

Mobile Network Simulator

Generated by Doxygen 1.8.15

1 Namespace Index	1
1.1 Namespace List	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Namespace Documentation	9
5.1 utils Namespace Reference	9
5.1.1 Detailed Description	9
5.1.2 Function Documentation	9
5.1.2.1 d2r()	9
5.1.2.2 generateFixedPoints()	10
5.1.2.3 generatePoints()	10
5.1.2.4 getFirstChildElement()	11
5.1.2.5 getNode()	11
5.1.2.6 getValue() [1/5]	11
5.1.2.7 getValue() [2/5]	12
5.1.2.8 getValue() [3/5]	12
5.1.2.9 getValue() [4/5]	13
5.1.2.10 getValue() [5/5]	13
5.1.2.11 printAntennaHeader()	14
5.1.2.12 printMobileOperatorHeader()	14
5.1.2.13 printPersonHeader()	14
5.1.2.14 printPhoneHeader()	14
5.1.2.15 r2d()	14
5.1.3 Variable Documentation	15
5.1.3.1 PI	15
6 Class Documentation	17
6.1 Agent Class Reference	17
6.1.1 Detailed Description	18
6.1.2 Constructor & Destructor Documentation	18
6.1.2.1 Agent()	19
6.1.2.2 ~Agent()	19
6.1.3 Member Function Documentation	19
6.1.3.1 getClock()	19
6.1.3.2 getId()	19
6.1.3.3 getMap()	20

6.1.3.4 getName()	20
6.1.3.5 operator==()	20
6.1.3.6 setMap()	20
6.1.3.7 toString()	21
6.1.4 Member Data Documentation	21
6.1.4.1 m_clock	21
6.1.4.2 m_id	21
6.1.4.3 m_map	21
6.2 AgentsCollection Class Reference	22
6.2.1 Detailed Description	22
6.2.2 Constructor & Destructor Documentation	22
6.2.2.1 AgentsCollection()	22
6.2.2.2 ~AgentsCollection()	22
6.2.3 Member Function Documentation	23
6.2.3.1 addAgent()	23
6.2.3.2 begin()	24
6.2.3.3 deleteAgent()	24
6.2.3.4 end()	24
6.2.3.5 getAgent()	25
6.2.3.6 getAgentListByType()	25
6.2.3.7 printAgents()	25
6.2.3.8 size()	25
6.2.4 Member Data Documentation	26
6.2.4.1 m_agents	26
6.3 Antenna Class Reference	26
6.3.1 Detailed Description	29
6.3.2 Constructor & Destructor Documentation	29
6.3.2.1 Antenna() [1/2]	29
6.3.2.2 Antenna() [2/2]	30
6.3.2.3 ~Antenna()	30
6.3.3 Member Function Documentation	30
6.3.3.1 alreadyRegistered()	31
6.3.3.2 attachDevice()	31
6.3.3.3 computePower() [1/2]	31
6.3.3.4 computePower() [2/2]	31
6.3.3.5 computeSignalMeasure()	32
6.3.3.6 computeSignalQuality() [1/2]	32
6.3.3.7 computeSignalQuality() [2/2]	32
6.3.3.8 computeSignalQualityDirectional()	33
6.3.3.9 computeSignalQualityOmnidirectional()	33
6.3.3.10 computeSignalStrength() [1/2]	33
6.3.3.11 computeSignalStrength() [2/2]	33

6.3.3.12 computeSignalStrengthDirectional()	34
6.3.3.13 computeSignalStrengthOmnidirectional()	34
6.3.3.14 createMapping()	34
6.3.3.15 detachDevice()	34
6.3.3.16 dumpCell()	35
6.3.3.17 findSD()	35
6.3.3.18 getAntennaOutputFileName()	35
6.3.3.19 getCoverageArea()	35
6.3.3.20 getCoverageAreaDirectional()	36
6.3.3.21 getCoverageAreaOmnidirectional()	36
6.3.3.22 getHandoverMechanism()	36
6.3.3.23 getMaxConnections()	36
6.3.3.24 getMin3db()	36
6.3.3.25 getMNO()	36
6.3.3.26 getName()	37
6.3.3.27 getNumActiveConections()	37
6.3.3.28 getPLE()	37
6.3.3.29 getPower()	37
6.3.3.30 getRmax()	38
6.3.3.31 getSmid()	38
6.3.3.32 getSmin()	38
6.3.3.33 getSSteep()	38
6.3.3.34 getType()	39
6.3.3.35 norm_dBLoss()	39
6.3.3.36 normalizeAngle()	39
6.3.3.37 projectToEPlane()	39
6.3.3.38 registerEvent()	39
6.3.3.39 S()	39
6.3.3.40 S0()	40
6.3.3.41 SDist()	40
6.3.3.42 searchMin()	40
6.3.3.43 setHandoverMechanism()	40
6.3.3.44 setLocationWithElevation()	40
6.3.3.45 setMaxConnections()	40
6.3.3.46 setPLE()	41
6.3.3.47 setPower()	41
6.3.3.48 setSmid()	41
6.3.3.49 setSSteep()	42
6.3.3.50 setType()	42
6.3.3.51 toString()	42
6.3.3.52 tryRegisterDevice()	42
6.3.4 Member Data Documentation	43

6.3.4.1 m_azim_dB_Back	43
6.3.4.2 m_beam_H	43
6.3.4.3 m_beam_V	43
6.3.4.4 m_cell	43
6.3.4.5 m_devices	43
6.3.4.6 m_direction	44
6.3.4.7 m_elev_dB_Back	44
6.3.4.8 m_file	44
6.3.4.9 m_handoverMechanism	44
6.3.4.10 m_height	44
6.3.4.11 m_mapping_azim	44
6.3.4.12 m_mapping_elev	44
6.3.4.13 m_maxConnections	44
6.3.4.14 m_MNO	45
6.3.4.15 m_ple	45
6.3.4.16 m_power	45
6.3.4.17 m_Qmin	45
6.3.4.18 m_rmax	45
6.3.4.19 m_S0	45
6.3.4.20 m_sd_azim	45
6.3.4.21 m_sd_elev	45
6.3.4.22 m_Smid	46
6.3.4.23 m_Smin	46
6.3.4.24 m_SSteep	46
6.3.4.25 m_tilt	46
6.3.4.26 m_type	46
6.4 AntennaInfo Class Reference	46
6.4.1 Detailed Description	47
6.4.2 Constructor & Destructor Documentation	47
6.4.2.1 AntennaInfo()	47
6.4.3 Member Function Documentation	47
6.4.3.1 getAntennaId()	47
6.4.3.2 getDeviceId()	48
6.4.3.3 getEventCode()	48
6.4.3.4 getTime()	48
6.4.3.5 getX()	48
6.4.3.6 getY()	49
6.4.3.7 operator<()	49
6.4.3.8 toString()	49
6.4.4 Member Data Documentation	49
6.4.4.1 m_antennaId	49
6.4.4.2 m_deviceId	50

6.4.4.3 m_eventCode	50
6.4.4.4 m_time	50
6.4.4.5 m_x	50
6.4.4.6 m_y	50
6.5 Clock Class Reference	50
6.5.1 Detailed Description	51
6.5.2 Constructor & Destructor Documentation	51
6.5.2.1 Clock() [1/2]	51
6.5.2.2 Clock() [2/2]	51
6.5.2.3 ~Clock()	51
6.5.3 Member Function Documentation	52
6.5.3.1 getCurrentTime()	52
6.5.3.2 getFinalTime()	52
6.5.3.3 getIncrement()	52
6.5.3.4 getInitialTime()	52
6.5.3.5 realTime()	53
6.5.3.6 reset()	53
6.5.3.7 setCurrentTime()	53
6.5.3.8 setFinalTime()	53
6.5.3.9 setIncrement()	53
6.5.3.10 setInitialTime()	54
6.5.3.11 tick()	54
6.5.4 Member Data Documentation	54
6.5.4.1 m_currentTime	54
6.5.4.2 m_finalTime	54
6.5.4.3 m_increment	55
6.5.4.4 m_initialTime	55
6.6 Constants Class Reference	55
6.6.1 Detailed Description	56
6.6.2 Member Data Documentation	56
6.6.2.1 ANTENNA_AZIM_DB_BACK	56
6.6.2.2 ANTENNA_BEAM_H	56
6.6.2.3 ANTENNA_BEAM_V	56
6.6.2.4 ANTENNA_DIRECTION	57
6.6.2.5 ANTENNA_ELEV_DB_BACK	57
6.6.2.6 ANTENNA_HEIGHT	57
6.6.2.7 ANTENNA_MAPPING_N	57
6.6.2.8 ANTENNA_MAX_CONNECTIONS	57
6.6.2.9 ANTENNA_MIN_3_DB	57
6.6.2.10 ANTENNA_POWER	57
6.6.2.11 ANTENNA_QMIN	57
6.6.2.12 ANTENNA_S_MID	58

6.6.2.13 ANTENNA_S_STEEP	58
6.6.2.14 ANTENNA_SMIN	58
6.6.2.15 ANTENNA_TILT	58
6.6.2.16 ANTENNAS_FILE_NAME	58
6.6.2.17 ATT_FACTOR	58
6.6.2.18 GRID_DIM_TILE_X	58
6.6.2.19 GRID_DIM_TILE_Y	59
6.6.2.20 GRID_FILE_NAME	59
6.6.2.21 GRID_X_ORIG	59
6.6.2.22 GRID_Y_ORIG	59
6.6.2.23 PERSONS_FILE_NAME	59
6.6.2.24 PHONE_CONNECTION_THRESHOLD	59
6.6.2.25 PHONE_POWER_THRESHOLD	60
6.6.2.26 PHONE_QUALITY_THRESHOLD	60
6.6.2.27 PHONE_STRENGTH_THRESHOLD	60
6.6.2.28 PRIOR_PROBABILITY	60
6.6.2.29 PROB_FILE_NAME_PREFIX	60
6.6.2.30 RANDOM_SEED	60
6.6.2.31 sep	60
6.6.2.32 SIM_DEFAULT_MNO_NAME	61
6.6.2.33 SIM_END_TIME	61
6.6.2.34 SIM_INCREMENT_TIME	61
6.6.2.35 SIM_INTERVAL_BETWEEN_STAYS	61
6.6.2.36 SIM_NO_ANTENNAS	61
6.6.2.37 SIM_NO_MOBILE_PHONES	61
6.6.2.38 SIM_NO_PERSONS	61
6.6.2.39 SIM_NUM_MNO	61
6.6.2.40 SIM_PROB_MOBILE_PHONE	62
6.6.2.41 SIM_PROB_SECOND_MOBILE_PHONE	62
6.6.2.42 SIM_START_TIME	62
6.6.2.43 SIM_STAY_TIME	62
6.6.2.44 SIM_TREND_ANGLE_1	62
6.6.2.45 SIM_TREND_ANGLE_2	62
6.7 CSVParser Class Reference	62
6.7.1 Detailed Description	63
6.7.2 Constructor & Destructor Documentation	63
6.7.2.1 CSVParser()	63
6.7.2.2 ~CSVParser()	64
6.7.3 Member Function Documentation	64
6.7.3.1 addRow()	64
6.7.3.2 columnCount()	64
6.7.3.3 deleteRow()	65

6.7.3.4	getFileName()	65
6.7.3.5	getHeader()	65
6.7.3.6	getHeaderElement()	65
6.7.3.7	getRow()	66
6.7.3.8	operator[]()	66
6.7.3.9	parseContent()	66
6.7.3.10	parseHeader()	67
6.7.3.11	rowCount()	67
6.7.3.12	sync()	67
6.7.4	Member Data Documentation	67
6.7.4.1	_content	67
6.7.4.2	_file	67
6.7.4.3	_header	67
6.7.4.4	_originalFile	68
6.7.4.5	_sep	68
6.7.4.6	_type	68
6.7.4.7	m_header	68
6.8	EMField Class Reference	68
6.8.1	Detailed Description	69
6.8.2	Constructor & Destructor Documentation	69
6.8.2.1	~EMField()	70
6.8.2.2	EMField() [1/2]	70
6.8.2.3	EMField() [2/2]	70
6.8.3	Member Function Documentation	70
6.8.3.1	addAntenna()	70
6.8.3.2	computeMaxPower()	70
6.8.3.3	computeMaxQuality()	71
6.8.3.4	computeMaxStrength()	71
6.8.3.5	connectionLikelihood()	72
6.8.3.6	connectionLikelihoodGrid()	72
6.8.3.7	getAntennaMin3DbArray()	73
6.8.3.8	getInRangeAntennas()	73
6.8.3.9	getSd()	73
6.8.3.10	instance()	73
6.8.3.11	isAntennaInRange()	74
6.8.3.12	operator=()	74
6.8.3.13	sumSignalQuality()	74
6.8.4	Member Data Documentation	75
6.8.4.1	m_antennaMin3DbArray	75
6.8.4.2	m_antennas	75
6.8.4.3	m_instance	75
6.8.4.4	m_sd	75

6.8.4.5 m_sumQuality	75
6.9 Grid Class Reference	75
6.9.1 Detailed Description	76
6.9.2 Constructor & Destructor Documentation	77
6.9.2.1 Grid()	77
6.9.2.2 ~Grid()	77
6.9.3 Member Function Documentation	77
6.9.3.1 computeProbability()	77
6.9.3.2 computeTileCenters()	78
6.9.3.3 dumpGrid()	78
6.9.3.4 getNoTiles()	78
6.9.3.5 getNoTilesX()	79
6.9.3.6 getNoTilesY()	79
6.9.3.7 getTileCenter()	79
6.9.3.8 getTileCenters()	79
6.9.3.9 getTileIndexX() [1/2]	80
6.9.3.10 getTileIndexX() [2/2]	80
6.9.3.11 getTileIndexY() [1/2]	80
6.9.3.12 getTileIndexY() [2/2]	80
6.9.3.13 getTileNo() [1/2]	81
6.9.3.14 getTileNo() [2/2]	81
6.9.3.15 getXOrigin()	81
6.9.3.16 getXTileDim()	82
6.9.3.17 getYOrigin()	82
6.9.3.18 getYTileDim()	82
6.9.3.19 toString()	82
6.9.3.20 useNetworkPrior()	82
6.9.3.21 useUniformPrior()	83
6.9.4 Member Data Documentation	83
6.9.4.1 m_noTilesX	83
6.9.4.2 m_noTilesY	83
6.9.4.3 m_tileCenters	83
6.9.4.4 m_xOrigin	83
6.9.4.5 m_xTileDim	83
6.9.4.6 m_yOrigin	83
6.9.4.7 m_yTileDim	84
6.10 HoldableAgent Class Reference	84
6.10.1 Detailed Description	86
6.10.2 Member Enumeration Documentation	86
6.10.2.1 CONNECTION_TYPE	86
6.10.3 Constructor & Destructor Documentation	86
6.10.3.1 HoldableAgent() [1/2]	86

6.10.3.2 <code>HoldableAgent()</code> [2/2]	87
6.10.3.3 <code>~HoldableAgent()</code>	87
6.10.4 Member Function Documentation	87
6.10.4.1 <code>getAntennas()</code>	87
6.10.4.2 <code>getHolder()</code>	88
6.10.4.3 <code>getName()</code>	88
6.10.4.4 <code>isConnected()</code>	88
6.10.4.5 <code>setHolder()</code>	88
6.10.4.6 <code>setLocation()</code>	89
6.10.4.7 <code>toString()</code>	89
6.10.4.8 <code>tryConnect()</code>	89
6.10.5 Member Data Documentation	90
6.10.5.1 <code>m_antennas</code>	90
6.10.5.2 <code>m_holder</code>	90
6.11 IDGenerator Class Reference	90
6.11.1 Detailed Description	91
6.11.2 Constructor & Destructor Documentation	91
6.11.2.1 <code>IDGenerator()</code>	91
6.11.3 Member Function Documentation	91
6.11.3.1 <code>instance()</code>	91
6.11.3.2 <code>next()</code>	92
6.11.4 Member Data Documentation	92
6.11.4.1 <code>m_id</code>	92
6.11.4.2 <code>m_instance</code>	92
6.12 ImmovableAgent Class Reference	92
6.12.1 Detailed Description	93
6.12.2 Constructor & Destructor Documentation	93
6.12.2.1 <code>ImmovableAgent()</code>	94
6.12.2.2 <code>~ImmovableAgent()</code>	95
6.12.3 Member Function Documentation	95
6.12.3.1 <code>getName()</code>	95
6.12.3.2 <code>toString()</code>	95
6.13 InputParser Class Reference	96
6.13.1 Detailed Description	96
6.13.2 Constructor & Destructor Documentation	96
6.13.2.1 <code>InputParser()</code>	96
6.13.3 Member Function Documentation	96
6.13.3.1 <code>cmdOptionExists()</code>	96
6.13.3.2 <code>getCmdOption()</code>	97
6.13.4 Member Data Documentation	97
6.13.4.1 <code>tokens</code>	97
6.14 LocatableAgent Class Reference	97

6.14.1 Detailed Description	99
6.14.2 Constructor & Destructor Documentation	99
6.14.2.1 LocatableAgent()	99
6.14.2.2 ~LocatableAgent()	99
6.14.3 Member Function Documentation	100
6.14.3.1 dumpLocation()	100
6.14.3.2 getLocation()	100
6.14.3.3 getName()	100
6.14.3.4 setLocation()	100
6.14.3.5 toString()	101
6.14.4 Member Data Documentation	101
6.14.4.1 m_location	101
6.15 Map Class Reference	101
6.15.1 Detailed Description	102
6.15.2 Constructor & Destructor Documentation	102
6.15.2.1 Map() [1/3]	102
6.15.2.2 Map() [2/3]	102
6.15.2.3 Map() [3/3]	103
6.15.2.4 ~Map()	103
6.15.3 Member Function Documentation	103
6.15.3.1 addGrid()	103
6.15.3.2 create_rectangle()	104
6.15.3.3 getBoundary()	104
6.15.3.4 getGlobalFactory()	104
6.15.3.5 getGrid()	104
6.15.3.6 setBoundary()	104
6.15.4 Member Data Documentation	105
6.15.4.1 m_boundary	105
6.15.4.2 m_globalFactory	105
6.15.4.3 m_grid	105
6.16 MobileOperator Class Reference	105
6.16.1 Detailed Description	106
6.16.2 Constructor & Destructor Documentation	107
6.16.2.1 MobileOperator()	107
6.16.2.2 ~MobileOperator()	107
6.16.3 Member Function Documentation	107
6.16.3.1 getAntennaCellsFile()	107
6.16.3.2 getMNOName()	108
6.16.3.3 getName()	108
6.16.3.4 getProbMobilePhone()	108
6.16.3.5 getSignalFile()	108
6.16.3.6 toString()	109

6.16.4 Member Data Documentation	109
6.16.4.1 m_antennaCellsFileName	109
6.16.4.2 m_name	109
6.16.4.3 m_probMobilePhone	109
6.16.4.4 m_signalMeasureFileName	109
6.17 MobilePhone Class Reference	110
6.17.1 Detailed Description	112
6.17.2 Constructor & Destructor Documentation	112
6.17.2.1 MobilePhone()	112
6.17.2.2 ~MobilePhone()	113
6.17.3 Member Function Documentation	113
6.17.3.1 getConnectionThreshold()	113
6.17.3.2 getMobileOperator()	113
6.17.3.3 getName()	113
6.17.3.4 move()	113
6.17.3.5 setMobileOperator()	114
6.17.3.6 toString()	114
6.17.3.7 tryConnect()	115
6.17.4 Member Data Documentation	115
6.17.4.1 m_connectedTo	115
6.17.4.2 m_connType	115
6.17.4.3 m_mno	115
6.17.4.4 m_threshold	115
6.18 MovableAgent Class Reference	116
6.18.1 Detailed Description	117
6.18.2 Constructor & Destructor Documentation	118
6.18.2.1 MovableAgent()	118
6.18.2.2 ~MovableAgent()	118
6.18.3 Member Function Documentation	118
6.18.3.1 getName()	118
6.18.3.2 getSpeed()	119
6.18.3.3 move()	119
6.18.3.4 setSpeed()	119
6.18.3.5 toString()	120
6.18.4 Member Data Documentation	120
6.18.4.1 m_speed	120
6.19 Person Class Reference	120
6.19.1 Detailed Description	123
6.19.2 Member Enumeration Documentation	123
6.19.2.1 AgeDistributions	123
6.19.2.2 Gender	123
6.19.3 Constructor & Destructor Documentation	124

6.19.3.1 Person()	124
6.19.3.2 ~Person()	124
6.19.4 Member Function Documentation	124
6.19.4.1 addDevice()	125
6.19.4.2 dumpDevices()	125
6.19.4.3 generateNewLocation()	125
6.19.4.4 getAge()	125
6.19.4.5 getAvgIntervalBetweenStays()	126
6.19.4.6 getAvgTimeStay()	126
6.19.4.7 getGender()	126
6.19.4.8 getName()	126
6.19.4.9 hasDevices()	127
6.19.4.10 move()	127
6.19.4.11 randomWalkClosedMap()	127
6.19.4.12 randomWalkClosedMapDrift()	128
6.19.4.13 setAge()	128
6.19.4.14 setLocation()	128
6.19.4.15 setNewLocation()	128
6.19.4.16 toString()	128
6.19.5 Member Data Documentation	129
6.19.5.1 m_age	129
6.19.5.2 m_avgIntervalBetweenStays	129
6.19.5.3 m_avgTimeStay	129
6.19.5.4 m_changeDirection	129
6.19.5.5 m_gender	129
6.19.5.6 m_idDevices	129
6.19.5.7 m_nextStay	129
6.19.5.8 m_timeStay	130
6.20 RandomNumberGenerator Class Reference	130
6.20.1 Detailed Description	131
6.20.2 Constructor & Destructor Documentation	131
6.20.2.1 RandomNumberGenerator() [1/3]	131
6.20.2.2 RandomNumberGenerator() [2/3]	131
6.20.2.3 RandomNumberGenerator() [3/3]	131
6.20.3 Member Function Documentation	132
6.20.3.1 generateBernoulliInt() [1/2]	132
6.20.3.2 generateBernoulliInt() [2/2]	133
6.20.3.3 generateBinomialInt() [1/2]	133
6.20.3.4 generateBinomialInt() [2/2]	134
6.20.3.5 generateExponentialDouble()	134
6.20.3.6 generateNormal2Double()	134
6.20.3.7 generateNormalDouble() [1/2]	135

