function Documentation

#### 7.23.3.1 [event\\_t](#)\* [scheduler\\_add\\_callback](#) ( [uint64\\_t](#) clock, [call\\_t](#) \* c, [callback\\_t](#) callback, void \* arg )

Schedule the callback of a function at a given time.

## Parameters

<i>clock</i>	time of the callback.
<i>c</i>	the call parameter given to the callback function.
<i>callback</i>	the function that is called back.
<i>arg</i>	a paramater that given to the callback function.

## Returns

An opaque object that references the callback-associated event.

## 7.23.3.2 void scheduler\_delete\_callback ( call\_t \* c, event\_t \* event )

Delete an event from the Scheduler.

## Parameters

<i>c</i>	the call parameter given to the callback function.
<i>event</i>	a paramater that describe the event we want to delete.

## 7.24 sodas.h File Reference

SOrted DAta Structure module declarations.

## Typedefs

- typedef int(\* [sodas\\_compare\\_t](#) )(void \*key0, void \*key1)  
*The prototype of a comparison function.*

## Functions

- int [sodas\\_init](#) (void)  
*Initialize the sodas module.*
- void \* [sodas\\_create](#) ([sodas\\_compare\\_t](#) compare)  
*Create an empty sodas structure.*
- void [sodas\\_destroy](#) (void \*sodas)  
*Destroy a sodas structure.*
- void [sodas\\_insert](#) (void \*sodas, void \*key, void \*data)  
*Insert an object in the sodas structure.*
- void \* [sodas\\_pop](#) (void \*sodas)  
*Remove the first object from the sodas structure.*
- void \* [sodas\\_delete](#) (void \*sodas, void \*key)  
*Remove a particular object from the sodas structure.*

- void \* [sodas\\_see\\_first](#) (void \*sodas)  
*Show the first object in the sodas structure.*

### 7.24.1 Detailed Description

Sorted Data Structure module declarations.

#### Author

Guillaume Chelius & Elyes Ben Hamida

#### Date

2007

Definition in file [sodas.h](#).

### 7.24.2 Typedef Documentation

#### 7.24.2.1 `sodas_compare_t`

The prototype of a comparison function.

Definition at line 14 of file sodas.h.

### 7.24.3 Function Documentation

#### 7.24.3.1 `void* sodas_create ( sodas_compare_t compare )`

Create an empty sodas structure.

#### Parameters

<i>comapre</i>	the comparison function used to sort the objects.
----------------	---

#### Returns

An opaque pointer to the sodas structure.

#### 7.24.3.2 `void* sodas_delete ( void * sodas, void * key )`

Remove a particular object from the sodas structure.

#### Parameters

<i>sodas</i>	the opaque pointer to the data structure.
<i>key</i>	the key of the object to remove.

### 7.24.3.3 void **sodas\_destroy** ( void \* *sodas* )

Destroy a sodas structure.

Objects in the structure are not deallocated.

#### Parameters

<i>sodas</i>	the opaque pointer to the sodas structure.
--------------	--

### 7.24.3.4 int **sodas\_init** ( void )

Initialize the sodas module.

Done by the wsnet core.

#### Returns

0 if success, -1 otherwise.

### 7.24.3.5 void **sodas\_insert** ( void \* *sodas*, void \* *key*, void \* *data* )

Insert an object in the sodas structure.

#### Parameters

<i>sodas</i>	the opaque pointer to the sodas structure.
<i>key</i>	the object key.
<i>data</i>	the object to insert.

### 7.24.3.6 void\* **sodas\_pop** ( void \* *sodas* )

Remove the first object from the sodas structure.

#### Parameters

<i>sodas</i>	the opaque pointer to the data structure.
--------------	---

#### Returns

The first object.

### 7.24.3.7 void\* **sodas\_see\_first** ( void \* *sodas* )

Show the first object in the sodas structure.

The object is not removed.

## Parameters

<code>sodas</code>	the opaque pointer to the data structure.
--------------------	---

## Returns

The first object.