6.20.3.8 generateNormalDouble() [2/2]	135
6.20.3.9 generateTruncatedNormalDouble()	136
6.20.3.10 generateUniformDouble() [1/2]	136
6.20.3.11 generateUniformDouble() [2/2]	137
6.20.3.12 generateUniformInt() [1/2]	137
6.20.3.13 generateUniformInt() [2/2]	137
6.20.3.14 instance() [1/2]	138
6.20.3.15 instance() [2/2]	138
6.20.3.16 normal_pdf()	138
6.20.3.17 operator=()	139
6.20.3.18 setSeed()	139
6.20.4 Member Data Documentation	139
6.20.4.1 m_bernoulli_distribution	139
6.20.4.2 m_binomial_distribution	139
6.20.4.3 m_exponential_double_distribution	140
6.20.4.4 m_generator	140
6.20.4.5 m_instance	140
6.20.4.6 m_normal_double_distribution	140
6.20.4.7 m_unif_double_distribution	140
6.20.4.8 m_unif_int_distribution	140
6.21 Row Class Reference	140
6.21.1 Detailed Description	141
6.21.2 Constructor & Destructor Documentation	141
6.21.2.1 Row()	141
6.21.2.2 ~Row()	141
6.21.3 Member Function Documentation	142
6.21.3.1 getValue()	142
6.21.3.2 operator[]() [1/2]	142
6.21.3.3 operator[]() [2/2]	142
6.21.3.4 push()	143
6.21.3.5 set()	143
6.21.3.6 size()	143
6.21.4 Friends And Related Function Documentation	144
6.21.4.1 operator<< [1/2]	144
6.21.4.2 operator<< [2/2]	144
6.21.5 Member Data Documentation	144
6.21.5.1 _header	145
6.21.5.2 _values	145
6.22 SimException Class Reference	145
6.22.1 Constructor & Destructor Documentation	146
6.22.1.1 SimException()	146
6.22.1.2 ~SimException()	146

6.22.2 Member Function Documentation	146
6.22.2.1 what()	146
6.22.3 Member Data Documentation	146
6.22.3.1 msg	146
6.23 Tablet Class Reference	147
6.23.1 Constructor & Destructor Documentation	148
6.23.1.1 Tablet()	149
6.23.1.2 ~Tablet()	149
6.23.2 Member Function Documentation	149
6.23.2.1 getName()	149
6.23.2.2 move()	149
6.23.2.3 toString()	150
6.23.2.4 tryConnect()	150
6.24 World Class Reference	151
6.24.1 Detailed Description	152
6.24.2 Constructor & Destructor Documentation	153
6.24.2.1 World() [1/2]	153
6.24.2.2 World() [2/2]	153
6.24.2.3 ~World()	154
6.24.3 Member Function Documentation	154
6.24.3.1 generateAntennas()	154
6.24.3.2 generateMobilePhones()	154
6.24.3.3 generatePopulation() [1/2]	154
6.24.3.4 generatePopulation() [2/2]	154
6.24.3.5 getAgents()	155
6.24.3.6 getAntennasFilename()	155
6.24.3.7 getClock()	155
6.24.3.8 getConnectionType()	155
6.24.3.9 getDefaultConnectionThreshold()	156
6.24.3.10 getGridDimTileX()	156
6.24.3.11 getGridDimTileY()	156
6.24.3.12 getGridFilename()	156
6.24.3.13 getMap()	157
6.24.3.14 getPersonsFilename()	157
6.24.3.15 getPrior()	157
6.24.3.16 getProbFilenames()	157
6.24.3.17 parseAntennas()	158
6.24.3.18 parsePersons()	158
6.24.3.19 parseProbabilities()	158
6.24.3.20 parseSimulationFile()	158
6.24.3.21 runSimulation()	158
6.24.3.22 setAgents()	158

6.24.3.23 setClock()	159
6.24.3.24 setMap()	159
6.24.3.25 whichMNO()	159
6.24.4 Member Data Documentation	159
6.24.4.1 m_agentsCollection	159
6.24.4.2 m_antennasFilename	160
6.24.4.3 m_clock	160
6.24.4.4 m_connThreshold	160
6.24.4.5 m_connType	160
6.24.4.6 m_endTime	160
6.24.4.7 m_GridDimTileX	160
6.24.4.8 m_GridDimTileY	160
6.24.4.9 m_gridFilename	160
6.24.4.10 m_intervalBetweenStays	161
6.24.4.11 m_map	161
6.24.4.12 m_mvType	161
6.24.4.13 m_personsFilename	161
6.24.4.14 m_prior	161
6.24.4.15 m_probFilenames	161
6.24.4.16 m_probSecMobilePhone	161
6.24.4.17 m_seed	161
6.24.4.18 m_startTime	162
6.24.4.19 m_stay	162
6.24.4.20 m_timeIncrement	162
7 File Documentation	163
7.1 include/Agent.h File Reference	163
7.2 include/AgentsCollection.h File Reference	164
7.2.1 Typedef Documentation	165
7.2.1.1 um_iterator	165
7.3 include/Antenna.h File Reference	165
7.4 include/AntennaInfo.h File Reference	166
7.5 include/AntennaType.h File Reference	168
7.5.1 Enumeration Type Documentation	168
7.5.1.1 AntennaType	168
7.6 include/Clock.h File Reference	169
7.7 include/Constants.h File Reference	170
7.8 include/CSVparser.hpp File Reference	170
7.8.1 Enumeration Type Documentation	171
7.8.1.1 DataType	171
7.9 include/EMField.h File Reference	171
7.10 include/EventType.h File Reference	172

7.10.1 Enumeration Type Documentation	173
7.10.1.1 EventType	173
7.11 include/Grid.h File Reference	173
7.12 include/HoldableAgent.h File Reference	174
7.13 include/IDGenerator.h File Reference	176
7.14 include/ImmovableAgent.h File Reference	176
7.15 include/InputParser.h File Reference	177
7.16 include/LocatableAgent.h File Reference	178
7.17 include/Map.h File Reference	179
7.18 include/MobileOperator.h File Reference	180
7.19 include/MobilePhone.h File Reference	181
7.20 include/MovableAgent.h File Reference	182
7.21 include/MovementType.h File Reference	184
7.21.1 Enumeration Type Documentation	184
7.21.1.1 MovementType	184
7.22 include/Person.h File Reference	185
7.23 include/PriorType.h File Reference	186
7.23.1 Enumeration Type Documentation	186
7.23.1.1 PriorType	186
7.24 include/RandomNumberGenerator.h File Reference	186
7.25 include/SimException.h File Reference	187
7.26 include/Tablet.h File Reference	188
7.27 include/Utils.h File Reference	188
7.28 include/World.h File Reference	189
Index	191

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

utils	9
---------------------------------	---

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Agent	17
LocatableAgent	97
ImmovableAgent	92
Antenna	26
MovableAgent	116
HoldableAgent	84
MobilePhone	110
Tablet	147
Person	120
MobileOperator	105
AgentsCollection	22
AntennaInfo	46
Clock	50
Constants	55
CSVParser	62
EMField	68
exception	
SimException	145
Grid	75
IDGenerator	90
InputParser	96
Map	101
RandomNumberGenerator	130
Row	140
World	151

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Agent	17
AgentsCollection	22
Antenna	26
AntennaInfo	46
Clock	50
Constants	55
CSVParser	62
EMField	68
Grid	75
HoldableAgent	84
IDGenerator	90
ImmovableAgent	92
InputParser	96
LocatableAgent	97
Map	101
MobileOperator	105
MobilePhone	110
MovableAgent	116
Person	120
RandomNumberGenerator	130
Row	140
SimException	145
Tablet	147
World	151

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

include/Agent.h	163
include/AgentsCollection.h	164
include/Antenna.h	165
include/AntennaInfo.h	166
include/AntennaType.h	168
include/Clock.h	169
include/Constants.h	170
include/CSVparser.hpp	170
include/EMField.h	171
include/EventType.h	172
include/Grid.h	173
include/HoldableAgent.h	174
include/IDGenerator.h	176
include/ImmovableAgent.h	176
include/InputParser.h	177
include/LocatableAgent.h	178
include/Map.h	179
include/MobileOperator.h	180
include/MobilePhone.h	181
include/MovableAgent.h	182
include/MovementType.h	184
include/Person.h	185
include/PriorType.h	186
include/RandomNumberGenerator.h	186
include/SimException.h	187
include/Tablet.h	188
include/Utils.h	188
include/World.h	189

Chapter 5

Namespace Documentation

5.1 utils Namespace Reference

Functions

- `vector< Point * > generatePoints` (const `Map` *m, unsigned long n, double percentHome, unsigned seed)
- `vector< Point * > generateFixedPoints` (const `Map` *m, unsigned long n, unsigned seed)
- `void printPersonHeader` ()
- `void printAntennaHeader` ()
- `void printPhoneHeader` ()
- `void printMobileOperatorHeader` ()
- `double r2d` (double x)
- `double d2r` (double x)
- `XMLNode * getNode` (XMLElement *el, const char *name)
- `XMLElement * getFirstChildElement` (XMLElement *el, const char *name) noexcept(false)
- `double getValue` (XMLElement *el, const char *name, double default_value)
- `int getValue` (XMLElement *el, const char *name, int default_value)
- `unsigned long getValue` (XMLElement *el, const char *name, unsigned long default_value)
- `const char * getValue` (XMLElement *el, const char *name, const char *default_value)
- `double getValue` (XMLElement *el, const char *name)

Variables

- `const double PI` = `std::atan(1.0) * 4`

5.1.1 Detailed Description

This namespace contains utility functions that don't belong to any class.

5.1.2 Function Documentation

5.1.2.1 d2r()

```
double utils::d2r (  
    double x ) [inline]
```

Transforms a number from degrees to radians.

Parameters

<i>x</i>	the angle to be transformed from degrees to radians.
----------	--

Returns

the value of *x* in radians.

5.1.2.2 generateFixedPoints()

```
vector<Point*> utils::generateFixedPoints (
    const Map * m,
    unsigned long n,
    unsigned seed )
```

Generates *n* random points on a map. The points have the same locations in all simulations.

Parameters

<i>m</i>	a pointer to a Map object where the points have to be located.
<i>n</i>	the number of points to generate.
<i>seed</i>	the seed of the random number generator used to generate the points.

Returns

a vector of pointers to Point objects.

5.1.2.3 generatePoints()

```
vector<Point*> utils::generatePoints (
    const Map * m,
    unsigned long n,
    double percentHome,
    unsigned seed )
```

Generates *n* random points on a map.

Parameters

<i>m</i>	a pointer to a Map object where the points have to be located.
<i>n</i>	the number of points to generate.
<i>seed</i>	the seed of the random number generator used to generate the points.
<i>percentHome</i>	a percent of the total number of points considered to be "home locations", i.e. each time they have the same values. The rest of the points differ from a simulation to another.

Returns

a vector of pointers to Point objects.

5.1.2.4 getFirstChildElement()

```
XMLElement* utils::getFirstChildElement (
    XMLElement * el,
    const char * name ) [noexcept]
```

Returns a pointer to an XMLElement which is the first child of another XMLElement. This function is used to parse the content of the configuration files.

Parameters

<i>el</i>	a pointer to the parent XMLElement.
<i>name</i>	the name of the child XMLElement.

Returns

a pointer to an XMLElement which is the first child of another XMLElement.

5.1.2.5 getNode()

```
XMLNode* utils::getNode (
    XMLElement * el,
    const char * name )
```

Returns a pointer to an XMLNode with a specific name that belongs to an XMLElement. This function is used to parse the content of the configuration files.

Parameters

<i>el</i>	a pointer to the the XMLElement where to search the XMLNode.
<i>name</i>	the name of the node.

Returns

a pointer to an XMLNode with a specific name that belongs to an XMLElement.

5.1.2.6 getValue() [1/5]

```
double utils::getValue (
    XMLElement * el,
```

```
const char * name,  
double default_value )
```

Returns a double value obtained by converting the text in an XMLNode to a double. In case the node does not exists, this function returns a default value passed as a parameter.

Parameters

<i>el</i>	a pointer to the XMLElement where the XMLNode is located.
<i>name</i>	the name of the XMLNode.
<i>default_value</i>	is the value returned in case the function doesn't find any XMLNode with the specified name under the XMLElement.

Returns

a double value obtained by converting the text in an XMLNode to a double.

5.1.2.7 `getValue()` [2/5]

```
int utils::getValue (  
    XMLElement * el,  
    const char * name,  
    int default_value )
```

Returns an int value obtained by converting the text in an XMLNode to an int. In case the node does not exists, this function returns a default value passed as a parameter.

Parameters

<i>el</i>	a pointer to the XMLElement where the XMLNode is located.
<i>name</i>	the name of the XMLNode.
<i>default_value</i>	is the value returned in case the function doesn't find any XMLNode with the specified name under the XMLElement.

Returns

an int value obtained by converting the text in an XMLNode to an int.

5.1.2.8 `getValue()` [3/5]

```
unsigned long utils::getValue (  
    XMLElement * el,  
    const char * name,  
    unsigned long default_value )
```

Returns an unsigned long value obtained by converting the text in an XMLNode to an unsigned long. In case the node does not exists, this function returns a default value passed as a parameter.

Parameters

<i>el</i>	a pointer to the XMLElement where the XMLNode is located.
<i>name</i>	the name of the XMLNode.
<i>default_value</i>	is the value returned in case the function doesn't find any XMLNode with the specified name under the XMLElement.

Returns

an unsigned long value obtained by converting the text in an XMLNode to an unsigned long.

5.1.2.9 `getValue()` [4/5]

```
const char* utils::getValue (
    XMLElement * el,
    const char * name,
    const char * default_value )
```

Returns a string (const char*) value obtained by converting the text in an XMLNode to a const char* pointer. In case the node does not exists, this function returns a default value passed as a parameter.

Parameters

<i>el</i>	a pointer to the XMLElement where the XMLNode is located.
<i>name</i>	the name of the XMLNode.
<i>default_value</i>	is the value returned in case the function doesn't find any XMLNode with the specified name under the XMLElement.

Returns

a const char* value obtained by converting the text in an XMLNode to a const char*.

5.1.2.10 `getValue()` [5/5]

```
double utils::getValue (
    XMLElement * el,
    const char * name )
```

Returns a double value obtained by converting the text in an XMLNode to a double. In case the node does not exist this method throws an exection.

Parameters

<i>el</i>	a pointer to the XMLElement where the XMLNode is located.
<i>name</i>	the name of the XMLNode.

Returns

a double value obtained by converting the text in an XMLNode to a double

5.1.2.11 printAntennaHeader()

```
void utils::printAntennaHeader ( )
```

Prints out a header containing the names of the member variables from the [Antenna](#) class in a human readable format It is used together with [Antenna::toString\(\)](#) to output the antennas set on console

5.1.2.12 printMobileOperatorHeader()

```
void utils::printMobileOperatorHeader ( )
```

Prints out a header containing the names of the member variables from the [MobileOperator](#) class in a human readable format It is used together with [MobileOperator::toString\(\)](#) to output the mobile network operators set on console

5.1.2.13 printPersonHeader()

```
void utils::printPersonHeader ( )
```

Prints out a header containing the names of the member variables from the [Person](#) class in a human readable format. It is used together with [Person::toString\(\)](#) to output the Persons set on console

5.1.2.14 printPhoneHeader()

```
void utils::printPhoneHeader ( )
```

Prints out a header containing the names of the member variables from the [MobilePhone](#) class in a human readable format It is used together with [MobilePhone::toString\(\)](#) to output the mobile phones set on console

5.1.2.15 r2d()

```
double utils::r2d (
    double x ) [inline]
```

Transforms a number from radians to degrees.

Parameters

x	the angle to be transformed from radians to degrees.
---	--

Returns

the value of x in degrees.

5.1.3 Variable Documentation**5.1.3.1 PI**

```
const double utils::PI = std::atan(1.0) * 4
```

Number PI.

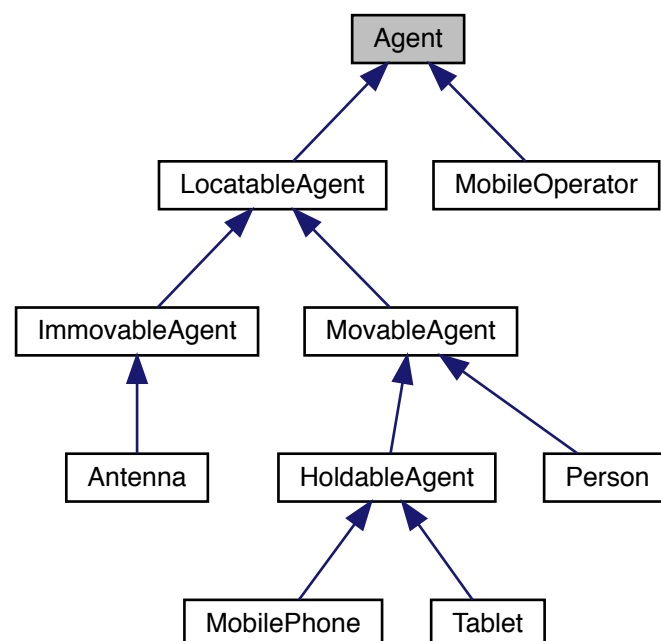
Chapter 6

Class Documentation

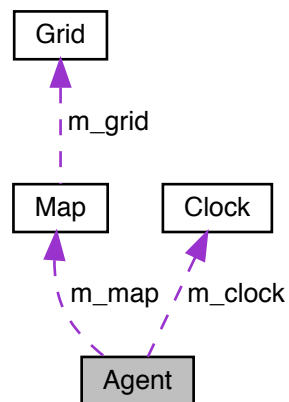
6.1 Agent Class Reference

```
#include <Agent.h>
```

Inheritance diagram for Agent:



Collaboration diagram for Agent:



Public Member Functions

- **Agent** (const **Map** *m, const unsigned long id, const **Clock** *clock)
- virtual **~Agent** ()
- bool **operator==** (const **Agent** &a)
- virtual const string **getName** () const =0
- virtual const string **toString** () const =0
- const **Map** * **getMap** () const
- void **setMap** (const **Map** *map)
- const **Clock** * **getClock** () const
- const unsigned long **getId** () const

Private Attributes

- const **Map** * **m_map**
- const unsigned long **m_id**
- const **Clock** * **m_clock**

6.1.1 Detailed Description

This is an abstract class, the base class for all agents involved in a simulation.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Agent()

```
Agent::Agent (
    const Map * m,
    const unsigned long id,
    const Clock * clock )
```

Constructor of the class. [Agent](#) is the base class for all agents used in the simulator: persons, antennas, devices, mnos. [Agent](#) is an abstract class, users should build specific subclasses.

Parameters

<i>m</i>	- a pointer to the Map object where the simulation take place.
<i>id</i>	- the id of this agent, it uniquely identifies the agent.
<i>clock</i>	- a pointer to a Clock object used by the simulator, the Clock is the same for all agents.

6.1.2.2 ~Agent()

```
virtual Agent::~Agent ( ) [virtual]
```

Default destructor of the class.

6.1.3 Member Function Documentation

6.1.3.1 getClock()

```
const Clock* Agent::getClock ( ) const
```

Returns a pointer to the [Clock](#) object used for simulation. All Agents use the same [Clock](#) object for a simulation.

Returns

6.1.3.2 getId()

```
const unsigned long Agent::getId ( ) const
```

Returns the id of the object.

Returns

the id of the object.

6.1.3.3 getMap()

```
const Map* Agent::getMap ( ) const
```

Getter that returns a pointer to the [Map](#) object passed to the constructor when an object was build.

Returns

a pointer to the [Map](#) object that was passed to the constructor. All agents use the same map for a simulation.

6.1.3.4 getName()

```
virtual const string Agent::getName ( ) const [pure virtual]
```

This function is used to obtain the name of the class. It is a pure virtual function, all subclasses implement it and return the actual name of the class.

Returns

the name of the class.

Implemented in [Antenna](#), [HoldableAgent](#), [MobilePhone](#), [Person](#), [ImmovableAgent](#), [MobileOperator](#), [MovableAgent](#), [LocatableAgent](#), and [Tablet](#).

6.1.3.5 operator==()

```
bool Agent::operator== (
    const Agent & a )
```

The equal operator for agents.

Parameters

<i>a</i>	the object with which we test the equality.
----------	---

Returns

true if this object is the equal to *a*, false otherwise. Two objects are considered to be equal if they have the same id.

6.1.3.6 setMap()

```
void Agent::setMap (
    const Map * map )
```

Sets the [Map](#) to be used by this agent during the simulation. It is not advisable to change the map during a simulation.

Parameters

<i>map</i>	pointer to a Map object.
------------	--

6.1.3.7 toString()

```
virtual const string Agent::toString ( ) const [pure virtual]
```

Builds a string with of the relevant information of the class. It is useful to output on the console or in a file the description of concrete agents.

Returns

a string representation of the class content. The values of the members are written in this string.

Implemented in [Antenna](#), [HoldableAgent](#), [Person](#), [MobilePhone](#), [MobileOperator](#), [MovableAgent](#), [LocatableAgent](#), [Tablet](#), and [ImmovableAgent](#).

6.1.4 Member Data Documentation

6.1.4.1 m_clock

```
const Clock* Agent::m_clock [private]
```

6.1.4.2 m_id

```
const unsigned long Agent::m_id [private]
```

6.1.4.3 m_map

```
const Map* Agent::m_map [private]
```

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

- [include/Agent.h](#)

6.2 AgentsCollection Class Reference

```
#include <AgentsCollection.h>
```

Public Member Functions

- [AgentsCollection](#) ()
- virtual [~AgentsCollection](#) ()
- void [addAgent](#) ([Agent](#) *a)
- [Agent](#) * [deleteAgent](#) ([Agent](#) *a)
- [Agent](#) * [getAgent](#) (const unsigned long id) const
- pair< [um_iterator](#), [um_iterator](#) > [getAgentListByType](#) (const string &agentType)
- [um_iterator](#) end ()
- [um_iterator](#) begin ()
- unsigned long [size](#) ()
- void [printAgents](#) ()

Private Attributes

- unordered_multimap< string, [Agent](#) * > [m_agents](#)

6.2.1 Detailed Description

This is actually a container for all the agents used for simulation. An agent could be an object of one the the derived classes of [Agent](#). The Agents are kept in an unordered_multimap as pairs <string, Agent*> where the first element of the pair is the name of the concrete agent (a person, a mobile device, an antenna, a mno, etc.) and the second element is a pointer to the actual object (agent).

6.2.2 Constructor & Destructor Documentation

6.2.2.1 AgentsCollection()

```
AgentsCollection::AgentsCollection ( )
```

The default constructor of the class.

6.2.2.2 ~AgentsCollection()

```
virtual AgentsCollection::~~AgentsCollection ( ) [virtual]
```

Default destructor: it iterates through the collection of agents and frees the memory allocated for each agent in the collection.

6.2.3 Member Function Documentation

6.2.3.1 addAgent()

```
void AgentsCollection::addAgent (
    Agent * a )
```

Adds a new [Agent](#) to the collection. For performance reasons the [AgentsCollection](#) class keep only a pointer to actual agents (objects).

Parameters

<i>a</i>	a pointer to the object (one of the derived classes of the Agent) to be added to the collection.
-----------------	---

6.2.3.2 begin()

```
um_iterator AgentsCollection::begin ( )
```

Iterator to the first agent of the container.

Returns

a random access iterator pointing to the first element (agent) in the container. If the container is empty, the returned iterator value shall not be dereferenced.

6.2.3.3 deleteAgent()

```
Agent* AgentsCollection::deleteAgent (
    Agent * a )
```

Removes an object from the collection.

Parameters

<i>a</i>	a pointer to the object to be removed from the collection.
-----------------	--

Returns**6.2.3.4 end()**

```
um_iterator AgentsCollection::end ( )
```

Iterator to the past-the-end of the collection. It does not point to any agent, and thus shall not be dereferenced.

Returns

an iterator referring to the past-the-end element in the agents container. If the container is empty, this function returns the same as `AgentsCollection::begin()`.

6.2.3.5 getAgent()

```
Agent* AgentsCollection::getAgent (
    const unsigned long id ) const
```

Returns a pointer to an agent from the collection. The agent/object is identified by its id.

Parameters

<i>id</i>	the id of the agent to be returned.
-----------	-------------------------------------

Returns

a pointer to the agent with the id equal to the parameter id. If there is no agent with the provided id, this method returns nullptr.

6.2.3.6 getAgentListByType()

```
pair<um_iterator, um_iterator> AgentsCollection::getAgentListByType (
    const string & agentType )
```

This method is used to get a subset with a certain type of agents: persons, mobile phones etc.

Parameters

<i>agentType</i>	is the name of the class of agents that the user wants to retrieve from the collection of all agents.
------------------	---

Returns

a std::pair of iterators of type unordered_multimap<string, Agent*>::iterator that can be used to iterate through to subset of the agents.

6.2.3.7 printAgents()

```
void AgentsCollection::printAgents ( )
```

Print out all agents in the collection.

6.2.3.8 size()

```
unsigned long AgentsCollection::size ( )
```

Returns

the number of elements in the container. This is the number of actual objects held in the container, which is not necessarily equal to its storage capacity.

6.2.4 Member Data Documentation

6.2.4.1 m_agents

```
unordered_multimap<string, Agent*> AgentsCollection::m_agents [private]
```

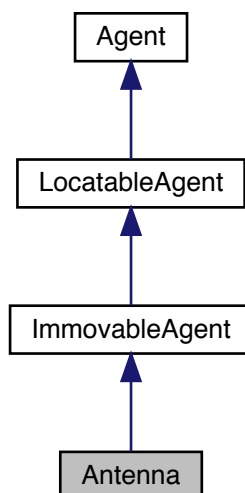
The documentation for this class was generated from the following file:

- include/[AgentsCollection.h](#)

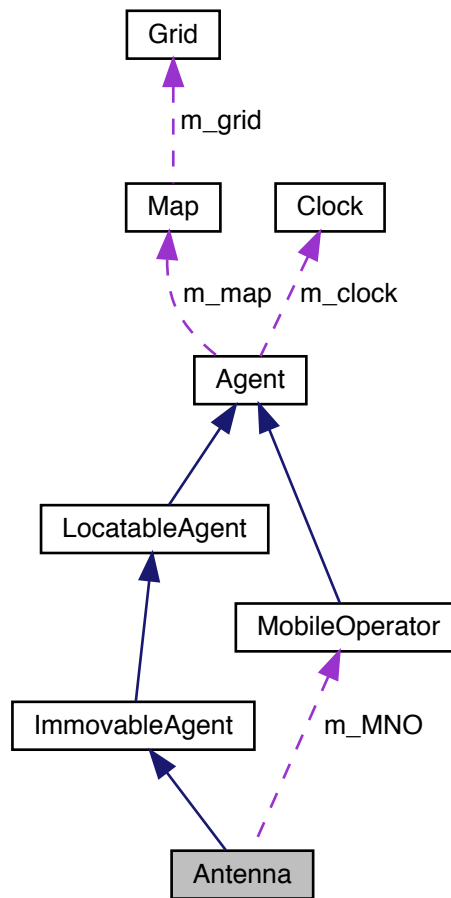
6.3 Antenna Class Reference

```
#include <Antenna.h>
```

Inheritance diagram for Antenna:



Collaboration diagram for Antenna:



Public Member Functions

- `Antenna` (const `Map` *m, const unsigned long id, Point *initPosition, const `Clock` *clock, double attenuation↔Factor, double power, unsigned long maxConnections, double smid, double ssteep, `AntennaType` type)
- `Antenna` (const `Map` *m, const `Clock` *clock, const unsigned long id, `XMLElement` *el, vector< `MobileOperator` * > mnos)
- virtual `~Antenna` ()
- const string `getName` () const override
- const string `toString` () const override
- double `getPLE` () const
- void `setPLE` (double ple)
- double `getPower` () const
- void `setPower` (double power)
- unsigned long `getMaxConnections` () const
- void `setMaxConnections` (int maxConnections)
- bool `tryRegisterDevice` (`HoldableAgent` *device)
- void `attachDevice` (`HoldableAgent` *device)

- void `dettachDevice` (`HoldableAgent *device`)
- `AntennaType` `getType` () const
- void `setType` (`AntennaType` type)
- double `S` (double dist) const
- double `getSmid` () const
- void `setSmid` (double smid)
- double `getSSteep` () const
- void `setSSteep` (double sSteep)
- double `computeSignalQuality` (const `Point *p`) const
- double `computeSignalQuality` (const `Coordinate c`) const
- double `computePower` (const `Point *p`) const
- double `computePower` (const `Coordinate c`) const
- `Geometry *` `getCoverageArea` ()
- `MobileOperator *` `getMNO` () const
- string `getAntennaOutputFileName` () const
- double `getRmax` () const
- double `getSmin` () const
- string `dumpCell` () const
- double `computeSignalStrength` (const `Point *p`) const
- double `computeSignalStrength` (const `Coordinate c`) const
- double `computeSignalMeasure` (`HoldableAgent::CONNECTION_TYPE` handoverType, const `Coordinate c`) const
- `HoldableAgent::CONNECTION_TYPE` `getHandoverMechanism` () const
- void `setHandoverMechanism` (`HoldableAgent::CONNECTION_TYPE` handoverMechanism)

Private Member Functions

- bool `alreadyRegistered` (`HoldableAgent *ag`)
- void `registerEvent` (`HoldableAgent *ag`, const `EventType` event, const bool verbose)
- unsigned long `getNumActiveConections` ()
- double `S0` () const
- double `SDist` (double dist) const
- double `computeSignalQualityOmnidirectional` (const `Coordinate c`) const
- double `computeSignalQualityDirectional` (const `Coordinate c`) const
- double `computeSignalStrengthDirectional` (const `Coordinate c`) const
- double `computeSignalStrengthOmnidirectional` (const `Coordinate c`) const
- void `setLocationWithElevation` ()
- double `projectToEPlane` (double b, double c, double beta) const
- vector< pair< double, double > > `createMapping` (double dbBack) const
- double `getMin3db` (double sd, double dbBack) const
- double `norm_dBLoss` (double angle, double dbBack, double sd) const
- double `normalizeAngle` (double angle) const
- double `searchMin` (double dg, vector< pair< double, double > > _3dbDegrees) const
- double `findSD` (double beamWidth, double dbBack, vector< pair< double, double > > &mapping) const
- `Geometry *` `getCoverageAreaOmnidirectional` ()
- `Geometry *` `getCoverageAreaDirectional` ()

Private Attributes

- double `m_ple`
- double `m_power`
- unsigned long `m_maxConnections`
- double `m_Smid`
- double `m_SSteep`
- Geometry * `m_cell`
- vector< `HoldableAgent` * > `m_devices`
- `AntennaType` `m_type`
- ofstream `m_file`
- double `m_S0`
- double `m_height`
- double `m_tilt`
- double `m_beam_V`
- double `m_beam_H`
- double `m_azim_dB_Back`
- double `m_elev_dB_Back`
- double `m_direction`
- `MobileOperator` * `m_MNO`
- double `m_rmax`
- double `m_Smin`
- double `m_Qmin`
- vector< pair< double, double > > `m_mapping_azim`
- vector< pair< double, double > > `m_mapping_elev`
- double `m_sd_azim`
- double `m_sd_elev`
- `HoldableAgent::CONNECTION_TYPE` `m_handoverMechanism`

6.3.1 Detailed Description

This class simulates an antenna of the mobile phone network.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 `Antenna()` [1/2]

```
Antenna::Antenna (
    const Map * m,
    const unsigned long id,
    Point * initPosition,
    const Clock * clock,
    double attenuationFactor,
    double power,
    unsigned long maxConnections,
    double smid,
    double ssteep,
    AntennaType type ) [explicit]
```

Constructor of the class. It builds an object providing directly the values for each parameter of the antenna. This constructor is used only for testing purposes. For a real simulation the other constructor of the class should be used.

Parameters

<i>m</i>	a pointer to the Map object used for the simulation
<i>id</i>	the id of the Antenna
<i>initPosition</i>	the position of the antenna on the map
<i>clock</i>	a pointer to the Clock object used for the simulation
<i>attenuationFactor</i>	the attenuation factor of the surrounding environment. In real life, it takes values between 2 (in open field) and 6 (inside buildings).
<i>power</i>	the power of the antenna in Watts.
<i>maxConnections</i>	the maximum number of the connections that the antenna can accept.
<i>smid</i>	is a parameter of an antenna. The significance of this parameter is described in mobloc R package.
<i>ssteep</i>	is a parameter of an antenna. The significance of this parameter is described in mobloc R package.
<i>type</i>	it could have two values AntennaType::OMNIDIRECTIONAL for omnidirectional antennas and AntennaType::DIRECTIONAL for directional antennas.

6.3.2.2 [Antenna\(\)](#) [2/2]

```
Antenna::Antenna (
    const Map * m,
    const Clock * clock,
    const unsigned long id,
    XMLElement * el,
    vector< MobileOperator * > mnos ) [explicit]
```

Constructor of the class. It builds an object taking the value of the antenna' parameters from an XML Element, usually when an [Antenna](#) object is built reading the xml configuration file.

Parameters

<i>m</i>	a pointer to the Map object used for the simulation
<i>clock</i>	a pointer to the Clock object used for the simulation
<i>id</i>	the id of the Antenna
<i>el</i>	the XML Element containing the parameters of the Antenna
<i>mnos</i>	a vector with pointers to MobileOperator objects.

6.3.2.3 [~Antenna\(\)](#)

```
virtual Antenna::~Antenna ( ) [virtual]
```

Destructor of the class. It closes the file where the [Antenna](#) dumps the registered events during the simulation.

6.3.3 Member Function Documentation

6.3.3.1 alreadyRegistered()

```
bool Antenna::alreadyRegistered (
    HoldableAgent * ag ) [private]
```

6.3.3.2 attachDevice()

```
void Antenna::attachDevice (
    HoldableAgent * device )
```

Connects a new mobile device and outputs an event `EventType::ATTACH_DEVICE` in the output file. Internally, the antenna keeps a vector with all connected mobile devices. When a new mobile device is connected it is added to this vector.

Parameters

<i>device</i>	a pointer to the object that represents the mobile device connected to this antenna.
---------------	--

6.3.3.3 computePower() [1/2]

```
double Antenna::computePower (
    const Point * p ) const
```

Computes the power of the signal given by an antenna in a certain location.

Parameters

<i>p</i>	the location where we want to compute the power of the signal.
----------	--

Returns

the power of the signal in the location given by Point p.

6.3.3.4 computePower() [2/2]

```
double Antenna::computePower (
    const Coordinate c ) const
```

Computes the power of the signal given by an antenna in a certain location.

Parameters

<i>c</i>	the location where we want to compute the power of the signal.
----------	--

Returns

the power of the signal in the location given by Coordinate *c*.

6.3.3.5 computeSignalMeasure()

```
double Antenna::computeSignalMeasure (
    HoldableAgent::CONNECTION_TYPE handoverType,
    const Coordinate c ) const
```

compute the signal strength, signal quality or signal power depending on the value of the handoverType parameter

Parameters

<i>handoverType</i>	the handover mechanism: signal quality, signal strength, signal power
<i>c</i>	- a pointer to a coordinate that defines the location where the signal quality/strength/power should be computed

Returns

the signal strength, signal quality or signal power depending on the value of the handoverType parameter

6.3.3.6 computeSignalQuality() [1/2]

```
double Antenna::computeSignalQuality (
    const Point * p ) const
```

Computes the signal quality given by an antenna in a certain location.

Parameters

<i>p</i>	the location where we want to compute the signal quality.
----------	---

Returns

the signal quality.

6.3.3.7 computeSignalQuality() [2/2]

```
double Antenna::computeSignalQuality (
    const Coordinate c ) const
```

Computes the signal quality given by an antenna in a certain location.

Parameters

c	represents the coordinates of the location where we want to compute the signal quality.
-----	---

Returns

the signal quality.

6.3.3.8 computeSignalQualityDirectional()

```
double Antenna::computeSignalQualityDirectional (  
    const Coordinate  $c$  ) const [private]
```

6.3.3.9 computeSignalQualityOmnidirectional()

```
double Antenna::computeSignalQualityOmnidirectional (  
    const Coordinate  $c$  ) const [private]
```

6.3.3.10 computeSignalStrength() [1/2]

```
double Antenna::computeSignalStrength (  
    const Point *  $p$  ) const
```

Computes the signal strength given by an antenna in a certain location.

Parameters

p	the location where we want to compute the signal strength.
-----	--

Returns

the signal strength.

6.3.3.11 computeSignalStrength() [2/2]

```
double Antenna::computeSignalStrength (  
    const Coordinate  $c$  ) const
```

Computes the signal strength given by an antenna in a certain location.

Parameters

<i>c</i>	the location where we want to compute the signal strength.
----------	--

Returns

the signal strength.

6.3.3.12 computeSignalStrengthDirectional()

```
double Antenna::computeSignalStrengthDirectional (  
    const Coordinate c ) const [private]
```

6.3.3.13 computeSignalStrengthOmnidirectional()

```
double Antenna::computeSignalStrengthOmnidirectional (  
    const Coordinate c ) const [private]
```

6.3.3.14 createMapping()

```
vector<pair<double, double> > Antenna::createMapping (  
    double dbBack ) const [private]
```

6.3.3.15 detachDevice()

```
void Antenna::detachDevice (  
    HoldableAgent * device )
```

Disconnects a mobile device from the antenna and outputs an event EventType::DEATCH_DEVICE in the output file. Internally, the mobile device is removed from the vector of the connected mobile devices.

Parameters

<i>device</i>	
---------------	--

6.3.3.16 dumpCell()

```
string Antenna::dumpCell ( ) const
```

Builds a wkt string that represents the coverage area of this antenna.

Returns

a wkt string that represents the coverage area of this antenna.

6.3.3.17 findSD()

```
double Antenna::findSD (
    double beamWidth,
    double dbBack,
    vector< pair< double, double >> & mapping ) const [private]
```

6.3.3.18 getAntennaOutputFileName()

```
string Antenna::getAntennaOutputFileName ( ) const
```

Builds the name of the output file where the events registered by this antenna during a simulation are saved.

Returns

the name of the output file where the events registered by this antenna during a simulation are saved.

6.3.3.19 getCoverageArea()

```
Geometry* Antenna::getCoverageArea ( )
```

Computes the coverage area of an antenna. It is defined as the area where the signal strength is greater than S_min

Returns

a Polygon* representing the coverage area of the antenna.

6.3.3.20 getCoverageAreaDirectional()

```
Geometry* Antenna::getCoverageAreaDirectional ( ) [private]
```

6.3.3.21 getCoverageAreaOmnidirectional()

```
Geometry* Antenna::getCoverageAreaOmnidirectional ( ) [private]
```

6.3.3.22 getHandoverMechanism()

```
HoldableAgent::CONNECTION_TYPE Antenna::getHandoverMechanism ( ) const
```

6.3.3.23 getMaxConnections()

```
unsigned long Antenna::getMaxConnections ( ) const
```

Returns the maximum number of mobile devices that an antenna can connect.

6.3.3.24 getMin3db()

```
double Antenna::getMin3db (
    double sd,
    double dbBack ) const [private]
```

6.3.3.25 getMNO()

```
MobileOperator* Antenna::getMNO ( ) const
```

Returns a pointer to an MNO object representing the MNO that own this antenna.

Returns

a pointer to an MNO object representing the MNO that own this antenna.

6.3.3.26 getName()

```
const string Antenna::getName ( ) const [override], [virtual]
```

Overrides the same method from the superclass.

Returns

the name of the class, i.e. "Antenna"

Implements [Agent](#).

6.3.3.27 getNumActiveConnections()

```
unsigned long Antenna::getNumActiveConnections ( ) [private]
```

6.3.3.28 getPLE()

```
double Antenna::getPLE ( ) const
```

Returns the surrounding environment' path loss exponent of the signal.

Returns

the signals' path loss exponent of the surrounding environment. In real life, it takes values between 2 (in open field) and 6 (inside buildings).

6.3.3.29 getPower()

```
double Antenna::getPower ( ) const
```

Returns the power of an antenna in Watts at the location of antenna. This power decreases with a power of the distance from antenna.

Returns

the power of an antenna in Watts.

6.3.3.30 getRmax()

```
double Antenna::getRmax ( ) const
```

Computes the radius of the coverage area for an omnidirectional antenna. This area is a circle where the signal strength is greater than S_min.

Returns

the radius of the coverage area for an omnidirectional antenna.

6.3.3.31 getSmid()

```
double Antenna::getSmid ( ) const
```

Returns the value of the Smid antenna parameter.

Returns

the value of the Smid antenna parameter.

6.3.3.32 getSmin()

```
double Antenna::getSmin ( ) const
```

Returns the value of the minimum signal strength that defines the coverage area of this antenna.

Returns

the value of the minimum signal strength that defines the coverage area of this antenna.

6.3.3.33 getSSteep()

```
double Antenna::getSSteep ( ) const
```

Returns the value of the Ssteep antenna parameter.

Returns

the value of the Ssteep antenna parameter.

6.3.3.34 getType()

```
AntennaType Antenna::getType ( ) const
```

Returns the antenna type: omnidirectional or directional

Returns

the antenna type : [AntennaType::OMNIDIRECTIONAL](#) or [AntennaType::DIRECTIONAL](#).

6.3.3.35 norm_dBloss()

```
double Antenna::norm_dBloss (
    double angle,
    double dbBack,
    double sd ) const [private]
```

6.3.3.36 normalizeAngle()

```
double Antenna::normalizeAngle (
    double angle ) const [private]
```

6.3.3.37 projectToEPlane()

```
double Antenna::projectToEPlane (
    double b,
    double c,
    double beta ) const [private]
```

6.3.3.38 registerEvent()

```
void Antenna::registerEvent (
    HoldableAgent * ag,
    const EventType event,
    const bool verbose ) [private]
```

6.3.3.39 S()

```
double Antenna::S (
    double dist ) const
```

Computes the signal strength at the distance dist from antenna location.

Parameters

<i>dist</i>	the distance from antenna location.
-------------	-------------------------------------

Returns

the signal strength.

6.3.3.40 S0()

```
double Antenna::S0 ( ) const [private]
```

6.3.3.41 SDist()

```
double Antenna::SDist (
    double dist ) const [private]
```

6.3.3.42 searchMin()

```
double Antenna::searchMin (
    double dg,
    vector< pair< double, double >> _3dBDegrees ) const [private]
```

6.3.3.43 setHandoverMechanism()

```
void Antenna::setHandoverMechanism (
    HoldableAgent::CONNECTION_TYPE handoverMechanism )
```

6.3.3.44 setLocationWithElevation()

```
void Antenna::setLocationWithElevation ( ) [private]
```

6.3.3.45 setMaxConnections()

```
void Antenna::setMaxConnections (
    int maxConnections )
```

Sets the number of mobile devices that an antenna can connect.

Parameters

<i>maxConnections</i>	the number of mobile devices that an antenna can connect.
-----------------------	---

6.3.3.46 setPLE()

```
void Antenna::setPLE (
    double ple )
```

Sets the surrounding environment' path loss exponent of the signal for an antenna.

Parameters

<i>ple</i>	the value of the surrounding environment' path loss exponent of the signal. In real life, it takes values between 2 (in open field) and 6 (inside buildings).
------------	---

6.3.3.47 setPower()

```
void Antenna::setPower (
    double power )
```

Sets the power of an antenna.