## 7.25 spadas.h File Reference

Space PArTitioning DAta Structure module declarations.

```
#include <include/types.h>
```

## Functions

- int [spadas\\_init](#) (void)  
*Initialize the spadas module.*
- void \* [spadas\\_create](#) ([position\\_t](#) \*area, double range)  
*Create an empty spadas structure.*
- void [spadas\\_destroy](#) (void \*spadas)  
*Destroy a spadas structure.*
- void [spadas\\_insert](#) (void \*spadas, void \*data, [position\\_t](#) \*position)  
*Insert an object in the spadas structure.*
- void [spadas\\_update](#) (void \*spadas, void \*data, [position\\_t](#) \*n\_position, [position\\_t](#) \*o\_position)  
*Update an object position in the spadas structure.*
- void [spadas\\_delete](#) (void \*spadas, void \*data, [position\\_t](#) \*position)  
*Remove an object from the spadas structure.*
- void \* [spadas\\_rangesearch](#) (void \*spadas, void \*data, [position\\_t](#) \*position, double range)  
*Return all objects close to a given position.*

### 7.25.1 Detailed Description

Space PArTitioning DAta Structure module declarations.

## Author

Guillaume Chelius & Elyes Ben Hamida

## Date

2007

Definition in file [spadas.h](#).

## 7.25.2 Function Documentation

### 7.25.2.1 void\* spadas\_create ( position\_t \* area, double range )

Create an empty spadas structure.

#### Parameters

<i>area</i>	the area where objects of the spadas will be.
<i>range</i>	a distance value used for spadas optimization.

#### Returns

An opaque pointer to the spadas structure.

### 7.25.2.2 void spadas\_delete ( void \* spadas, void \* data, position\_t \* position )

Remove an object from the spadas structure.

#### Parameters

<i>spadas</i>	the opaque pointer to the spadas structure.
<i>data</i>	the object to remove.
<i>position</i>	the object position.

### 7.25.2.3 void spadas\_destroy ( void \* spadas )

Destroy a spadas structure.

Objects in the structure are not deallocated.

#### Parameters

<i>spadas</i>	the opaque pointer to the spadas structure.
---------------	---

### 7.25.2.4 int spadas\_init ( void )

Initialize the spadas module.

Done by the wsnet core.

#### Returns

0 if success, -1 otherwise.



7.25.2.5 void **spadas\_insert** ( void \* *spadas*, void \* *data*, position\_t \* *position* )

Insert an object in the spadas structure.

## Parameters

<i>spadas</i>	the opaque pointer to the spadas structure.
<i>data</i>	the object to insert.
<i>position</i>	the object position.

7.25.2.6 void\* **spadas\_rangesearch** ( void \* *spadas*, void \* *data*, position\_t \* *position*, double *range* )

Return all objects close to a given position.

## Parameters

<i>spadas</i>	the opaque pointer to the spadas structure.
<i>data</i>	an object which can be excluded from the search.
<i>position</i>	the central position.
<i>range</i>	distance to the central position.

## Returns

A das containing all objects that are at distance  $\leq$  range from the central position.

7.25.2.7 void **spadas\_update** ( void \* *spadas*, void \* *data*, position\_t \* *n\_position*, position\_t \* *o\_position* )

Update an object position in the spadas structure.

## Parameters

<i>spadas</i>	the opaque pointer to the spadas structure.
<i>data</i>	the object.
<i>n_position</i>	the new object position.
<i>o_position</i>	the old object position.

## 7.26 timer.h File Reference

Generic timer.