Parameters

<i>power</i>	the value of the antenna's power.
--------------	-----------------------------------

6.3.3.48 setSmid()

```
void Antenna::setSmid (
    double smid )
```

Sets the value of the Smid antenna parameter.

Parameters

<i>smid</i>	the value of the Smid antenna parameter.
-------------	--

6.3.3.49 setSSteep()

```
void Antenna::setSSteep (
    double sSteep )
```

Sets the value of the Ssteep antenna parameter.

Parameters

<i>sSteep</i>	the value of the Ssteep antenna parameter.
---------------	--

6.3.3.50 setType()

```
void Antenna::setType (
    AntennaType type )
```

Sets the antenna type.

Parameters

<i>type</i>	the antenna type. It could take the following two values: AntennaType::OMNIDIRECTIONAL or AntennaType::DIRECTIONAL .
-------------	--

6.3.3.51 toString()

```
const string Antenna::toString ( ) const [override], [virtual]
```

Overrides the same method from the superclass. It is used to write the characteristics of the [Antenna](#) in a file or console.

Returns

a string that describes the parameters of the [Antenna](#).

Implements [Agent](#).

6.3.3.52 tryRegisterDevice()

```
bool Antenna::tryRegisterDevice (
    HoldableAgent * device )
```

Tries to register a mobile device as being connected to this antenna.

Parameters

<i>device</i>	a pointer to the object that represents a mobile device.
---------------	--

Returns

true if the connection is successful, false otherwise.

6.3.4 Member Data Documentation

6.3.4.1 m_azim_dB_Back

```
double Antenna::m_azim_dB_Back [private]
```

6.3.4.2 m_beam_H

```
double Antenna::m_beam_H [private]
```

6.3.4.3 m_beam_V

```
double Antenna::m_beam_V [private]
```

6.3.4.4 m_cell

```
Geometry* Antenna::m_cell [private]
```

6.3.4.5 m_devices

```
vector<HoldableAgent*> Antenna::m_devices [private]
```

6.3.4.6 m_direction

```
double Antenna::m_direction [private]
```

6.3.4.7 m_elev_dB_Back

```
double Antenna::m_elev_dB_Back [private]
```

6.3.4.8 m_file

```
ofstream Antenna::m_file [private]
```

6.3.4.9 m_handoverMechanism

```
HoldableAgent::CONNECTION_TYPE Antenna::m_handoverMechanism [private]
```

6.3.4.10 m_height

```
double Antenna::m_height [private]
```

6.3.4.11 m_mapping_azim

```
vector<pair<double, double> > Antenna::m_mapping_azim [private]
```

6.3.4.12 m_mapping_elev

```
vector<pair<double, double> > Antenna::m_mapping_elev [private]
```

6.3.4.13 m_maxConnections

```
unsigned long Antenna::m_maxConnections [private]
```

6.3.4.14 m_MNO

```
MobileOperator* Antenna::m_MNO [private]
```

6.3.4.15 m_ple

```
double Antenna::m_ple [private]
```

6.3.4.16 m_power

```
double Antenna::m_power [private]
```

6.3.4.17 m_Qmin

```
double Antenna::m_Qmin [private]
```

6.3.4.18 m_rmax

```
double Antenna::m_rmax [private]
```

6.3.4.19 m_S0

```
double Antenna::m_S0 [private]
```

6.3.4.20 m_sd_azim

```
double Antenna::m_sd_azim [private]
```

6.3.4.21 m_sd_elev

```
double Antenna::m_sd_elev [private]
```

6.3.4.22 m_Smid

```
double Antenna::m_Smid [private]
```

6.3.4.23 m_Smin

```
double Antenna::m_Smin [private]
```

6.3.4.24 m_SSteep

```
double Antenna::m_SSteep [private]
```

6.3.4.25 m_tilt

```
double Antenna::m_tilt [private]
```

6.3.4.26 m_type

```
AntennaType Antenna::m_type [private]
```

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

- include/[Antenna.h](#)

6.4 AntennaInfo Class Reference

```
#include <AntennaInfo.h>
```

Public Member Functions

- [AntennaInfo](#) (const unsigned long time, const unsigned long antennald, const unsigned long event, const unsigned long deviceId, const double x, const double y)
- unsigned long [getAntennald](#) () const
- unsigned long [getDeviceId](#) () const
- unsigned long [getEventCode](#) () const
- unsigned long [getTime](#) () const
- double [getX](#) () const
- double [getY](#) () const
- const string [toString](#) () const
- bool [operator<](#) (const [AntennaInfo](#) &ai) const

Private Attributes

- unsigned long [m_time](#)
- unsigned long [m_antennald](#)
- unsigned long [m_eventCode](#)
- unsigned long [m_deviceId](#)
- double [m_x](#)
- double [m_y](#)

6.4.1 Detailed Description

This class is used to encapsulate all the information about an event generated by an antenna: the timestamp, antennald, event code, the device id, the exact location of the event.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 AntennaInfo()

```
AntennaInfo::AntennaInfo (
    const unsigned long time,
    const unsigned long antennaId,
    const unsigned long event,
    const unsigned long deviceId,
    const double x,
    const double y )
```

Constructor of the class, builds an object with the values of the fields provided as arguments.

Parameters

<i>time</i>	the timestamp of the event.
<i>antennaId</i>	the id of the antenna that registered an event.
<i>event</i>	the event code.
<i>deviceId</i>	the id of the device that generated the event.
<i>x</i>	x coordinate of the device when the event was generated.
<i>y</i>	y coordinate of the device when the event was generated.

6.4.3 Member Function Documentation

6.4.3.1 getAntennald()

```
unsigned long AntennaInfo::getAntennaId ( ) const
```

Returns

the id of the antenna that registered the event.

6.4.3.2 getDeviceId()

```
unsigned long AntennaInfo::getDeviceId ( ) const
```

Returns

the id of the device that generated the event.

6.4.3.3 getEventCode()

```
unsigned long AntennaInfo::getEventCode ( ) const
```

Returns

the event code. It could take the following values: 0 - a device is connected to an antenna, 1 - a device is disconnected from an antenna, 2 - a device is already connected to the antenna that registered the event, 3 - a device was detected in the coverage area of an antenna but the connection to the antenna failed (from different reasons).

6.4.3.4 getTime()

```
unsigned long AntennaInfo::getTime ( ) const
```

Returns

the timestamp of the event.

6.4.3.5 getX()

```
double AntennaInfo::getX ( ) const
```

Returns

x coordinate of the device that generated the event.

6.4.3.6 getY()

```
double AntennaInfo::getY ( ) const
```

Returns

y coordinate of the device that generated the event.

6.4.3.7 operator<()

```
bool AntennaInfo::operator< (
    const AntennaInfo & ai ) const
```

Overloaded operator to compare to objects

Parameters

<i>ai</i>	the other object to compare to.
-----------	---------------------------------

Returns

true if this object is less than ai, false otherwise. An object is less than another object if the timestamp value of the first object is less than the timestamp of the second one.

6.4.3.8 toString()

```
const string AntennaInfo::toString ( ) const
```

Returns

a string representation of an object of this class.

6.4.4 Member Data Documentation

6.4.4.1 m_antennaId

```
unsigned long AntennaInfo::m_antennaId [private]
```

6.4.4.2 m_deviceId

```
unsigned long AntennaInfo::m_deviceId [private]
```

6.4.4.3 m_eventCode

```
unsigned long AntennaInfo::m_eventCode [private]
```

6.4.4.4 m_time

```
unsigned long AntennaInfo::m_time [private]
```

6.4.4.5 m_x

```
double AntennaInfo::m_x [private]
```

6.4.4.6 m_y

```
double AntennaInfo::m_y [private]
```

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

- [include/AntennaInfo.h](#)

6.5 Clock Class Reference

```
#include <Clock.h>
```

Public Member Functions

- [Clock](#) ()
- [Clock](#) (unsigned long start, unsigned long end, unsigned long incr)
- virtual [~Clock](#) ()
- unsigned long [tick](#) ()
- unsigned long [getCurrentTime](#) () const
- void [setCurrentTime](#) (unsigned long currentTime)
- unsigned long [getIncrement](#) () const
- void [setIncrement](#) (unsigned long increment)
- unsigned long [getInitialTime](#) () const
- void [setInitialTime](#) (unsigned long initialTime)
- time_t [realTime](#) ()
- unsigned long [getFinalTime](#) () const
- void [setFinalTime](#) (unsigned long finalTime)
- void [reset](#) ()

Private Attributes

- unsigned long [m_initialTime](#)
- unsigned long [m_currentTime](#)
- unsigned long [m_increment](#)
- unsigned long [m_finalTime](#)

6.5.1 Detailed Description

This is the clock used to synchronize the simulation. All the agents and all other objects involved in a simulation use the same [Clock](#) object. The [Clock](#) will be initialized with the value of the starting and ending time of the simulation and will keep a current time during the simulation. At each step of the simulation the current time will be increased by an increment. Starting time, ending time, current time and the time increment are only conventional units and they do not depend in any way on the real clock of the computer.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 [Clock\(\)](#) [1/2]

```
Clock::Clock ( )
```

Default constructor

6.5.2.2 [Clock\(\)](#) [2/2]

```
Clock::Clock (
    unsigned long start,
    unsigned long end,
    unsigned long incr )
```

Constructor of the class. It takes the starting and ending time of the simulation as parameters as well as the time increment.

Parameters

<i>start</i>	the initial moment when the simulation starts.
<i>end</i>	the time when simulation ends.
<i>incr</i>	the time increment. At each step of the simulation the current time is incremented by this quantity.

6.5.2.3 [~Clock\(\)](#)

```
virtual Clock::~~Clock ( ) [virtual]
```

Default destructor

6.5.3 Member Function Documentation

6.5.3.1 `getCurrentTime()`

```
unsigned long Clock::getCurrentTime ( ) const
```

Returns

the current time of the simulator.

6.5.3.2 `getFinalTime()`

```
unsigned long Clock::getFinalTime ( ) const
```

Returns

the ending time of the simulation.

6.5.3.3 `getIncrement()`

```
unsigned long Clock::getIncrement ( ) const
```

Returns

the time increment used in simulation.

6.5.3.4 `getInitialTime()`

```
unsigned long Clock::getInitialTime ( ) const
```

Returns

the starting time of the simulation.

6.5.3.5 `realTime()`

```
time_t Clock::realTime ( )
```

Returns

the real time read from the computer clock. It is used only to register the exact date and time of a simulation.

6.5.3.6 `reset()`

```
void Clock::reset ( )
```

Resets the current time and makes it equal to the starting time such that a new simulation can begin.

6.5.3.7 `setCurrentTime()`

```
void Clock::setCurrentTime (
    unsigned long currentTime )
```

Sets the current time of the simulator.

Parameters

<i>currentTime</i>	the value of the current time to be set.
--------------------	--

6.5.3.8 `setFinalTime()`

```
void Clock::setFinalTime (
    unsigned long finalTime )
```

Sets the ending time of a simulation.

Parameters

<i>finalTime</i>	the value of the ending time of a simulation.
------------------	---

6.5.3.9 `setIncrement()`

```
void Clock::setIncrement (
    unsigned long increment )
```

Sets the time increment to be used in a simulation.

Parameters

<i>increment</i>	the value of the time increment.
------------------	----------------------------------

6.5.3.10 setInitialTime()

```
void Clock::setInitialTime (
    unsigned long initialTime )
```

Sets the starting time of the simulation.

Parameters

<i>initialTime</i>	the value of the starting time of the simulation.
--------------------	---

6.5.3.11 tick()

```
unsigned long Clock::tick ( )
```

increments the current time.

Returns

the current time after incrementation.

6.5.4 Member Data Documentation**6.5.4.1 m_currentTime**

```
unsigned long Clock::m_currentTime [private]
```

6.5.4.2 m_finalTime

```
unsigned long Clock::m_finalTime [private]
```


6.5.4.3 m_increment

```
unsigned long Clock::m_increment [private]
```

6.5.4.4 m_initialTime

```
unsigned long Clock::m_initialTime [private]
```

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

- [include/Clock.h](#)

6.6 Constants Class Reference

```
#include <Constants.h>
```

Static Public Attributes

- static const double [PHONE_POWER_THRESHOLD](#)
- static const double [PHONE_QUALITY_THRESHOLD](#)
- static const double [PHONE_STRENGTH_THRESHOLD](#)
- static const double [PHONE_CONNECTION_THRESHOLD](#)
- static const double [ANTENNA_POWER](#)
- static const double [ATT_FACTOR](#)
- static const unsigned long [ANTENNA_MAX_CONNECTIONS](#)
- static const double [ANTENNA_S_MID](#)
- static const double [ANTENNA_S_STEEP](#)
- static const unsigned long [SIM_NO_PERSONS](#)
- static const unsigned long [SIM_NO_ANTENNAS](#)
- static const unsigned long [SIM_NO_MOBILE_PHONES](#)
- static const unsigned long [SIM_START_TIME](#)
- static const unsigned long [SIM_END_TIME](#)
- static const unsigned long [SIM_INCREMENT_TIME](#)
- static const unsigned long [SIM_STAY_TIME](#)
- static const unsigned long [SIM_INTERVAL_BETWEEN_STAYS](#)
- static const char [sep](#)
- static const char * [GRID_FILE_NAME](#)
- static const unsigned long [GRID_X_ORIG](#)
- static const unsigned long [GRID_Y_ORIG](#)
- static const double [GRID_DIM_TILE_X](#)
- static const double [GRID_DIM_TILE_Y](#)
- static const char * [PROB_FILE_NAME_PREFIX](#)
- static const char * [PERSONS_FILE_NAME](#)
- static const char * [ANTENNAS_FILE_NAME](#)
- static const [PriorType](#) [PRIOR_PROBABILITY](#)
- static const double [ANTENNA_HEIGHT](#)
- static const double [ANTENNA_TILT](#)

- static const double [ANTENNA_AZIM_DB_BACK](#)
- static const double [ANTENNA_ELEV_DB_BACK](#)
- static const double [ANTENNA_BEAM_H](#)
- static const double [ANTENNA_BEAM_V](#)
- static const double [ANTENNA_DIRECTION](#)
- static const unsigned int [ANTENNA_MAPPING_N](#)
- static const unsigned int [ANTENNA_MIN_3_DB](#)
- static const double [ANTENNA_SMIN](#)
- static const double [ANTENNA_QMIN](#)
- static const unsigned int [SIM_NUM_MNO](#)
- static const char * [SIM_DEFAULT_MNO_NAME](#)
- static const double [SIM_PROB_MOBILE_PHONE](#)
- static const double [SIM_PROB_SECOND_MOBILE_PHONE](#)
- static const double [SIM_TREND_ANGLE_1](#)
- static const double [SIM_TREND_ANGLE_2](#)
- static const int [RANDOM_SEED](#)

6.6.1 Detailed Description

These are some constants used in the process of the simulation, most of them are only used for testing and rapid development of some methods, the real values of the parameters being read from the configuration files.

6.6.2 Member Data Documentation

6.6.2.1 ANTENNA_AZIM_DB_BACK

```
const double Constants::ANTENNA_AZIM_DB_BACK [static]
```

6.6.2.2 ANTENNA_BEAM_H

```
const double Constants::ANTENNA_BEAM_H [static]
```

6.6.2.3 ANTENNA_BEAM_V

```
const double Constants::ANTENNA_BEAM_V [static]
```

6.6.2.4 ANTENNA_DIRECTION

```
const double Constants::ANTENNA_DIRECTION [static]
```

6.6.2.5 ANTENNA_ELEV_DB_BACK

```
const double Constants::ANTENNA_ELEV_DB_BACK [static]
```

6.6.2.6 ANTENNA_HEIGHT

```
const double Constants::ANTENNA_HEIGHT [static]
```

the antenna height

6.6.2.7 ANTENNA_MAPPING_N

```
const unsigned int Constants::ANTENNA_MAPPING_N [static]
```

6.6.2.8 ANTENNA_MAX_CONNECTIONS

```
const unsigned long Constants::ANTENNA_MAX_CONNECTIONS [static]
```

The maximum number of devices an antenna can connect.

6.6.2.9 ANTENNA_MIN_3_DB

```
const unsigned int Constants::ANTENNA_MIN_3_DB [static]
```

6.6.2.10 ANTENNA_POWER

```
const double Constants::ANTENNA_POWER [static]
```

[Antenna](#) power in Watts.

6.6.2.11 ANTENNA_QMIN

```
const double Constants::ANTENNA_QMIN [static]
```

6.6.2.12 ANTENNA_S_MID

```
const double Constants::ANTENNA_S_MID [static]
```

The Smid parameter of an antenna

6.6.2.13 ANTENNA_S_STEEP

```
const double Constants::ANTENNA_S_STEEP [static]
```

The Sstepp parameter of an antenna

6.6.2.14 ANTENNA_SMIN

```
const double Constants::ANTENNA_SMIN [static]
```

6.6.2.15 ANTENNA_TILT

```
const double Constants::ANTENNA_TILT [static]
```

6.6.2.16 ANTENNAS_FILE_NAME

```
const char* Constants::ANTENNAS_FILE_NAME [static]
```

The name of the file where the exact positions of the antennas are saved during simulation. They are needed for later analysis.

6.6.2.17 ATT_FACTOR

```
const double Constants::ATT_FACTOR [static]
```

Attenuation factor of the signal. It usually takes values between 2 in open field and 6 inside buildings

6.6.2.18 GRID_DIM_TILE_X

```
const double Constants::GRID_DIM_TILE_X [static]
```

6.6.2.19 GRID_DIM_TILE_Y

```
const double Constants::GRID_DIM_TILE_Y [static]
```

6.6.2.20 GRID_FILE_NAME

```
const char* Constants::GRID_FILE_NAME [static]
```

The name of the file where the description of the grid is saved

6.6.2.21 GRID_X_ORIG

```
const unsigned long Constants::GRID_X_ORIG [static]
```

6.6.2.22 GRID_Y_ORIG

```
const unsigned long Constants::GRID_Y_ORIG [static]
```

6.6.2.23 PERSONS_FILE_NAME

```
const char* Constants::PERSONS_FILE_NAME [static]
```

The name of the file where the exact positions of the persons are saved during simulation. They are needed for later analysis.

6.6.2.24 PHONE_CONNECTION_THRESHOLD

```
const double Constants::PHONE_CONNECTION_THRESHOLD [static]
```

This value is interpreted according to the connection type:

- if the connection uses power it is the minimum value of the signal power received by a phone not considered as noise. Below this value the signal is unusable and the connection between a mobile phone and an antenna is not possible.
- if the connection uses signal quality it is the minimum value of the signal quality received by a phone not considered as noise. Below this value the signal is unusable and the connection between a mobile phone and an antenna is not possible.
- if the connection uses signal strength it is the minimum value of the signal strength received by a phone not considered as noise. Below this value the signal is unusable and the connection between a mobile phone and an antenna is not possible.

6.6.2.25 PHONE_POWER_THRESHOLD

```
const double Constants::PHONE_POWER_THRESHOLD [static]
```

If the signal received by a mobile device has a power below this level, the signal is considered only noise and unusable.

6.6.2.26 PHONE_QUALITY_THRESHOLD

```
const double Constants::PHONE_QUALITY_THRESHOLD [static]
```

If the signal received by a mobile device has a quality below this level, the signal is considered only noise and unusable.

6.6.2.27 PHONE_STRENGTH_THRESHOLD

```
const double Constants::PHONE_STRENGTH_THRESHOLD [static]
```

If the signal received by a mobile device has a quality below this level, the signal is considered only noise and unusable.

6.6.2.28 PRIOR_PROBABILITY

```
const PriorType Constants::PRIOR_PROBABILITY [static]
```

Indicates how the prior probability is computed: uniform, register, network

6.6.2.29 PROB_FILE_NAME_PREFIX

```
const char* Constants::PROB_FILE_NAME_PREFIX [static]
```

The name of the file where the probabilities of mobile phones locations are saved

6.6.2.30 RANDOM_SEED

```
const int Constants::RANDOM_SEED [static]
```

6.6.2.31 sep

```
const char Constants::sep [static]
```

The separator used when information is saved in output files

6.6.2.32 SIM_DEFAULT_MNO_NAME

```
const char* Constants::SIM_DEFAULT_MNO_NAME [static]
```

6.6.2.33 SIM_END_TIME

```
const unsigned long Constants::SIM_END_TIME [static]
```

Default ending time of a simulation

6.6.2.34 SIM_INCREMENT_TIME

```
const unsigned long Constants::SIM_INCREMENT_TIME [static]
```

Default time increment for a simulation

6.6.2.35 SIM_INTERVAL_BETWEEN_STAYS

```
const unsigned long Constants::SIM_INTERVAL_BETWEEN_STAYS [static]
```

6.6.2.36 SIM_NO_ANTENNAS

```
const unsigned long Constants::SIM_NO_ANTENNAS [static]
```

The number of antenna used for a simulation

6.6.2.37 SIM_NO_MOBILE_PHONES

```
const unsigned long Constants::SIM_NO_MOBILE_PHONES [static]
```

The number of the mobile devices used for a simulation

6.6.2.38 SIM_NO_PERSONS

```
const unsigned long Constants::SIM_NO_PERSONS [static]
```

The number of persons used for a simulation

6.6.2.39 SIM_NUM_MNO

```
const unsigned int Constants::SIM_NUM_MNO [static]
```

6.6.2.40 SIM_PROB_MOBILE_PHONE

```
const double Constants::SIM_PROB_MOBILE_PHONE [static]
```

6.6.2.41 SIM_PROB_SECOND_MOBILE_PHONE

```
const double Constants::SIM_PROB_SECOND_MOBILE_PHONE [static]
```

6.6.2.42 SIM_START_TIME

```
const unsigned long Constants::SIM_START_TIME [static]
```

Default starting time of a simulation

6.6.2.43 SIM_STAY_TIME

```
const unsigned long Constants::SIM_STAY_TIME [static]
```

6.6.2.44 SIM_TREND_ANGLE_1

```
const double Constants::SIM_TREND_ANGLE_1 [static]
```

6.6.2.45 SIM_TREND_ANGLE_2

```
const double Constants::SIM_TREND_ANGLE_2 [static]
```

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

- [include/Constants.h](#)

6.7 CSVParser Class Reference

```
#include <CSVparser.hpp>
```


Public Member Functions

- [CSVParser](#) (const string &data, const [DataType](#) &type=[eFILE](#), char sep=',', bool hasHeader=true)
- [~CSVParser](#) (void)
- [Row](#) & [getRow](#) (unsigned int row) const
- unsigned int [rowCount](#) (void) const
- unsigned int [columnCount](#) (void) const
- vector< string > [getHeader](#) (void) const
- const string [getHeaderElement](#) (unsigned int pos) const
- const string & [getFileName](#) (void) const
- bool [deleteRow](#) (unsigned int row)
- bool [addRow](#) (unsigned int pos, const vector< string > &r)
- void [sync](#) (void) const
- [Row](#) & [operator\[\]](#) (unsigned int row) const

Protected Member Functions

- void [parseHeader](#) (void)
- void [parseContent](#) (void)

Private Attributes

- string [_file](#)
- const [DataType](#) [_type](#)
- const char [_sep](#)
- vector< string > [_originalFile](#)
- vector< string > [_header](#)
- vector< [Row](#) * > [_content](#)
- bool [m_header](#)

6.7.1 Detailed Description

This class is used to read and parse a csv file or to write some values as a csv file.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 CSVParser()

```
CSVParser::CSVParser (
    const string & data,
    const DataType & type = eFILE,
    char sep = ',',
    bool hasHeader = true )
```

Constructor of the class. It need the name of the csv file, the file type, the separator and a boolean that indicates if the file has header or not.