## Data Structures

- struct [qtimer\\_s](#)

*timer structure parameters: parameters for next\_trigger functions (eg: period for periodic timer) conditional\_end: pointer to a function that return 1 if the timer must be destroyed and 0 otherwise callback\_function: function to callback when the timer triggers.*

- struct [exponential\\_s](#)
- struct [uniform\\_random\\_s](#)

## Typedefs

- typedef struct [qtimer\\_s](#) [qtimer\\_t](#)

*timer structure parameters: parameters for next\_trigger functions (eg: period for periodic timer) conditional\_end: pointer to a function that return 1 if the timer must be destroyed and 0 otherwise callback\_function: function to callback when the timer triggers.*

- typedef struct [exponential\\_s](#) [exponential\\_parameters\\_t](#)
- typedef struct [uniform\\_random\\_s](#) [uniform\\_random\\_parameters\\_t](#)

## Functions

- uint64\_t [periodic\\_trigger](#) ([call\\_t](#) \*c, void \*timer\_id)  
*qtimer and not timer because timer\_t is already defined in /usr/lib/time.c so I added my first name initial ;)*
- uint64\_t [exponential\\_trigger](#) ([call\\_t](#) \*c, void \*timer\_id)
- uint64\_t [uniform\\_random\\_trigger](#) ([call\\_t](#) \*c, void \*timer\_id)
- int [never\\_stop](#) ([call\\_t](#) \*c, void \*timer\_id)
- void \* [create\\_timer](#) ([call\\_t](#) \*c, void \*callback\_function, void \*conditional\_end, void \*next\_trigger, void \*trigger\_parameters)
- void \* [start\\_timer](#) (void \*timer\_id, uint64\_t delay)
- void [destroy\\_timer](#) (void \*timer\_id)
- void [change\\_parameter](#) (void \*timer\_id, void \*new\_parameters)
- int [timer\\_init](#) (void)
- void [timer\\_clean](#) (void)
- [qtimer\\_t](#) \* [fetch\\_timer](#) (void \*timer\_id)

### 7.26.1 Detailed Description

Generic timer.

Author

Quentin Lampin

Date

2009

Definition in file [timer.h](#).

## 7.26.2 Typedef Documentation

7.26.2.1 `typedef struct exponential_s exponential_parameters_t`

7.26.2.2 `typedef struct qtimer_s qtimer_t`

timer structure parameters: parameters for next\_trigger functions (eg: period for periodic timer) conditional\_end: pointer to a function that return 1 if the timer must be destroyed and 0 otherwise callback\_function: function to callback when the timer triggers.

7.26.2.3 `typedef struct uniform_random_s uniform_random_parameters_t`

## 7.26.3 Function Documentation

7.26.3.1 `void change_parameter ( void * timer_id, void * new_parameters )`

7.26.3.2 `void* create_timer ( call_t * c, void * callback_function, void * conditional_end, void * next_trigger, void * trigger_parameters )`

7.26.3.3 `void destroy_timer ( void * timer_id )`

7.26.3.4 `uint64_t exponential_trigger ( call_t * c, void * timer_id )`

7.26.3.5 `qtimer_t* fetch_timer ( void * timer_id )`

7.26.3.6 `int never_stop ( call_t * c, void * timer_id )`

7.26.3.7 `uint64_t periodic_trigger ( call_t * c, void * timer_id )`

qtimer and not timer because timer\_t is already defined in /usr/lib/time.c so I added my first name initial ;)

7.26.3.8 `void* start_timer ( void * timer_id, uint64_t delay )`

7.26.3.9 `void timer_clean ( void )`

7.26.3.10 `int timer_init ( void )`

7.26.3.11 `uint64_t uniform_random_trigger ( call_t * c, void * timer_id )`

## 7.27 types.h File Reference

Type declarations.

```
#include <stdlib.h> #include <inttypes.h> #include <math.h>
```

## Data Structures

- struct [\\_array](#)  
*An array of integers containing its size.*
- struct [\\_position](#)  
*A position in the 3D space.*
- struct [\\_angle](#)  
*An angle in the 3D space.*
- struct [\\_destination](#)  
*A packet destination.*
- struct [\\_call](#)  
*A parameter that identifies who we are calling and who has called us.*
- struct [\\_packet](#)  
*A radio packet.*
- struct [\\_ioctl\\_message](#)
- struct [\\_param](#)  
*A parameter for the "init" and "setnode" entity functions.*