Parameters

<i>data</i>	the name of the file
<i>type</i>	the file type: could be eFILE for normal text files or ePURE if the input is a string
<i>sep</i>	the separator of the individual values in a line of the csv file
<i>hasHeader</i>	true means that the csv file has a header line, false that it doesn't have a header

6.7.2.2 ~CSVParser()

```
CSVParser::~~CSVParser (
    void )
```

Destructor

6.7.3 Member Function Documentation**6.7.3.1 addRow()**

```
bool CSVParser::addRow (
    unsigned int pos,
    const vector< string > & r )
```

Inserts a [Row](#) object at a given position

Parameters

<i>pos</i>	the position where we want to insert the Row object
<i>r</i>	a vector containing the values in the Row .

Returns

true if the insertion is successful, false otherwise (i.e. the *pos* parameter is outside the limits of the container that stores the Rows of the csv file).

6.7.3.2 columnCount()

```
unsigned int CSVParser::columnCount (
    void ) const
```

Returns the number of the columns of the csv file.

Returns

the number of the columns of the csv file.

6.7.3.3 deleteRow()

```
bool CSVParser::deleteRow (
    unsigned int row )
```

Removes a row specified by its number

Parameters

<i>row</i>	the number of the row to be deleted
------------	-------------------------------------

Returns

true if the removal succeeded, false otherwise

6.7.3.4 getFileName()

```
const string& CSVParser::getFileName (
    void ) const
```

Returns the name of the csv file

Returns

the name of the csv file

6.7.3.5 getHeader()

```
vector<string> CSVParser::getHeader (
    void ) const
```

Returns a vector containing the names of the columns as they are specified in the header line of the csv file.

Returns

a vector containing the names of the columns as they are specified in the header line of the csv file.

6.7.3.6 getHeaderElement()

```
const string CSVParser::getHeaderElement (
    unsigned int pos ) const
```

Returns the name of a specific column given by its position in the header line

Parameters

<i>pos</i>	the number of the column
------------	--------------------------

Returns

the name of a specific column given by its position in the header line

6.7.3.7 `getRow()`

```
Row& CSVParser::getRow (
    unsigned int row ) const
```

Returns a [Row](#) object specified by its number in the file

Parameters

<i>row</i>	the number of the line that was used to build the Row object
------------	--

Returns

a [Row](#) object specified by its number in the file

6.7.3.8 `operator[]()`

```
Row& CSVParser::operator[] (
    unsigned int row ) const
```

Overloaded operator

Parameters

<i>row</i>	the number of the row to be retrieved
------------	---------------------------------------

Returns

the [Row](#) object at the position specified by row

6.7.3.9 `parseContent()`

```
void CSVParser::parseContent (
    void ) [protected]
```

6.7.3.10 parseHeader()

```
void CSVParser::parseHeader (
    void ) [protected]
```

6.7.3.11 rowCount()

```
unsigned int CSVParser::rowCount (
    void ) const
```

Returns the number of lines in the csv file without counting the header line, if it exists

Returns

the number of lines in the csv file without counting the header line, if it exists

6.7.3.12 sync()

```
void CSVParser::sync (
    void ) const
```

Flushes the content to a file on disk and then closes the file.

6.7.4 Member Data Documentation

6.7.4.1 _content

```
vector<Row *> CSVParser::_content [private]
```

6.7.4.2 _file

```
string CSVParser::_file [private]
```

6.7.4.3 _header

```
vector<string> CSVParser::_header [private]
```

6.7.4.4 `_originalFile`

```
vector<string> CSVParser::_originalFile [private]
```

6.7.4.5 `_sep`

```
const char CSVParser::_sep [private]
```

6.7.4.6 `_type`

```
const DataType CSVParser::_type [private]
```

6.7.4.7 `m_header`

```
bool CSVParser::m_header [private]
```

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

- [include/CSVparser.hpp](#)

6.8 EMField Class Reference

```
#include <EMField.h>
```

Collaboration diagram for EMField:



Public Member Functions

- virtual `~EMField ()`
- void `addAntenna (Antenna *a)`
- pair< `Antenna *`, double > `computeMaxPower` (const Point *p, const unsigned long mnold)
- pair< `Antenna *`, double > `computeMaxQuality` (const Point *p, const unsigned long mnold)
- pair< `Antenna *`, double > `computeMaxStrength` (const Point *p, const unsigned long mnold)
- vector< pair< `Antenna *`, double > > `getInRangeAntennas` (const Point *p, const double threshold, const `HoldableAgent::CONNECTION_TYPE` connType, unsigned long mnold)
- bool `isAntennaInRange` (const Point *p, `Antenna *a`, const double threshold, const `HoldableAgent::CONNECTION_TYPE` connType)
- double `connectionLikelihood` (`Antenna *a`, const Point *p)
- vector< double > `sumSignalQuality` (const `Grid *grid`, const unsigned long mnold)
- double `connectionLikelihoodGrid` (`Antenna *a`, const `Grid *g`, unsigned long tileIndex)
- const double * `getAntennaMin3DbArray` () const
- double * `getSd` () const

Static Public Member Functions

- static `EMField * instance` ()

Private Member Functions

- `EMField ()`
- `EMField` (const `EMField &`)
- `EMField & operator=` (const `EMField &`)

Private Attributes

- vector< `Antenna *` > `m_antennas`
- map< const unsigned long, vector< double > > `m_sumQuality`
- double * `m_antennaMin3DbArray`
- double * `m_sd`

Static Private Attributes

- static `EMField * m_instance`

6.8.1 Detailed Description

This utility singleton class is used to compute different measures of the electromagnetic field radiated by an antenna (power, signal strength etc) and it also provides methods needed to decide to which antenna a mobile device connects.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 `~EMField()`

```
virtual EMField::~~EMField ( ) [virtual]
```

Default destructor.

6.8.2.2 `EMField()` [1/2]

```
EMField::EMField ( ) [private]
```

6.8.2.3 `EMField()` [2/2]

```
EMField::EMField (
    const EMField & ) [private]
```

6.8.3 Member Function Documentation

6.8.3.1 `addAntenna()`

```
void EMField::addAntenna (
    Antenna * a )
```

Add a pointer to an [Antenna](#) object to an internal collection needed for computations. Although these pointers are kept in an `AgentCollection` object they are also added to a local vector in this class for performance reasons.

Parameters

<i>a</i>	a pointer to the Antenna object
----------	---

6.8.3.2 `computeMaxPower()`

```
pair<Antenna*, double> EMField::computeMaxPower (
    const Point * p,
    const unsigned long mnoId )
```

Returns a pair made of a pointer to an [Antenna](#) object and its power with the property that in the location specified by parameter *p*, the [Antenna](#) returned by this method provides the highest power (the power of the field is considered to decrease according a power-law).

Parameters

p	the location where we want to find which Antenna provides the highest power the of electromagnetic field.
$mno \leftrightarrow Id$	the id of the MNO for which we compute the power. Only antennas belonging to this MNO will be considered during computations.

Returns

a pair<Antenna*, double> containing a pointer to the [Antenna](#) object that provides the highest power of the field in the location specified by p.

6.8.3.3 computeMaxQuality()

```
pair<Antenna*, double> EMField::computeMaxQuality (
    const Point * p,
    const unsigned long mnoId )
```

Returns a pair made of a pointer to an [Antenna](#) object and its signal quality with the property that in the location specified by p, the [Antenna](#) returned by this method provides signal with the highest quality. The signal quality in this pair is the computed in location given by p.

Parameters

p	indicates the location where we want to find which Antenna provides the highest quality of the signal.
$mno \leftrightarrow Id$	the id of the MNO for which we compute the signal quality. Only antennas belonging to this MNO will be considered during computations.

Returns

a pair<Antenna*, double> containing a pointer to the [Antenna](#) object that provides a signal with the highest quality in location given by p.

6.8.3.4 computeMaxStrength()

```
pair<Antenna*, double> EMField::computeMaxStrength (
    const Point * p,
    const unsigned long mnoId )
```

Returns a pair made of a pointer to an [Antenna](#) object and its signal strength with the property that in the location specified by p, the [Antenna](#) returned by this method provides signal with the highest strength. The signal strength in this pair is the computed in location given by p.

Parameters

p	indicates the location where we want to find which Antenna provides the highest strength of the signal.
$mno \leftrightarrow Id$	the id of the MNO for which we compute the signal strength. Only antennas belonging to this MNO will be considered during computations.

Returns

a pair<Antenna*, double> containing a pointer to the [Antenna](#) object that provides a signal with the highest strength in location given by p.

6.8.3.5 connectionLikelihood()

```
double EMField::connectionLikelihood (
    Antenna * a,
    const Point * p )
```

Computes the connection likelihood for [Antenna](#) indicated by a in a certain location given by p. The connection likelihood is computed dividing the signal quality provided by [Antenna](#) indicated through p by the sum of the signal quality provided by all antennas of an MNO.

Parameters

<i>a</i>	a pointer to an Antenna object.
<i>p</i>	a location in space.

Returns

the connection likelihood for [Antenna](#) a in location p.

6.8.3.6 connectionLikelihoodGrid()

```
double EMField::connectionLikelihoodGrid (
    Antenna * a,
    const Grid * g,
    unsigned long tileIndex )
```

Computes the connection likelihood for [Antenna](#) indicated by a in the center of the tile indicated by tileIndex

Parameters

<i>a</i>	a pointer to an Antenna object.
<i>g</i>	a pointer to the reference Grid object
<i>tileIndex</i>	the index of the tile where we want to compute the connection likelihood.

Returns

the connection likelihood for [Antenna](#) a in the center of the tile with the index tileIndex.

6.8.3.7 getAntennaMin3DbArray()

```
const double* EMField::getAntennaMin3DbArray ( ) const
```

6.8.3.8 getInRangeAntennas()

```
vector<pair<Antenna*, double> > EMField::getInRangeAntennas (
    const Point * p,
    const double threshold,
    const HoldableAgent::CONNECTION_TYPE connType,
    unsigned long mnoId )
```

Returns a vector of pairs made up of a pointer to an [Antenna](#) object and its power, signal quality or signal strength. All the antennas in this vector provides a signal with a power or signal quality greater than the threshold provided as threshold, i.e. this vector contains all antennas that have in their coverage area the location given by point p.

Parameters

<i>p</i>	the location where we want to have the list with the all antennas that covers it.
<i>threshold</i>	the lowest limit of the power or signal quality below which the signal is considered to be only noise, i.e. it defines the limit of the coverage area.
<i>connType</i>	indicates the mechanism used to set up a connection between an antenna and a mobile phone
<i>mnoId</i>	the id of the MNO for which we build the resulting vector. Only antennas belonging to this MNO will be considered during computations.

Returns

a vector of pairs made up of a pointer to an [Antenna](#) object and its power, signal quality or signal strength, according to the value of the connType. All the antennas in this vector provides a signal with a power, signal quality or signal strength greater than the threshold.

6.8.3.9 getSd()

```
double* EMField::getSd ( ) const
```

6.8.3.10 instance()

```
static EMField* EMField::instance ( ) [inline], [static]
```

Returns an instance of this class. This class is a singleton.

Returns

an instance of this class.

6.8.3.11 isAntennaInRange()

```
bool EMField::isAntennaInRange (
    const Point * p,
    Antenna * a,
    const double threshold,
    const HoldableAgent::CONNECTION_TYPE connType )
```

Checks if *p* is in the coverage area of *Antenna* pointed out by *a*. The coverage area is considered the area where the signal quality or the power of the field is greater than the value of *threshold*.

Parameters

<i>p</i>	the location that we want to check the power or the quality of the signal
<i>a</i>	pointer to an <i>Antenna</i> object for which we want to check if it covers the point <i>p</i> .
<i>threshold</i>	the lower limit of the power or signal quality below which the signal is considered only noise.
<i>connType</i>	indicates the mechanism used to set up a connection between an antenna and a mobile phone.

Returns

true is the *Antenna* object provide enough power or signal quality in the location given as *p*.

6.8.3.12 operator=()

```
EMField& EMField::operator= (
    const EMField & ) [private]
```

6.8.3.13 sumSignalQuality()

```
vector<double> EMField::sumSignalQuality (
    const Grid * grid,
    const unsigned long mnoID )
```

Computes the sum of the signal quality given by all antennas belonging to an MNO for all tiles in the reference grid. The signal quality is computed in the center of each tile.

Parameters

<i>grid</i>	the grid of tiles where this method computes the sum of the signal quality. This grid is set at the beginning of the simulation and it overlaps the <i>Map</i> .
<i>mnoID</i>	the id of the MNO for which we want to compute this sum.

Returns

a vector containing the sum of the signal quality given by all antennas of an MNO, for all tiles in the reference grid. An element of the vector corresponds to a tile in the grid. The tiles are linearized in a row-major order starting with the bottom-left corner.

6.8.4 Member Data Documentation**6.8.4.1 m_antennaMin3DbArray**

```
double* EMField::m_antennaMin3DbArray [private]
```

6.8.4.2 m_antennas

```
vector<Antenna*> EMField::m_antennas [private]
```

6.8.4.3 m_instance

```
EMField* EMField::m_instance [static], [private]
```

6.8.4.4 m_sd

```
double* EMField::m_sd [private]
```

6.8.4.5 m_sumQuality

```
map<const unsigned long, vector<double> > EMField::m_sumQuality [private]
```

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

- include/[EMField.h](#)

6.9 Grid Class Reference

```
#include <Grid.h>
```

Public Member Functions

- [Grid](#) (double xOrig, double yOrig, double xTiledim, double yTiledim, unsigned long noTilesX, unsigned long noTilesY)
- virtual [~Grid](#) ()
- unsigned long [getNoTilesX](#) () const
- unsigned long [getNoTilesY](#) () const
- double [getXTileDim](#) () const
- double [getYTileDim](#) () const
- double [getXOrigin](#) () const
- double [getYOrigin](#) () const
- string [toString](#) () const
- unsigned long [getTileIndexX](#) (const Point *p) const
- unsigned long [getTileIndexY](#) (const Point *p) const
- const unsigned long [getNoTiles](#) () const
- vector< double > [computeProbability](#) (unsigned long t, [MobilePhone](#) *m, vector< [AntennaInfo](#) > &data, pair< [um_iterator](#), [um_iterator](#) > it, [PriorType](#) prior) const
- Coordinate [getTileCenter](#) (unsigned long tileIndex) const
- unsigned long [getTileNo](#) (const Point *p) const
- unsigned long [getTileIndexX](#) (double x) const
- unsigned long [getTileIndexY](#) (double y) const
- unsigned long [getTileNo](#) (double x, double y) const
- void [dumpGrid](#) (const string &gridFileName) const
- Coordinate * [getTileCenters](#) () const

Private Member Functions

- Coordinate * [computeTileCenters](#) ()
- vector< double > [useNetworkPrior](#) (unsigned long t, bool connected, vector< [AntennaInfo](#) >::iterator ai, pair< [um_iterator](#), [um_iterator](#) > antennas_iterator) const
- vector< double > [useUniformPrior](#) (unsigned long t, bool connected, vector< [AntennaInfo](#) >::iterator ai, pair< [um_iterator](#), [um_iterator](#) > antennas_iterator) const

Private Attributes

- double [m_xOrigin](#)
- double [m_yOrigin](#)
- double [m_xTileDim](#)
- double [m_yTileDim](#)
- unsigned long [m_noTilesX](#)
- unsigned long [m_noTilesY](#)
- Coordinate * [m_tileCenters](#)

6.9.1 Detailed Description

This class implements a grid of rectangular tiles overlapped on the map of the simulation. This grid is used to compute the "observed" location of a mobile phone. This means that we compute the probability of a mobile device to be in a specific tile of the grid using the data recorded by each antenna during the simulation. A finer grid will give a more accurate location but the computational cost increase when the size of the tiles decrease. The tiles of the grid are indexed starting with 0 for the tile in the bottom left corner of the grid in a row-major ordering. The last tile, with the biggest index, is the tile in the upper-right corner of the grid.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Grid()

```
Grid::Grid (
    double xOrig,
    double yOrig,
    double xTiledim,
    double yTiledim,
    unsigned long noTilesX,
    unsigned long noTilesY )
```

Constructor of the class. Build a [Grid](#) object with the specified parameters.

Parameters

<i>xOrig</i>	the x coordinate of the origin of the grid (i.e. the x coordinate of the bottom left corner of the grid).
<i>yOrig</i>	the y coordinate of the origin of the grid (i.e. the y coordinate of the bottom left corner of the grid).
<i>xTiledim</i>	the dimension of a tile on X axis.
<i>yTiledim</i>	the dimension of a tile on Y axis.
<i>noTilesX</i>	the number of tiles on X axis.
<i>noTilesY</i>	the number of tiles on Y axis.

6.9.2.2 ~Grid()

```
virtual Grid::~~Grid ( ) [virtual]
```

Default destructor.

6.9.3 Member Function Documentation

6.9.3.1 computeProbability()

```
vector<double> Grid::computeProbability (
    unsigned long t,
    MobilePhone * m,
    vector< AntennaInfo > & data,
    pair< um\_iterator, um\_iterator > it,
    PriorType prior ) const
```

Computes the posterior probability of a mobile device to be in a tile of the [Grid](#) according to the method described in the paper "Deriving geographic location of mobile devices from network data" by Martijn Tennekes, Yvonne A.P.M. Gootzen, Shan H. Shah.

Parameters

<i>t</i>	the time instant when the posterior localization probability is computed.
<i>m</i>	a pointer to a MobilePhone object for which the posterior localization probability is computed.
<i>data</i>	a vector of AntennaInfo objects generated and recorded by each antenna during the simulation. It contains the events recorder by each antenna during the simulation.
<i>it</i>	an iterator to access all objects of type Antenna from the AgentsCollection container.
<i>prior</i>	is used to set the method of computing the prior probabilities. It could take 3 values: PriorType::UNIFORM , PriorType::NETWORK or PriorType::REGISTER . Currently only UNIFORM and NETWORK methods are implemented.

Returns

a vector with the posterior probability of the mobile phone given by *m* to be localized in a tile. The index of a value in this vector indicates the corresponding tile index. The size of this vector is equal to the total number of tiles in the [Grid](#).

6.9.3.2 computeTileCenters()

```
Coordinate* Grid::computeTileCenters ( ) [private]
```

6.9.3.3 dumpGrid()

```
void Grid::dumpGrid (
    const string & gridFileName ) const
```

Writes the grid description in a .csv file for later processing.

Parameters

<i>gridFileName</i>	the name of the output file.
---------------------	------------------------------

6.9.3.4 getNoTiles()

```
const unsigned long Grid::getNoTiles ( ) const
```

Computes the total number of tiles in the grid.

Returns

the total number of tiles in the grid.

6.9.3.5 getNoTilesX()

```
unsigned long Grid::getNoTilesX ( ) const
```

Returns

the number of tiles of the grid on X axis direction.

6.9.3.6 getNoTilesY()

```
unsigned long Grid::getNoTilesY ( ) const
```

Returns

the number of tiles of the grid on Y axis direction.

6.9.3.7 getTileCenter()

```
Coordinate Grid::getTileCenter (
    unsigned long tileIndex ) const
```

Computes the coordinates of the tile center given by its index in the grid.

Parameters

<i>tileIndex</i>	the tile index.
------------------	-----------------

Returns

the coordinates of the center of the tile.

6.9.3.8 getTileCenters()

```
Coordinate* Grid::getTileCenters ( ) const
```

Returns a vector containing the coordinates of the tile centers.

Returns

a vector containing the coordinates of the tile centers.

6.9.3.9 `getTileIndexX()` [1/2]

```
unsigned long Grid::getTileIndexX (  
    const Point * p ) const
```

Returns the tile index on X axis that contains a given point in space, specified by p.

Parameters

<i>p</i>	a pointer to the point for which we need the tile index.
----------	--

Returns

the tile index on X axis that contains the point specified by p, i.e. a number between 0 and [getNoTilesX\(\)](#) - 1.

6.9.3.10 `getTileIndexX()` [2/2]

```
unsigned long Grid::getTileIndexX (  
    double x ) const
```

6.9.3.11 `getTileIndexY()` [1/2]

```
unsigned long Grid::getTileIndexY (  
    const Point * p ) const
```

Returns the tile index on Y axis that contains a given point in space, specified by p.

Parameters

<i>p</i>	the point in space for which we need the tile index.
----------	--

Returns

the tile index on Y axis that contains the point specified by p, i.e. a number between 0 and [getNoTilesY\(\)](#) - 1.

6.9.3.12 `getTileIndexY()` [2/2]

```
unsigned long Grid::getTileIndexY (  
    double y ) const
```

6.9.3.13 `getTileNo()` [1/2]

```
unsigned long Grid::getTileNo (
    const Point * p ) const
```

Computes the tile index of the tile that contains the Point indicated by p.

Parameters

<i>p</i>	a pointer to a Point object.
----------	------------------------------

Returns

the tile index of the tile that contains the Point indicated by p.

6.9.3.14 `getTileNo()` [2/2]

```
unsigned long Grid::getTileNo (
    double x,
    double y ) const
```

Computes the tile index of the tile that contains a point with coordinates indicated by x and y.

Parameters

<i>x</i>	x coordinate of a location.
<i>y</i>	y coordinate of a location.

Returns

the tile index of the tile that contains a point with coordinates indicated by x and y.

6.9.3.15 `getXOrigin()`

```
double Grid::getXOrigin ( ) const
```

Returns

the x coordinate of the origin of the grid (i.e. the x coordinate of the bottom left corner of the grid).

6.9.3.16 getXTileDim()

```
double Grid::getXTileDim ( ) const
```

Returns

the dimension of a tile on X axis direction.

6.9.3.17 getYOrigin()

```
double Grid::getYOrigin ( ) const
```

Returns

the y coordinate of the origin of the grid (i.e. the y coordinate of the bottom left corner of the grid).

6.9.3.18 getYTileDim()

```
double Grid::getYTileDim ( ) const
```

Returns

the dimension of a tile on Y axis direction.

6.9.3.19 toString()

```
string Grid::toString ( ) const
```

Returns

a string representation of an object of type [Grid](#). This is useful to write a textual description of the grid in a file for later processing.

6.9.3.20 useNetworkPrior()

```
vector<double> Grid::useNetworkPrior (
    unsigned long t,
    bool connected,
    vector< AntennaInfo >::iterator ai,
    pair< um\_iterator, um\_iterator > antennas_iterator ) const [private]
```

6.9.3.21 useUniformPrior()

```
vector<double> Grid::useUniformPrior (
    unsigned long t,
    bool connected,
    vector< AntennaInfo >::iterator ai,
    pair< um_iterator, um_iterator > antennas_iterator ) const [private]
```

6.9.4 Member Data Documentation

6.9.4.1 m_noTilesX

```
unsigned long Grid::m_noTilesX [private]
```

6.9.4.2 m_noTilesY

```
unsigned long Grid::m_noTilesY [private]
```

6.9.4.3 m_tileCenters

```
Coordinate* Grid::m_tileCenters [private]
```

6.9.4.4 m_xOrigin

```
double Grid::m_xOrigin [private]
```

6.9.4.5 m_xTileDim

```
double Grid::m_xTileDim [private]
```

6.9.4.6 m_yOrigin

```
double Grid::m_yOrigin [private]
```

6.9.4.7 m_yTileDim

```
double Grid::m_yTileDim [private]
```

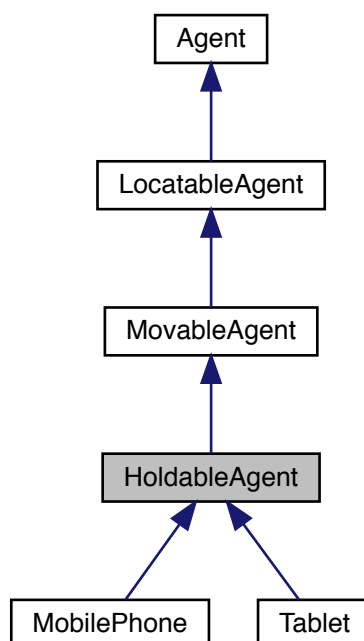
The documentation for this class was generated from the following file:

- [include/Grid.h](#)

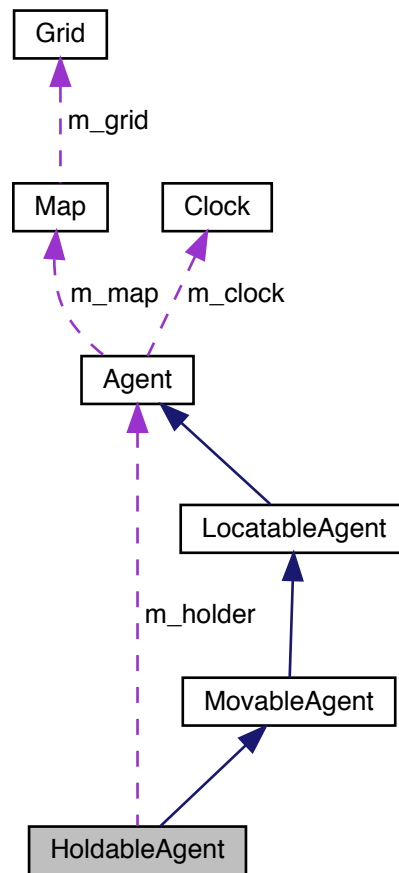
6.10 HoldableAgent Class Reference

```
#include <HoldableAgent.h>
```

Inheritance diagram for HoldableAgent:



Collaboration diagram for HoldableAgent:



Public Types

- enum `CONNECTION_TYPE` { `USING_POWER`, `USING_SIGNAL_QUALITY`, `USING_SIGNAL_STRENGTH`, `UNKNOWN` }

Public Member Functions

- `HoldableAgent` (const `Map` *m, const unsigned long id, Point *initPosition, `Agent` *holder, const `Clock` *clock)
- `HoldableAgent` (const `HoldableAgent` &h)
- virtual `~HoldableAgent` ()
- `Agent` * `getHolder` () const
- void `setHolder` (`Agent` *holder)
- const string `getName` () const override
- const string `toString` () const override
- virtual bool `tryConnect` ()=0
- virtual bool `isConnected` () const
- void `setLocation` (Point *location) override
- vector< `Antenna` * > `getAntennas` () const

Private Attributes

- `Agent * m_holder`
- `vector< Antenna * > m_antennas`

6.10.1 Detailed Description

This is the superclass for all agents that represent a device that can be held by a person.

6.10.2 Member Enumeration Documentation

6.10.2.1 CONNECTION_TYPE

```
enum HoldableAgent::CONNECTION_TYPE
```

an enumeration of the modes used by the device to connect to an antenna: USING_POWER - connects to the antenna that provides the maximum power of the field in the location of the device. USING_SIGNAL_QUALITY - connects to the antenna that provides the maximum value of the signal quality in the location of the device. USING_SIGNAL_STRENGTH - connects to the antenna that provides the maximum value of the signal strength in the location of the device. UNKNOWN - this should be an error.

Enumerator

USING_POWER	
USING_SIGNAL_QUALITY	
USING_SIGNAL_STRENGTH	
UNKNOWN	

6.10.3 Constructor & Destructor Documentation

6.10.3.1 HoldableAgent() [1/2]

```
HoldableAgent::HoldableAgent (
    const Map * m,
    const unsigned long id,
    Point * initPosition,
    Agent * holder,
    const Clock * clock ) [explicit]
```

Constructor of the class. It builds an `HoldableAgent` object with the parameters provided by user.

Parameters

<i>m</i>	a pointer to a Map object used for this simulation.
<i>id</i>	the id of the object.
<i>initPosition</i>	the initial location on the map of the object.
<i>holder</i>	a pointer to an Agent that owns this device.
<i>clock</i>	a pointer to a Clock object used by this simulation.

6.10.3.2 HoldableAgent() [2/2]

```
HoldableAgent::HoldableAgent (
    const HoldableAgent & h )
```

Copy constructor.

Parameters

<i>h</i>	another object of the same type.
----------	----------------------------------

6.10.3.3 ~HoldableAgent()

```
virtual HoldableAgent::~~HoldableAgent ( ) [virtual]
```

Destructor

6.10.4 Member Function Documentation

6.10.4.1 getAntennas()

```
vector<Antenna> HoldableAgent::getAntennas ( ) const
```

Returns

a vector with pointers to antennas where this device is connected

6.10.4.2 getHolder()

```
Agent* HoldableAgent::getHolder ( ) const
```

Returns a pointer to an [Agent](#) object that owns this device

Returns

a pointer to an [Agent](#) object that owns this device

6.10.4.3 getName()

```
const string HoldableAgent::getName ( ) const [override], [virtual]
```

Returns the name of the class

Returns

the name of the class

Implements [Agent](#).

Reimplemented in [MobilePhone](#), and [Tablet](#).

6.10.4.4 isConnected()

```
virtual bool HoldableAgent::isConnected ( ) const [virtual]
```

check if this device is connected to an antenna

Returns

true if the device is conneted to an antenna, false otherwise.

6.10.4.5 setHolder()

```
void HoldableAgent::setHolder (
    Agent * holder )
```

Sets the owner of this device

Parameters

<i>holder</i>	a pointer to the owner of this device
---------------	---------------------------------------

6.10.4.6 setLocation()

```
void HoldableAgent::setLocation (
    Point * location ) [override], [virtual]
```

Sets the location of this device. After a new location is set, the device tries to connect to an antenna, i.e. the [tryConnect\(\)](#) method is called. Since this is an abstract class and the connection type is unknown it is the responsibility of subclasses to override this method and provide the correct mode of connection.

Parameters

<i>location</i>	a point on the Map of the simulation
-----------------	--

Reimplemented from [LocatableAgent](#).

6.10.4.7 toString()

```
const string HoldableAgent::toString ( ) const [override], [virtual]
```

Returns a string representation of this class, useful to print it to the console or in a file.

Returns

a string representation of this class, useful to print it to the console or in a file.

Implements [Agent](#).

Reimplemented in [MobilePhone](#), and [Tablet](#).

6.10.4.8 tryConnect()

```
virtual bool HoldableAgent::tryConnect ( ) [pure virtual]
```

Called when a device wants to connect to an antenna

Returns

true if the connection succeeds, false otherwise.

Implemented in [MobilePhone](#), and [Tablet](#).

6.10.5 Member Data Documentation

6.10.5.1 m_antennas

```
vector<Antenna*> HoldableAgent::m_antennas [private]
```

6.10.5.2 m_holder

```
Agent* HoldableAgent::m_holder [private]
```

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

- include/[HoldableAgent.h](#)

6.11 IDGenerator Class Reference

```
#include <IDGenerator.h>
```

Collaboration diagram for IDGenerator:



Public Member Functions

- unsigned long [next](#) ()

Static Public Member Functions

- static [IDGenerator](#) * [instance](#) ()

Private Member Functions

- [IDGenerator](#) ()

Private Attributes

- unsigned long [m_id](#)

Static Private Attributes

- static [IDGenerator](#) * [m_instance](#)

6.11.1 Detailed Description

This singleton class is used to generate unique identifiers for all agents in the simulation. The ids are unsigned long integers.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 IDGenerator()

```
IDGenerator::IDGenerator ( ) [inline], [private]
```

6.11.3 Member Function Documentation

6.11.3.1 instance()

```
static IDGenerator* IDGenerator::instance ( ) [inline], [static]
```

Returns an instance of this class.

Returns

an instance of this class.

Here is the call graph for this function:



6.11.3.2 next()

```
unsigned long IDGenerator::next ( ) [inline]
```

Generates the next unique identifier.

Returns

a unique identifier.

6.11.4 Member Data Documentation

6.11.4.1 m_id

```
unsigned long IDGenerator::m_id [private]
```

6.11.4.2 m_instance

```
IDGenerator* IDGenerator::m_instance [static], [private]
```

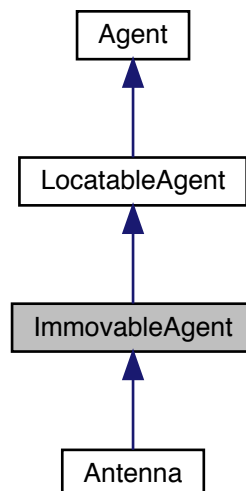
The documentation for this class was generated from the following file:

- include/[IDGenerator.h](#)

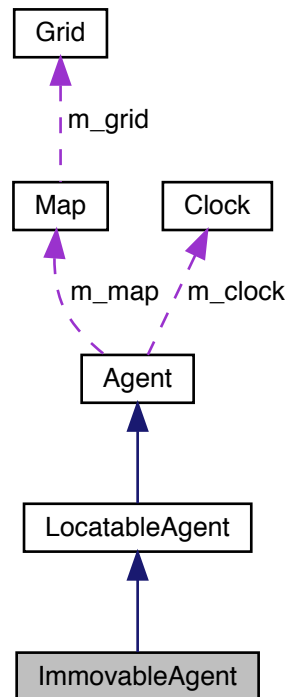
6.12 ImmovableAgent Class Reference

```
#include <ImmovableAgent.h>
```

Inheritance diagram for ImmovableAgent:



Collaboration diagram for ImmovableAgent:



Public Member Functions

- [ImmovableAgent](#) (const [Map](#) *m, const unsigned long id, Point *initialPosition, const [Clock](#) *clock)
- virtual [~ImmovableAgent](#) ()
- const string [toString](#) () const override
- const string [getName](#) () const override

6.12.1 Detailed Description

This is a class that represents an agent that can have a location on map but it cannot move. The only subclass of it is [Antenna](#).

6.12.2 Constructor & Destructor Documentation

6.12.2.1 `ImmovableAgent()`

```
ImmovableAgent::ImmovableAgent (
    const Map * m,
    const unsigned long id,
    Point * initialPosition,
    const Clock * clock ) [explicit]
```

Constructor of the class. Build an `ImmovableAgent` object with the parameters provided by the user.

Parameters

<i>m</i>	a pointer to a Map object used in this simulation.
<i>id</i>	the id of this object.
<i>initialPosition</i>	the initial location on map.
<i>clock</i>	a pointer to a Clock object used in this simulation.

6.12.2.2 ~ImmovableAgent()

```
virtual ImmovableAgent::~~ImmovableAgent ( ) [virtual]
```

Default destructor.

6.12.3 Member Function Documentation**6.12.3.1 getName()**

```
const string ImmovableAgent::getName ( ) const [override], [virtual]
```

Returns the name of this class.

Returns

the name of this class.

Implements [Agent](#).

6.12.3.2 toString()

```
const string ImmovableAgent::toString ( ) const [override], [virtual]
```

Builds a string representation of this class.

Returns

a string representation of this class. It is used to write details of the [ImmovableAgent](#) objects in a file or on console.

Implements [Agent](#).

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

- [include/ImmovableAgent.h](#)

6.13 InputParser Class Reference

```
#include <InputParser.h>
```

Public Member Functions

- [InputParser](#) (int &argc, char **argv)
- const string & [getCmdOption](#) (const string &option) const
- bool [cmdOptionExists](#) (const string &option) const

Private Attributes

- vector< string > [tokens](#)

6.13.1 Detailed Description

Utility class used to parse the command line and extract the parameters and their values. An option is passed with a "-" sign in front of it. Example: \$simulator -s simulation.xml -p persons Here -s and -p are options and simulation.xml and persons.xml are their corresponding values.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 InputParser()

```
InputParser::InputParser (
    int & argc,
    char ** argv )
```

Constructor of the class.

Parameters

<i>argc</i>	the number of the arguments from the command line.
<i>argv</i>	an array with the parameters passed in the command line.

6.13.3 Member Function Documentation

6.13.3.1 cmdOptionExists()

```
bool InputParser::cmdOptionExists (
    const string & option ) const
```

Checks if an option was passed as a command line parameter.

Parameters

<i>option</i>	the option that we are checking.
---------------	----------------------------------

Returns

true if the option is present in the command line, false otherwise.

6.13.3.2 getCmdOption()

```
const string& InputParser::getCmdOption (
    const string & option ) const
```

Returns the value of an option passed as a command line parameter.

Parameters

<i>option</i>	an option from the command line.
---------------	----------------------------------

Returns

the value of the option passed as a command line parameter.

6.13.4 Member Data Documentation

6.13.4.1 tokens

```
vector<string> InputParser::tokens [private]
```

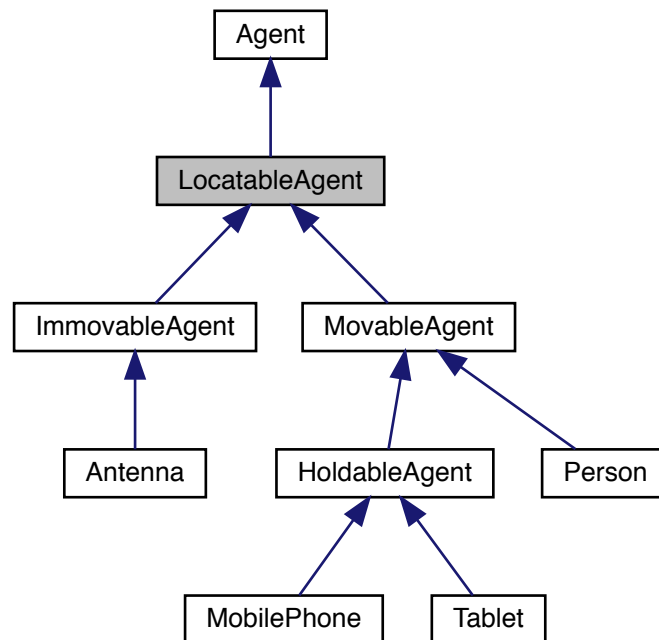
The documentation for this class was generated from the following file:

- include/[InputParser.h](#)

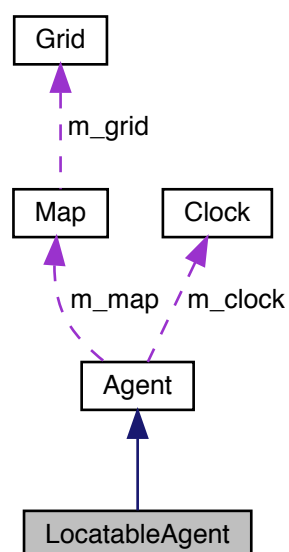
6.14 LocatableAgent Class Reference

```
#include <LocatableAgent.h>
```

Inheritance diagram for LocatableAgent:



Collaboration diagram for LocatableAgent:



Public Member Functions

- [LocatableAgent](#) (const [Map](#) *m, const unsigned long id, Point *initLocation, const [Clock](#) *clock)
- virtual [~LocatableAgent](#) ()
- const string [getName](#) () const override
- const string [toString](#) () const override
- virtual Point * [getLocation](#) () const
- virtual void [setLocation](#) (Point *location)
- const string [dumpLocation](#) ()

Private Attributes

- Point * [m_location](#)

6.14.1 Detailed Description

This class extends the [Agent](#) class and defines an object with a location on a map.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 LocatableAgent()

```
LocatableAgent::LocatableAgent (
    const Map * m,
    const unsigned long id,
    Point * initLocation,
    const Clock * clock ) [explicit]
```

Constructor of the class. Builds an object that has a location on the map of the simulation.

Parameters

<i>m</i>	a pointer to a Map object used in this simulation.
<i>id</i>	the id of the object.
<i>initLocation</i>	the initial location of the object.
<i>clock</i>	a pointer to a Clock object used in this simulation.

6.14.2.2 ~LocatableAgent()

```
virtual LocatableAgent::~~LocatableAgent ( ) [virtual]
```

Destructor

6.14.3 Member Function Documentation

6.14.3.1 dumpLocation()

```
const string LocatableAgent::dumpLocation ( )
```

Builds a human readable string representation of the location

Returns

a human readable string representation of the location

6.14.3.2 getLocation()

```
virtual Point* LocatableAgent::getLocation ( ) const [virtual]
```

Returns

the location on the map as a pointer to a Point object.

6.14.3.3 getName()

```
const string LocatableAgent::getName ( ) const [override], [virtual]
```

Returns the name of this class.

Returns

the name of this class.

Implements [Agent](#).

Reimplemented in [MobilePhone](#), [Person](#), [MovableAgent](#), and [Tablet](#).

6.14.3.4 setLocation()

```
virtual void LocatableAgent::setLocation (
    Point * location ) [virtual]
```

Sets the location of the agent on the map.

Parameters

<i>location</i>	the location of the agent on the map passed as a pointer to a Point object.
-----------------	---

Reimplemented in [Person](#), and [HoldableAgent](#).

6.14.3.5 toString()

```
const string LocatableAgent::toString ( ) const [override], [virtual]
```

Builds a human readable string representation of this class useful to output it to a file or on the screen.

Returns

a string representation of this class.

Implements [Agent](#).

Reimplemented in [Person](#), [MobilePhone](#), [MovableAgent](#), and [Tablet](#).

6.14.4 Member Data Documentation

6.14.4.1 m_location

```
Point* LocatableAgent::m_location [private]
```

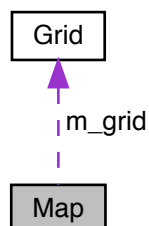
The documentation for this class was generated from the following file:

- include/[LocatableAgent.h](#)

6.15 Map Class Reference

```
#include <Map.h>
```

Collaboration diagram for Map:



Public Member Functions

- [Map](#) ()
- [Map](#) (double llX, double llY, double width, double height)
- [Map](#) (string wktFile)
- virtual [~Map](#) ()
- const GeometryFactory::Ptr & [getGlobalFactory](#) () const
- Geometry * [getBoundary](#) () const
- void [setBoundary](#) (Geometry *boundary)
- const [Grid](#) * [getGrid](#) () const
- void [addGrid](#) (double dimTileX, double dimTileY)

Private Member Functions

- Polygon * [create_rectangle](#) (double llX, double llY, double width, double height)

Private Attributes

- GeometryFactory::Ptr [m_globalFactory](#)
- Geometry * [m_boundary](#)
- [Grid](#) * [m_grid](#)

6.15.1 Detailed Description

This is the map where the simulation takes place. It could be a simple rectangle or any kind of geometry object(s) read from a wkt file. The map has a boundary that is implemented as a Geometry object, a factory object of GeometryFactory type used to create other objects. All geometric and geographic features of the simulator uses the GEOS library. GEOS is an open source C++ library which is just a port to C++ of the well known Java Topology Suite.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 [Map\(\)](#) [1/3]

```
Map::Map ( )
```

Creates a map with a null boundary. The user may set the boundary later, using [setBoundary\(\)](#) method.