## Typedefs

- typedef struct [\\_array](#) [array\\_t](#)  
*An array of integers containing its size.*
- typedef int [nodeid\\_t](#)  
*A node identifier.*
- typedef int [entityid\\_t](#)  
*An entity identifier.*
- typedef int [measureid\\_t](#)  
*A measure identifier.*
- typedef int [packetid\\_t](#)  
*A packet identifier.*
- typedef struct [\\_position](#) [position\\_t](#)  
*A position in the 3D space.*
- typedef struct [\\_angle](#) [angle\\_t](#)  
*An angle in the 3D space.*
- typedef struct [\\_destination](#) [destination\\_t](#)  
*A packet destination.*
- typedef struct [\\_call](#) [call\\_t](#)  
*A parameter that identifies who we are calling and who has called us.*
- typedef struct [\\_packet](#) [packet\\_t](#)  
*A radio packet.*
- typedef struct [\\_ioctl\\_message](#) [ioctl\\_message\\_t](#)  
*An ioctl message.*
- typedef struct [\\_param](#) [param\\_t](#)  
*A parameter for the "init" and "setnode" entity functions.*
- typedef int(\* [callback\\_t](#))([call\\_t](#) \*c, void \*arg)  
*Prototype of a callback function.*

### 7.27.1 Detailed Description

Type declarations.

#### Author

Guillaume Chelius & Elyes Ben Hamida & Quentin Lampin

#### Date

2007

Definition in file [types.h](#).

### 7.27.2 Typedef Documentation

#### 7.27.2.1 `angle_t`

An angle in the 3D space.

#### 7.27.2.2 `array_t`

An array of integers containing its size.

#### 7.27.2.3 `call_t`

A parameter that identifies who we are calling and who has called us.

Kind of a self pointer.

#### 7.27.2.4 `callback_t`

Prototype of a callback function.

Definition at line 187 of file types.h.

#### 7.27.2.5 `destination_t`

A packet destination.

May be a node address or a geographical position.

#### 7.27.2.6 `entityid_t`

An entity identifier.

Definition at line 42 of file types.h.

#### 7.27.2.7 `ioctl_message_t`

An ioctl message.

#### 7.27.2.8 `measureid_t`

A measure identifier.

Definition at line 51 of file types.h.

#### 7.27.2.9 `nodeid_t`

A node identifier.

Definition at line 33 of file types.h.

#### 7.27.2.10 `packet_t`

A radio packet.

#### 7.27.2.11 `packetid_t`

A packet identifier.

Definition at line 60 of file types.h.

#### 7.27.2.12 `param_t`

A parameter for the "init" and "setnode" entity functions.

#### 7.27.2.13 `position_t`

A position in the 3D space.

## 7.28 `worldsens_debug.h` File Reference

### Defines

- #define `WSNET_S_DBG` 0
- #define `WSNET_S_EXC_DBG` 0
- #define `WSNET_S_DBG_OUT`(x...) `fprintf(stderr, x)`
- #define `WSNET_S_DBG_DBG`(x...) `do { } while (0)`
- #define `WSNET_S_DBG_EXC`(x...) `do { } while (0)`

### 7.28.1 Define Documentation

#### 7.28.1.1 `#define WSNET_S_DBG 0`

Definition at line 20 of file worldsens\_debug.h.

#### 7.28.1.2 `#define WSNET_S_DBG_DBG( x... ) do { } while (0)`

Definition at line 33 of file worldsens\_debug.h.

#### 7.28.1.3 `#define WSNET_S_DBG_EXC( x... ) do { } while (0)`

Definition at line 39 of file worldsens\_debug.h.

#### 7.28.1.4 `#define WSNET_S_DBG_OUT( x... ) fprintf(stderr, x)`

Definition at line 28 of file worldsens\_debug.h.

#### 7.28.1.5 `#define WSNET_S_EXC_DBG 0`

Definition at line 21 of file worldsens\_debug.h.