6.15.2.2 [Map\(\)](#) [2/3]

```
Map::Map (
    double llX,
    double llY,
    double width,
    double height )
```

Build a simple map of a rectangular shape.

Parameters

<i>llX</i>	X coordinate of the bottom left corner of the rectangle.
<i>llY</i>	Y coordinate of the bottom left corner of the rectangle.
<i>width</i>	the width of the rectangle.
<i>height</i>	the height of the rectangle.

6.15.2.3 Map() [3/3]

```
Map::Map (
    string wktFile )
```

Builds a map reading it from a .wkt file.

Parameters

<i>wktFile</i>	the name of the .wkt file that contains the description of the map. Currently, the first row of this file should contain the external boundary geometry of the map.
----------------	---

6.15.2.4 ~Map()

```
virtual Map::~~Map ( ) [virtual]
```

Default destructor

6.15.3 Member Function Documentation

6.15.3.1 addGrid()

```
void Map::addGrid (
    double dimTileX,
    double dimTileY )
```

Adds a [Grid](#) that overlaps this [Map](#) objects.

Parameters

<i>dimTileX</i>	the dimension on OX of a tile of the Grid object.
<i>dimTileY</i>	the dimension on OY of a tile of the Grid object.

6.15.3.2 create_rectangle()

```
Polygon* Map::create_rectangle (
    double llX,
    double llY,
    double width,
    double height ) [private]
```

6.15.3.3 getBoundary()

```
Geometry* Map::getBoundary ( ) const
```

Returns a pointer to the Geometry object that represents the external boundary of the map.

Returns

a pointer to the Geometry object that represents the external boundary of the map.

6.15.3.4 getGlobalFactory()

```
const GeometryFactory::Ptr& Map::getGlobalFactory ( ) const
```

Returns a pointer to the GeometryFactory object which is a factory object used to create other geometric objects. The GEOS library allows users to create geometric objects only using this factory, all the constructors are made private.

Returns

a pointer to the GeometryFactory object used to create other geometric objects.

6.15.3.5 getGrid()

```
const Grid* Map::getGrid ( ) const
```

Returns a pointer to the [Grid](#) object associated with this [Map](#). After creation of a [Map](#) the user should associate a [Grid](#) that overlaps this [Map](#). The grid is used to compute the probability of the localization of different events during the simulation.

Returns

a pointer the [Grid](#) object associated with this [Map](#).

6.15.3.6 setBoundary()

```
void Map::setBoundary (
    Geometry * boundary )
```

Sets the boundary of the [Map](#) object.

Parameters

<i>boundary</i>	the boundary of the Map object.
-----------------	---

6.15.4 Member Data Documentation

6.15.4.1 m_boundary

```
Geometry* Map::m_boundary [private]
```

6.15.4.2 m_globalFactory

```
GeometryFactory::Ptr Map::m_globalFactory [private]
```

6.15.4.3 m_grid

```
Grid* Map::m_grid [private]
```

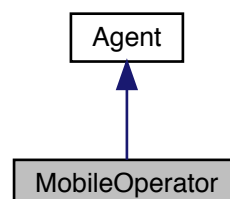
The documentation for this class was generated from the following file:

- [include/Map.h](#)

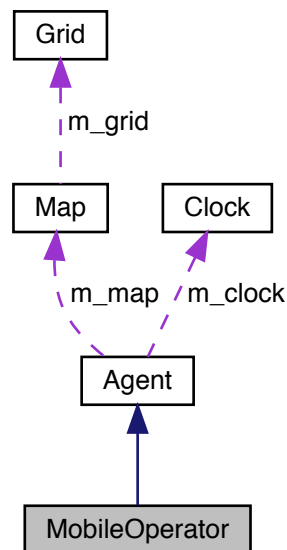
6.16 MobileOperator Class Reference

```
#include <MobileOperator.h>
```

Inheritance diagram for MobileOperator:



Collaboration diagram for MobileOperator:



Public Member Functions

- `MobileOperator` (const `Map` *m, const unsigned long id, const `Clock` *clock, const char *name, const double probMobilePhone)
- virtual `~MobileOperator` ()
- const string `getName` () const override
- const string `toString` () const override
- const string `getMNOName` () const
- const double `getProbMobilePhone` () const
- ofstream & `getAntennaCellsFile` ()
- ofstream & `getSignalFile` ()

Private Attributes

- const string `m_name`
- const double `m_probMobilePhone`
- ofstream `m_antennaCellsFileName`
- ofstream `m_signalMeasureFileName`

6.16.1 Detailed Description

This class represents a Mobile Operator company. Currently a simulation can be run with 1 or 2 mobile operators. A mobile operator own a set of antennas and has a set of mobile phone subscribed.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 MobileOperator()

```
MobileOperator::MobileOperator (
    const Map * m,
    const unsigned long id,
    const Clock * clock,
    const char * name,
    const double probMobilePhone )
```

Constructor of this class. It builds a [MobileOperator](#) object with the characteristics provided by user through parameters. It builds a string representing the name of the file where the coverage area of all the antennas that belong to this [MobileOperator](#) are saved in .csv format and then open this file and writes the header of the file. The name of this file is built concatenating "AntennaCells_" with the name of the Mobile Operator. The extension of the file is ".csv". A line of this file contains the antenna id followed by a wkt text representing the coverage area of antenna. It also builds another string representing the name of a file where the signal quality values for all antennas that belong to this [MobileOperator](#) are saved. The signal quality is computed in the center of each tile of the [Grid](#) object set for a simulation. The file is opened and then it writes the header of the file. The name of this file is built concatenating "SignalQuality_" with the name of the Mobile Operator. The extension of the file is ".csv". A line of this file contains the antenna id followed by a set of values for the signal quality computed in the center of each tile of the grid.

Parameters

<i>m</i>	a pointer to a Map object where the simulation take place.
<i>id</i>	the id of the MobileOperator object.
<i>clock</i>	a pointer to a Clock object used for a simulation.
<i>name</i>	the name of the Mobile Operator.
<i>probMobilePhone</i>	represents the probability that a person will have a cell phone at this company.

6.16.2.2 ~MobileOperator()

```
virtual MobileOperator::~~MobileOperator ( ) [virtual]
```

Default destructor.

6.16.3 Member Function Documentation

6.16.3.1 getAntennaCellsFile()

```
ofstream& MobileOperator::getAntennaCellsFile ( )
```

Returns

a file where the coverage area of all the antennas that belong to this [MobileOperator](#) are saved in csv format.

6.16.3.2 getMNOName()

```
const string MobileOperator::getMNOName ( ) const
```

The name of the Mobile Operator. It should be provided as a parameter to the constructor of the class.

Returns

The name of the Mobile Operator

6.16.3.3 getName()

```
const string MobileOperator::getName ( ) const [override], [virtual]
```

Overrides the same method from the superclass.

Returns

the name of the class, i.e. "MobileOperator".

Implements [Agent](#).

6.16.3.4 getProbMobilePhone()

```
const double MobileOperator::getProbMobilePhone ( ) const
```

The probability that a person will have a cell phone at this company. It should be provided as a parameter to the constructor of the class.

Returns

the probability that a person will have a cell phone at this company.

6.16.3.5 getSignalFile()

```
ofstream& MobileOperator::getSignalFile ( )
```

Returns

a file where the signal quality/strength/power values for all antennas belonging to this mobile Operator and all tiles of the grid are saved.

6.16.3.6 toString()

```
const string MobileOperator::toString ( ) const [override], [virtual]
```

Overrides the same method from the superclass. It is used to write the characteristics of the Mobile Operator to a file or to console.

Returns

a string that describes the parameters of the MobieOperator.

Implements [Agent](#).

6.16.4 Member Data Documentation

6.16.4.1 m_antennaCellsFileName

```
ofstream MobileOperator::m_antennaCellsFileName [private]
```

6.16.4.2 m_name

```
const string MobileOperator::m_name [private]
```

6.16.4.3 m_probMobilePhone

```
const double MobileOperator::m_probMobilePhone [private]
```

6.16.4.4 m_signalMeasureFileName

```
ofstream MobileOperator::m_signalMeasureFileName [private]
```

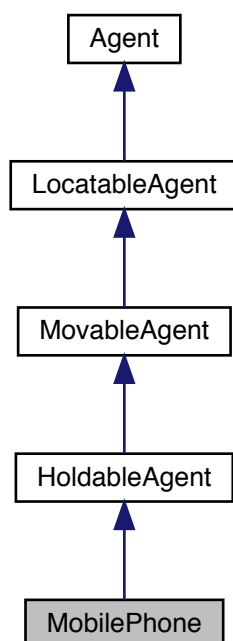
The documentation for this class was generated from the following file:

- include/[MobileOperator.h](#)

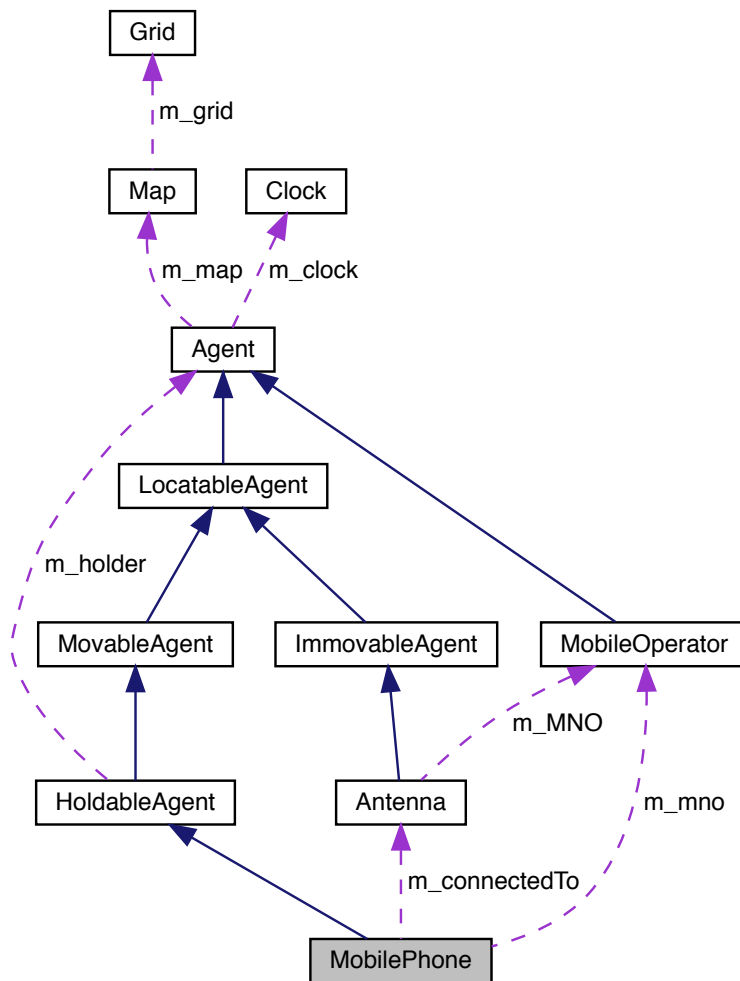
6.17 MobilePhone Class Reference

```
#include <MobilePhone.h>
```

Inheritance diagram for MobilePhone:



Collaboration diagram for MobilePhone:



Public Member Functions

- **MobilePhone** (const **Map** *m, const unsigned long id, Point *initPosition, **Agent** *holder, const **Clock** *clock, double threshold, **HoldableAgent::CONNECTION_TYPE** connType)
- virtual **~MobilePhone** ()
- const string **getName** () const override
- const string **toString** () const override
- Point * **move** (**MovementType** mvType) override
- bool **tryConnect** () override
- const **MobileOperator** * **getMobileOperator** () const
- void **setMobileOperator** (**MobileOperator** *mno)
- double **getConnectionThreshold** () const

Private Attributes

- double `m_threshold`
- `Antenna * m_connectedTo`
- `HoldableAgent::CONNECTION_TYPE m_connType`
- `MobileOperator * m_mno`

Additional Inherited Members

6.17.1 Detailed Description

This class represents a mobile phone. A mobile phone is own by a [Person](#) and it moves on the map together with its owner. While moving, at every time step it tries to connect to an antenna. The connection event is triggered by `setLocation()`. The connection to antenna is determined by the signal emitted by antennas. A parameter in the simulation configuration file set the criterion used to connect: the power of the signal, the signal strength or the signal quality.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 MobilePhone()

```
MobilePhone::MobilePhone (
    const Map * m,
    const unsigned long id,
    Point * initPosition,
    Agent * holder,
    const Clock * clock,
    double threshold,
    HoldableAgent::CONNECTION_TYPE connType ) [explicit]
```

Builds a new MobilewPhone object with the parameters provided by the user.

Parameters

<i>m</i>	a pointer to the Map object where the simulation takes place.
<i>id</i>	the id of the mobile phone.
<i>initPosition</i>	the initial location of the phone on the map.
<i>holder</i>	a pointer to the Agent object that owns this mobile phone.
<i>clock</i>	a pointer to the Clock object used in this simulation.
<i>threshold</i>	the minimum power, signal qauality or signal strength of the field below which the mobile phone cannot connect to an antenna.
<i>connType</i>	the criterion used for the connection to an antenna: based on the power of the signal or based on the signal quality. It could take three values: HoldableAgent::CONNECTION_TYPE::USING_POWER, HoldableAgent::CONNECTION_TYPE::USING_SIGNAL_QUALITY or HoldableAgent::CONNECTION_TYPE::USING_SIGNAL_STRENGTH.

6.17.2.2 ~MobilePhone()

```
virtual MobilePhone::~~MobilePhone ( ) [virtual]
```

The default destructor.

6.17.3 Member Function Documentation

6.17.3.1 getConnectionThreshold()

```
double MobilePhone::getConnectionThreshold ( ) const
```

Returns the minimum value of the signal strength/power/quality below which the phone cannot use the signal (i.e. the signal is considered noise). The returned value is interpreted as signal strength, power or quality according to the connection type.

Returns

the minimum value of the signal strength/power/quality below which the phone cannot use the signal (i.e. the signal is considered noise).

6.17.3.2 getMobileOperator()

```
const MobileOperator* MobilePhone::getMobileOperator ( ) const
```

Returns the [MobileOperator](#) object of this mobile phone. Each [MobilePhone](#) should belong to a Mobile Operator.

Returns

the [MobileOperator](#) object of this mobile phone. Each [MobilePhone](#) should belong to a Mobile Operator.

6.17.3.3 getName()

```
const string MobilePhone::getName ( ) const [override], [virtual]
```

Returns the name of this class.

Returns

the name of this class.

Reimplemented from [HoldableAgent](#).

6.17.3.4 move()

```
Point* MobilePhone::move (
    MovementType mvType ) [inline], [override], [virtual]
```

Makes a step on the map according to an algorithm. The direction and the length of the step is determined by the mvType parameter and by the [Person](#) object who owns this phone.

Parameters

<i>mvType</i>	selects the way people and their phones are moving on the map. At this moment only RANDOM_WALK_CLOSED_MAP and RANDOM_WALK_CLOSED_MAP_WITH_DRIFT are implemented. RANDOM_WALK_CLOSED_MAP means that at each time instant the direction is generated as a uniformly distributed random value and the step length is computed multiplying the speed with the time interval set in the simulation configuration file. If a step projects it outside the map, it stops on the boundary. MovementType::RANDOM_WALK_CLOSED_MAP_WITH_DRIFT means that there is a preference in the direction of the movement. There are two constants defined, SIM_TREND_ANGLE_1 and SIM_TREND_ANGLE_2 (3PI/4 and 5PI/4), and in the first half of the simulation the direction is generated as a normal distributed random value with the mean equals to SIM_TREND_ANGLE_1 and sd = 0.1 while during the second half of the simulation it is generated as a normal distributed random value with the mean equals to SIM_TREND_ANGLE_2 and the same sd. Again, a MovableAgent can only move inside the map boundary. If a step projects it outside the map, it stops on the boundary.
---------------	---

Returns

the final location after the movement.

Implements [MovableAgent](#).

6.17.3.5 setMobileOperator()

```
void MobilePhone::setMobileOperator (
    MobileOperator * mno )
```

Sets the [MobileOperator](#) object which owns this phone.

Parameters

<i>mno</i>	the MobileOperator object which owns this phone.
------------	--

6.17.3.6 toString()

```
const string MobilePhone::toString ( ) const [override], [virtual]
```

Returns a human readable string representation of this class useful to output it to a file or console.

Returns

a human readable string representation of this class.

Reimplemented from [HoldableAgent](#).

6.17.3.7 tryConnect()

```
bool MobilePhone::tryConnect ( ) [override], [virtual]
```

This method is called after the phone moves (together with its owner) to a new location. It tries to connect the mobile phone to an antenna. The connection method is determined by inspecting the `m_connType`: using the power of the signal, using the quality of the signal or using the signal strength. The value of the `m_connType` is set by the constructor of the class. If the connection is successfully a pointer to the [Antenna](#) object where this mobile phone was connected is stored internally.

Returns

true if the connection succeeds, false otherwise.

Implements [HoldableAgent](#).

6.17.4 Member Data Documentation

6.17.4.1 m_connectedTo

```
Antenna\* MobilePhone::m_connectedTo [private]
```

6.17.4.2 m_connType

```
HoldableAgent::CONNECTION\_TYPE MobilePhone::m_connType [private]
```

6.17.4.3 m_mno

```
MobileOperator\* MobilePhone::m_mno [private]
```

6.17.4.4 m_threshold

```
double MobilePhone::m_threshold [private]
```

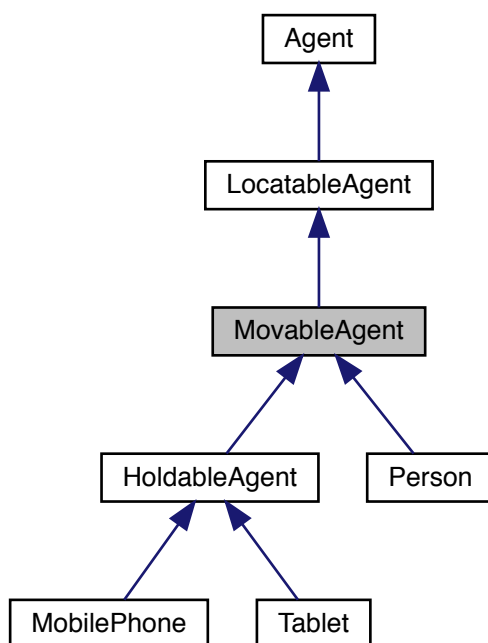
The documentation for this class was generated from the following file:

- [include/MobilePhone.h](#)

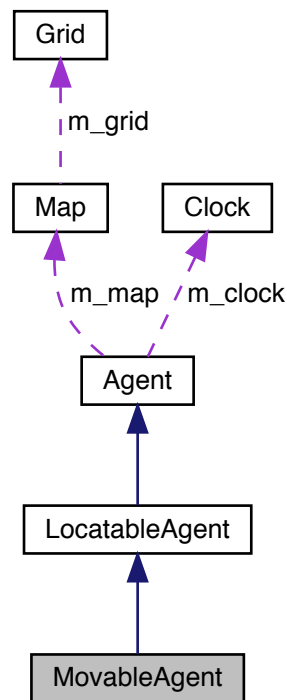
6.18 MovableAgent Class Reference

```
#include <MovableAgent.h>
```

Inheritance diagram for MovableAgent:



Collaboration diagram for MovableAgent:



Public Member Functions

- **MovableAgent** (const **Map** *m, const unsigned long id, Point *initPosition, const **Clock** *clock, double initSpeed)
- virtual **~MovableAgent** ()
- const string **getName** () const override
- const string **toString** () const override
- virtual Point * **move** (**MovementType** type)=0
- double **getSpeed** () const
- void **setSpeed** (double speed)

Private Attributes

- double **m_speed**

6.18.1 Detailed Description

This class represents an **Agent** that can move inside the map. This is an abstract class. It is used as a base class for all agents that can move. Currently **Person** and **MobilePhone** classes inherit it and are already implemented.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 MovableAgent()

```
MovableAgent::MovableAgent (
    const Map * m,
    const unsigned long id,
    Point * initPosition,
    const Clock * clock,
    double initSpeed ) [explicit]
```

Constructor of the class.

Parameters

<i>m</i>	a pointer to a Map object used by the simulation.
<i>id</i>	the id of the object.
<i>initPosition</i>	the initial location on map.
<i>clock</i>	a pointer to the Clock object used by this simulation.
<i>initSpeed</i>	the initial speed of the agent. Depending on the derived classes, this value could be read from a configuration file. For example, for the Person class, this value is specified in the persons.xml configuration file.

6.18.2.2 ~MovableAgent()

```
virtual MovableAgent::~MovableAgent ( ) [virtual]
```

The default destructor.

6.18.3 Member Function Documentation

6.18.3.1 getName()

```
const string MovableAgent::getName ( ) const [override], [virtual]
```

Returns the name of the class.

Returns

the name of the class.

Reimplemented from [LocatableAgent](#).

Reimplemented in [Person](#), and [Tablet](#).

6.18.3.2 `getSpeed()`

```
double MovableAgent::getSpeed ( ) const
```

Returns the speed of this agent.

Returns

the speed of this agent.

6.18.3.3 `move()`

```
virtual Point* MovableAgent::move (
    MovementType type ) [pure virtual]
```

A pure virtual method used to move the agent to a new location on the map.