## 7.29 worldsens\_pkt.h File Reference

Worldsens packet format.

```
#include <inttypes.h>
```

### Data Structures

- struct [\\_worldsens\\_c\\_header](#)
- struct [\\_worldsens\\_c\\_connect\\_req](#)
- struct [\\_worldsens\\_c\\_sync\\_ack](#)
- struct [\\_worldsens\\_c\\_byte\\_tx](#)
- struct [\\_worldsens\\_c\\_measure\\_req](#)
- struct [\\_worldsens\\_c\\_disconnect](#)
- struct [\\_worldsens\\_s\\_header](#)
- struct [\\_worldsens\\_s\\_connect\\_rsp\\_ok](#)
- struct [\\_worldsens\\_s\\_connect\\_rsp\\_nok](#)
- struct [\\_worldsens\\_s\\_sync\\_release](#)
- struct [\\_worldsens\\_s\\_backtrack](#)
- struct [\\_worldsens\\_s\\_sync\\_reminder](#)
- struct [\\_worldsens\\_s\\_byte\\_rx](#)

- struct [\\_worldsens\\_s\\_byte\\_sr\\_rx](#)
- struct [\\_worldsens\\_s\\_measure\\_rsp](#)
- struct [\\_worldsens\\_s\\_measure\\_sr\\_rsp](#)
- struct [\\_worldsens\\_s\\_killsim](#)
- struct [\\_worldsens\\_s\\_kill](#)
- union [\\_worldsens\\_pkt](#)

## Defines

- #define [\\_\\_PACKED\\_\\_](#) [\\_\\_attribute\\_\\_\(\(packed\)\)](#)
- #define [WORLDSSENS\\_MAX\\_PKTLENGTH](#) (sizeof(union [\\_worldsens\\_pkt](#)))
- #define [WORLDSSENS\\_MAX\\_MODELS\\_SIZE](#) 1200

## Typedefs

- typedef uint8\_t [ws\\_pkt\\_type](#)
- typedef uint32\_t [ws\\_id\\_node](#)
- typedef int64\_t [ws\\_id\\_resource](#)
- typedef uint64\_t [ws\\_id\\_rp](#)
- typedef uint64\_t [ws\\_id\\_seq](#)
- typedef uint64\_t [ws\\_frequency](#)
- typedef uint64\_t [ws\\_power](#)
- typedef uint64\_t [ws\\_measure](#)
- typedef uint64\_t [ws\\_sinr](#)
- typedef uint64\_t [ws\\_time](#)
- typedef uint8\_t [ws\\_data](#)

## Enumerations

- enum [worldsens\\_pkt\\_type](#) { [WORLDSSENS\\_UNKNOWN](#) = 0, [WORLDSSENS\\_C\\_CONNECT\\_REQ](#), [WORLDSSENS\\_C\\_SYNC\\_ACK](#), [WORLDSSENS\\_C\\_BYTE\\_TX](#), [WORLDSSENS\\_C\\_MEASURE\\_REQ](#), [WORLDSSENS\\_C\\_DISCONNECT](#), [WORLDSSENS\\_S\\_CONNECT\\_RSP\\_OK](#), [WORLDSSENS\\_S\\_CONNECT\\_RSP\\_NOK](#), [WORLDSSENS\\_S\\_SYNC\\_RELEASE](#), [WORLDSSENS\\_S\\_SYNC\\_REMINDER](#), [WORLDSSENS\\_S\\_BACKTRACK](#), [WORLDSSENS\\_S\\_BYTE\\_RX](#), [WORLDSSENS\\_S\\_BYTE\\_SR\\_RX](#), [WORLDSSENS\\_S\\_MEASURE\\_RSP](#), [WORLDSSENS\\_S\\_MEASURE\\_SR\\_RSP](#), [WORLDSSENS\\_S\\_KILLSIM](#), [WORLDSSENS\\_S\\_KILL](#), [WORLDSSENS\\_LASTID](#) }

## Functions

- int [worldsens\\_packet\\_hton](#) (union [\\_worldsens\\_pkt](#) \*pkt)
- int [worldsens\\_packet\\_ntoh](#) (union [\\_worldsens\\_pkt](#) \*pkt)
- int [worldsens\\_packet\\_dump](#) (union [\\_worldsens\\_pkt](#) \*pkt)



### 7.29.1 Detailed Description

Worldsens packet format.

**Author**

Guillaume Chelius, Antoine Fraboulet, Loic Lemaitre

**Date**

2007

Definition in file [worldsens\\_pkt.h](#).

### 7.29.2 Define Documentation

#### 7.29.2.1 #define \_\_PACKED\_\_ \_\_attribute\_\_((packed))

Definition at line 14 of file worldsens\_pkt.h.

#### 7.29.2.2 #define WORLDSSENS\_MAX\_MODELS\_SIZE 1200

Definition at line 124 of file worldsens\_pkt.h.

#### 7.29.2.3 #define WORLDSSENS\_MAX\_PKTLENGTH (sizeof(union \_worldsens\_pkt))

Definition at line 16 of file worldsens\_pkt.h.

### 7.29.3 Typedef Documentation

#### 7.29.3.1 typedef uint8\_t ws\_data

Definition at line 69 of file worldsens\_pkt.h.

#### 7.29.3.2 typedef uint64\_t ws\_frequency

Definition at line 63 of file worldsens\_pkt.h.

#### 7.29.3.3 typedef uint32\_t ws\_id\_node

Definition at line 56 of file worldsens\_pkt.h.

#### 7.29.3.4 typedef int64\_t ws\_id\_resource

Definition at line 57 of file worldsens\_pkt.h.

#### 7.29.3.5 typedef uint64\_t ws\_id\_rp

Definition at line 58 of file worldsens\_pkt.h.

#### 7.29.3.6 typedef uint64\_t ws\_id\_seq

Definition at line 59 of file worldsens\_pkt.h.

#### 7.29.3.7 typedef uint64\_t ws\_measure

Definition at line 65 of file worldsens\_pkt.h.

#### 7.29.3.8 typedef uint8\_t ws\_pkt\_type

Definition at line 54 of file worldsens\_pkt.h.

#### 7.29.3.9 typedef uint64\_t ws\_power

Definition at line 64 of file worldsens\_pkt.h.

#### 7.29.3.10 typedef uint64\_t ws\_sinr

Definition at line 66 of file worldsens\_pkt.h.

#### 7.29.3.11 typedef uint64\_t ws\_time

Definition at line 68 of file worldsens\_pkt.h.

### 7.29.4 Enumeration Type Documentation

#### 7.29.4.1 enum woldsens\_pkt\_type

Enumerator:

**WORLDSENS\_UNKNOWN**  
**WORLDSENS\_C\_CONNECT\_REQ**  
**WORLDSENS\_C\_SYNC\_ACK**  
**WORLDSENS\_C\_BYTE\_TX**  
**WORLDSENS\_C\_MEASURE\_REQ**  
**WORLDSENS\_C\_DISCONNECT**  
**WORLDSENS\_S\_CONNECT\_RSP\_OK**  
**WORLDSENS\_S\_CONNECT\_RSP\_NOK**

**WORLDSENS\_S\_SYNC\_RELEASE**  
**WORLDSENS\_S\_SYNC\_REMINDER**  
**WORLDSENS\_S\_BACKTRACK**  
**WORLDSENS\_S\_BYTE\_RX**  
**WORLDSENS\_S\_BYTE\_SR\_RX**  
**WORLDSENS\_S\_MEASURE\_RSP**  
**WORLDSENS\_S\_MEASURE\_SR\_RSP**  
**WORLDSENS\_S\_KILLSIM**  
**WORLDSENS\_S\_KILL**  
**WORLDSENS\_LASTID**

Definition at line 23 of file worldsens\_pkt.h.

#### 7.29.5 Function Documentation

7.29.5.1 int worldsens\_packet\_dump ( union \_worldsens\_pkt \* pkt )

7.29.5.2 int worldsens\_packet\_hton ( union \_worldsens\_pkt \* pkt )

7.29.5.3 int worldsens\_packet\_ntoh ( union \_worldsens\_pkt \* pkt )