Parameters

<i>type</i>	<p>the type of the movement. At this moment there are two values accepted for this parameter: MovementType::RANDOM_WALK_CLOSED_MAP and MovementType::RANDOM_WALK_CLOSED_MAP_WITH_DRIFT. MovementType::RANDOM_WALK_CLOSED_MAP means that at each time instant, the direction is generated as a uniformly distributed random number and the step length is computed multiplying the speed with the time interval set in the simulation configuration file. The agent can only move inside the map boundary. If a step projects it outside the map, it stops on the boundary. MovementType::RANDOM_WALK_CLOSED_MAP_WITH_DRIFT means that there is a preference in the direction of the movement. There are two constants defined, <code>SIM_TREND_ANGLE_1</code> and <code>SIM_TREND_ANGLE_2</code> ($3\pi/4$ and $5\pi/4$), and in the first half of the simulation the direction is generated as a normal distributed random value with the mean equals to <code>SIM_TREND_ANGLE_1</code> and <code>sd = 0.1</code> while during the second half of the simulation it is generated as a normal distributed random value with the mean equals to <code>SIM_TREND_ANGLE_2</code> and the same <code>sd</code>. Again, a MovableAgent can only move inside the map boundary. If a step projects it outside the map, it stops on the boundary.</p>
-------------	---

Returns

Implemented in [Person](#), [MobilePhone](#), and [Tablet](#).

6.18.3.4 `setSpeed()`

```
void MovableAgent::setSpeed (
    double speed )
```

Sets the speed of this agent.

Parameters

<i>speed</i>	the speed of this agent.
--------------	--------------------------

6.18.3.5 toString()

```
const string MovableAgent::toString ( ) const  [override], [virtual]
```

Builds and returns a human readable string representation of the agent.

Returns

a human readable string representation of the agent.

Reimplemented from [LocatableAgent](#).

Reimplemented in [Person](#), and [Tablet](#).

6.18.4 Member Data Documentation**6.18.4.1 m_speed**

```
double MovableAgent::m_speed  [private]
```

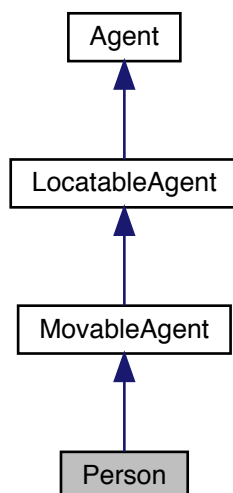
The documentation for this class was generated from the following file:

- [include/MovableAgent.h](#)

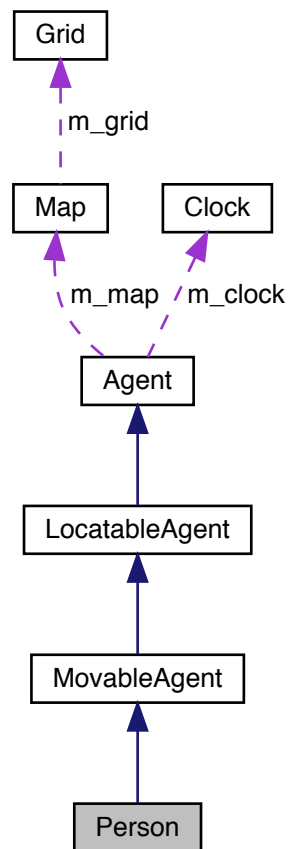
6.19 Person Class Reference

```
#include <Person.h>
```

Inheritance diagram for Person:



Collaboration diagram for Person:



Public Types

- enum [Gender](#) { [MALE](#), [FEMALE](#) }
- enum [AgeDistributions](#) { [NORMAL](#), [UNIFORM](#) }

Public Member Functions

- [Person](#) (const [Map](#) *m, const unsigned long id, Point *initPosition, const [Clock](#) *clock, double initSpeed, int age, [Gender](#) gender, unsigned long timeStay, unsigned long intervalBetweenStays)
- virtual [~Person](#) ()
- const string [getName](#) () const override
- const string [toString](#) () const override
- string [dumpDevices](#) ()
- bool [hasDevices](#) ()
- int [getAge](#) () const
- void [setAge](#) (int age)
- Point * [move](#) ([MovementType](#) mvType) override
- virtual void [setLocation](#) (Point *pt) override

- void `addDevice` (string type, `Agent *agent`)
- `Gender` `getGender` () const
- unsigned long `getAvgTimeStay` () const
- unsigned long `getAvgIntervalBetweenStays` () const

Private Member Functions

- void `randomWalkClosedMap` ()
- void `randomWalkClosedMapDrift` ()
- Point * `generateNewLocation` (double theta)
- void `setNewLocation` (Point *p, bool changeDirection)

Private Attributes

- int `m_age`
- `Gender` `m_gender`
- unordered_multimap< string, `Agent *` > `m_idDevices`
- bool `m_changeDirection`
- unsigned long `m_avgTimeStay`
- unsigned long `m_timeStay`
- unsigned long `m_avgIntervalBetweenStays`
- unsigned long `m_nextStay`

6.19.1 Detailed Description

This class represents a person that can have 0,1 or 2 mobile phone(s). During the simulation the person move around the map, carrying his/her mobile devices.

6.19.2 Member Enumeration Documentation

6.19.2.1 AgeDistributions

```
enum Person::AgeDistributions
```

Enumerator

NORMAL	
UNIFORM	

6.19.2.2 Gender

```
enum Person::Gender
```

Enumerator

MALE	
FEMALE	

6.19.3 Constructor & Destructor Documentation

6.19.3.1 Person()

```

Person::Person (
    const Map * m,
    const unsigned long id,
    Point * initPosition,
    const Clock * clock,
    double initSpeed,
    int age,
    Gender gender,
    unsigned long timeStay,
    unsigned long intervalBetweenStays ) [explicit]

```

Builds a new [Person](#) object with the characteristics given as parameters.

Parameters

<i>m</i>	a pointer to the Map object where this Person move.
<i>id</i>	the id of the Person .
<i>initPosition</i>	the initial location of the person on the map.
<i>clock</i>	a pointer to a Clock object used for this simulation.
<i>initSpeed</i>	the initial speed of this person. It is provided in the configuration file.
<i>age</i>	the age of the person. The age is generated using a uniform or a normal distribution.
<i>gender</i>	the gender of the person.
<i>timeStay</i>	the average time of a stop
<i>intervalBetweenStays</i>	the average time between two consecutive stops.

6.19.3.2 ~Person()

```

virtual Person::~~Person ( ) [virtual]

```

The default destructor.

6.19.4 Member Function Documentation

6.19.4.1 addDevice()

```
void Person::addDevice (
    string type,
    Agent * agent )
```

Add a mobile device to this person. Internally, all mobile devices are kept in an unordered_multimap as pairs <name of the device class, pointer to the device object>

Parameters

<i>type</i>	the name of the device's class.
<i>agent</i>	a pointer to the device object.

6.19.4.2 dumpDevices()

```
string Person::dumpDevices ( )
```

Builds a string containing a list with the ids of the mobile devices that this person owns.

Returns

a string containing a list with the ids of the mobile devices that this person owns.

6.19.4.3 generateNewLocation()

```
Point* Person::generateNewLocation (
    double theta ) [private]
```

6.19.4.4 getAge()

```
int Person::getAge ( ) const
```

Returns the age of the person.

Returns

the age of the person.

6.19.4.5 getAvgIntervalBetweenStays()

```
unsigned long Person::getAvgIntervalBetweenStays ( ) const
```

The average time interval between two stops. It is given in the simulation.xml configuration file.

Returns

The average time interval between two stops.

6.19.4.6 getAvgTimeStay()

```
unsigned long Person::getAvgTimeStay ( ) const
```

The average time interval a person stay in the same location. It is given in the simulation.xml configuration file.

Returns

the average time a person stay in the same location.

6.19.4.7 getGender()

```
Gender Person::getGender ( ) const
```

Returns the gender of the person.

Returns

the gender of the person.

6.19.4.8 getName()

```
const string Person::getName ( ) const [override], [virtual]
```

Returns the name of this class.

Returns

the name of this class.

Reimplemented from [MovableAgent](#).

6.19.4.9 hasDevices()

```
bool Person::hasDevices ( )
```

returns true if this person has at least a mobile device, false otherwise.

Returns

6.19.4.10 move()

```
Point* Person::move (
    MovementType mvType ) [override], [virtual]
```

Move the person to another location. Computes the coordinates of the new location and then call [setLocation\(\)](#) with this new coordinates.

Parameters

<i>mvType</i>	specifies the method used to compute the new position of the person, i.e. how the direction and the length of the step are computed. It can have the following values: <code>RANDOM_WALK_CLOSED_MAP</code> - the agent moves randomly inside the map boundary. The direction is generated as a random value at each time step and the step length is computed multiplying the speed with the time interval. <code>RANDOM_WALK_CLOSED_MAP_WITH_DRIFT</code> : the agent moves in a preferential direction. There are two constants defining these directions: <code>SIM_TREND_ANGLE_1</code> and <code>SIM_TREND_ANGLE_2</code> ($3\pi/4$ and $5\pi/4$). The actual direction is generated as a normally distributed random value with means equals to these constants and $sd = 0.1$. In both cases, a Person makes several steps then a stop. The average time length of a stop is given in the simulation configuration file while the actual values are generated as normal distributed random values with the mean read from the configuration file and $sd = 20\%$ from mean. The average time interval elapsed between two stop is read from the simulation configuration file and the actual values are generated as exponential distributed random values.
---------------	---

Returns

a pointer to a `Point` object that represents the new location.

Implements [MovableAgent](#).

6.19.4.11 randomWalkClosedMap()

```
void Person::randomWalkClosedMap ( ) [private]
```

6.19.4.12 randomWalkClosedMapDrift()

```
void Person::randomWalkClosedMapDrift ( ) [private]
```

6.19.4.13 setAge()

```
void Person::setAge (
    int age )
```

Sets the age of the person.

Parameters

<i>age</i>	the age of the person.
------------	------------------------

6.19.4.14 setLocation()

```
virtual void Person::setLocation (
    Point * pt ) [override], [virtual]
```

Sets the location of the person on the map.

Parameters

<i>pt</i>	a pointer to a Point object that represent the location of the person on the map. If the person has mobile devices (phone, tablets) this function calls setLocation() for all mobile devices too.
-----------	---

Reimplemented from [LocatableAgent](#).

6.19.4.15 setNewLocation()

```
void Person::setNewLocation (
    Point * p,
    bool changeDirection ) [private]
```

6.19.4.16 toString()

```
const string Person::toString ( ) const [override], [virtual]
```

Builds and returns a human readable string representation of the person.

Returns

a human readable string representation of the person.

Reimplemented from [MovableAgent](#).

6.19.5 Member Data Documentation

6.19.5.1 m_age

```
int Person::m_age [private]
```

6.19.5.2 m_avgIntervalBetweenStays

```
unsigned long Person::m_avgIntervalBetweenStays [private]
```

6.19.5.3 m_avgTimeStay

```
unsigned long Person::m_avgTimeStay [private]
```

6.19.5.4 m_changeDirection

```
bool Person::m_changeDirection [private]
```

6.19.5.5 m_gender

```
Gender Person::m_gender [private]
```

6.19.5.6 m_idDevices

```
unordered_multimap<string, Agent*> Person::m_idDevices [private]
```

6.19.5.7 m_nextStay

```
unsigned long Person::m_nextStay [private]
```

6.19.5.8 m_timeStay

```
unsigned long Person::m_timeStay [private]
```

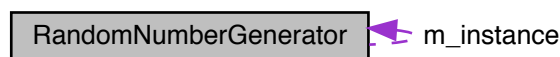
The documentation for this class was generated from the following file:

- include/[Person.h](#)

6.20 RandomNumberGenerator Class Reference

```
#include <RandomNumberGenerator.h>
```

Collaboration diagram for RandomNumberGenerator:



Public Member Functions

- double * [generateNormal2Double](#) (const double m1, const double sd1, const double m2, const double sd2, int n)
- double [generateNormalDouble](#) (const double m, const double sd)
- double * [generateNormalDouble](#) (const double m, const double sd, const int n)
- double * [generateTruncatedNormalDouble](#) (const double a, const double b, const double m, const double sd, const unsigned long n)
- double [generateUniformDouble](#) (const double min, const double max)
- double * [generateUniformDouble](#) (const double min, const double max, const int n)
- double [generateExponentialDouble](#) (const double lambda)
- int [generateUniformInt](#) (const int min, const int max)
- int * [generateUniformInt](#) (const int min, const int max, const int n)
- int [generateBinomialInt](#) (const int max, const double p)
- int [generateBernoulliInt](#) (const double p)
- int * [generateBinomialInt](#) (const int max, const double p, const int n)
- int * [generateBernoulliInt](#) (const double p, const int n)
- double [normal_pdf](#) (double x, double m, double s)
- void [setSeed](#) (unsigned seed)

Static Public Member Functions

- static [RandomNumberGenerator](#) * [instance](#) ()
- static [RandomNumberGenerator](#) * [instance](#) (unsigned seed)

Private Member Functions

- [RandomNumberGenerator](#) ()
- [RandomNumberGenerator](#) (unsigned seed)
- [RandomNumberGenerator](#) (const [RandomNumberGenerator](#) &)
- [RandomNumberGenerator](#) & operator= (const [RandomNumberGenerator](#) &)

Private Attributes

- uniform_int_distribution< int > [m_unif_int_distribution](#)
- uniform_real_distribution< double > [m_unif_double_distribution](#)
- normal_distribution< double > [m_normal_double_distribution](#)
- exponential_distribution< double > [m_exponential_double_distribution](#)
- binomial_distribution< int > [m_binomial_distribution](#)
- bernoulli_distribution [m_bernoulli_distribution](#)
- std::mt19937 [m_generator](#)

Static Private Attributes

- static [RandomNumberGenerator](#) * [m_instance](#)

6.20.1 Detailed Description

Utility singleton class to generate random numbers according to different distributions.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 RandomNumberGenerator() [1/3]

```
RandomNumberGenerator::RandomNumberGenerator ( ) [private]
```

6.20.2.2 RandomNumberGenerator() [2/3]

```
RandomNumberGenerator::RandomNumberGenerator (
    unsigned seed ) [private]
```

6.20.2.3 RandomNumberGenerator() [3/3]

```
RandomNumberGenerator::RandomNumberGenerator (
    const RandomNumberGenerator & ) [private]
```

6.20.3 Member Function Documentation

6.20.3.1 generateBernoulliInt() [1/2]

```
int RandomNumberGenerator::generateBernoulliInt (
    const double p )
```

Generates an int random value from a Bernoulli distribution.

Parameters

p	the parameter of the Bernoulli distribution.
-----	--

Returns

an int random value from a Bernoulli distribution.

6.20.3.2 generateBernoulliInt() [2/2]

```
int* RandomNumberGenerator::generateBernoulliInt (
    const double  $p$ ,
    const int  $n$  )
```

Generates n int random values from a Bernoulli distribution.

Parameters

p	the parameter of the Bernoulli distribution.
n	the number of values to be generated.

Returns

array with n int values from a Bernoulli distribution.

6.20.3.3 generateBinomialInt() [1/2]

```
int RandomNumberGenerator::generateBinomialInt (
    const int  $max$ ,
    const double  $p$  )
```

Generates a int random value from a binomial distribution inside the interval $[0, max]$.

Parameters

max	the upper limit of the value.
p	the parameter of the binomial distribution.

Returns

a int random value from a binomial distribution inside the interval $[0, max]$.

6.20.3.4 generateBinomialInt() [2/2]

```
int* RandomNumberGenerator::generateBinomialInt (
    const int max,
    const double p,
    const int n )
```

Generates n int random values from a binomial distribution inside the interval [0, max].

Parameters

<i>max</i>	the upper limit of the value.
<i>p</i>	the parameter of the binomial distribution.
<i>n</i>	the number of values to be generated.

Returns

array with n int values from a binomial distribution.

6.20.3.5 generateExponentialDouble()

```
double RandomNumberGenerator::generateExponentialDouble (
    const double lambda )
```

Generates a random value distributed according to an exponential distribution with parameter lambda.

Parameters

<i>lambda</i>	the parameter of the exponential distribution.
---------------	--

Returns

a random value distributed according to an exponential distribution with parameter lambda.

6.20.3.6 generateNormal2Double()

```
double* RandomNumberGenerator::generateNormal2Double (
    const double m1,
    const double sd1,
    const double m2,
    const double sd2,
    int n )
```

Generates n random numbers with a normal distribution. Half of them are N(m1,sd1), the other half N(m2,sd2).

Parameters

<i>m1</i>	the mean of the first normal distribution.
<i>sd1</i>	the standard deviation of the first normal distribution.
<i>m2</i>	the mean of the second normal distribution.
<i>sd2</i>	the standard deviation of the second normal distribution.
<i>n</i>	the total number of values to be generated.

Returns

an array with random numbers according to two normal distributions.

6.20.3.7 generateNormalDouble() [1/2]

```
double RandomNumberGenerator::generateNormalDouble (
    const double m,
    const double sd )
```

Generates a random value, normally distributed with mean m and standard distribution sd

Parameters

<i>m</i>	the mean of the normal distribution.
<i>sd</i>	the standard deviation of the normal distribution.

Returns

a random value, normally distributed with mean m and standard distribution sd.

6.20.3.8 generateNormalDouble() [2/2]

```
double* RandomNumberGenerator::generateNormalDouble (
    const double m,
    const double sd,
    const int n )
```

Generates an array with n double values normally distributed with mean m and standard deviation sd.

Parameters

<i>m</i>	the mean of the normal distribution.
<i>sd</i>	the standard deviation of the normal distribution.
<i>n</i>	the number of values to be generated.

Returns

an array with n double values normally distributed with mean m and standard deviation sd .

6.20.3.9 generateTruncatedNormalDouble()

```
double* RandomNumberGenerator::generateTruncatedNormalDouble (
    const double a,
    const double b,
    const double m,
    const double sd,
    const unsigned long n )
```

Generates n double values from a truncated normal distribution. All values will be in $[a, b]$.

Parameters

<i>a</i>	the inferior limit of the truncated normal distribution.
<i>b</i>	the superior limit of the truncated normal distribution.
<i>m</i>	the mean of the normal distribution.
<i>sd</i>	the standard deviation of the normal distribution.
<i>n</i>	the number of values to be generated.

Returns

an array with n double values from a truncated normal distribution.

6.20.3.10 generateUniformDouble() [1/2]

```
double RandomNumberGenerator::generateUniformDouble (
    const double min,
    const double max )
```

Generates a random double value from a uniform distribution which lies inside $[min, max]$.

Parameters

<i>min</i>	the lower limit of the value.
<i>max</i>	the upper limit of the value.

Returns

a double value, uniformly distributed in $[min, max]$.

6.20.3.11 generateUniformDouble() [2/2]

```
double* RandomNumberGenerator::generateUniformDouble (
    const double min,
    const double max,
    const int n )
```

Generates *n* uniform distributed random values which lie inside [*min*, *max*].

Parameters

<i>min</i>	the lower limit of the values.
<i>max</i>	the upper limit of the values.
<i>n</i>	the number of values to be generated.

Returns

n array with *n* double values from a uniform distribution.

6.20.3.12 generateUniformInt() [1/2]

```
int RandomNumberGenerator::generateUniformInt (
    const int min,
    const int max )
```

Generates a random int value from a uniform distribution which lies inside [*min*, *max*].

Parameters

<i>min</i>	the lower limit of the value.
<i>max</i>	the upper limit of the value.

Returns

an int value, uniformly distributed in [*min*, *max*].

6.20.3.13 generateUniformInt() [2/2]

```
int* RandomNumberGenerator::generateUniformInt (
    const int min,
    const int max,
    const int n )
```

Generates *n* uniform distributed random values which lie inside [*min*, *max*].

Parameters

<i>min</i>	the lower limit of the values.
<i>max</i>	the upper limit of the values.
<i>n</i>	the number of values to be generated.

Returns

an array with n int values from a uniform distribution.

6.20.3.14 instance() [1/2]

```
static RandomNumberGenerator* RandomNumberGenerator::instance ( ) [inline], [static]
```

Returns an instance of this class.

Returns

n instance of this class.

6.20.3.15 instance() [2/2]

```
static RandomNumberGenerator* RandomNumberGenerator::instance (
    unsigned seed ) [inline], [static]
```

Returns an instance of this class and also sets the seed of the random number generator.

Returns

n instance of this class.

6.20.3.16 normal_pdf()

```
double RandomNumberGenerator::normal_pdf (
    double x,
    double m,
    double s )
```

The value of the PDF of the normal distribution for x.

Parameters

x	the value for which we need the PDF.
m	the mean of the normal distribution.
s	the standard deviation of the normal distribution.

Returns

The value of the PDF of the normal distribution for x .

6.20.3.17 operator=()

```
RandomNumberGenerator& RandomNumberGenerator::operator= (
    const RandomNumberGenerator & ) [private]
```

6.20.3.18 setSeed()

```
void RandomNumberGenerator::setSeed (
    unsigned seed )
```

Sets the seed of the random number generator.

Parameters

<i>seed</i>	
-------------	--

6.20.4 Member Data Documentation**6.20.4.1 m_bernoulli_distribution**

```
bernoulli_distribution RandomNumberGenerator::m_bernoulli_distribution [private]
```

6.20.4.2 m_binomial_distribution

```
binomial_distribution<int> RandomNumberGenerator::m_binomial_distribution [private]
```

6.20.4.3 m_exponential_double_distribution

```
exponential_distribution<double> RandomNumberGenerator::m_exponential_double_distribution  
[private]
```

6.20.4.4 m_generator

```
std::mt19937 RandomNumberGenerator::m_generator [private]
```

6.20.4.5 m_instance

```
RandomNumberGenerator* RandomNumberGenerator::m_instance [static], [private]
```

6.20.4.6 m_normal_double_distribution

```
normal_distribution<double> RandomNumberGenerator::m_normal_double_distribution [private]
```

6.20.4.7 m_unif_double_distribution

```
uniform_real_distribution<double> RandomNumberGenerator::m_unif_double_distribution [private]
```

6.20.4.8 m_unif_int_distribution

```
uniform_int_distribution<int> RandomNumberGenerator::m_unif_int_distribution [private]
```

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

- include/[RandomNumberGenerator.h](#)

6.21 Row Class Reference

```
#include <CSVparser.hpp>
```

Public Member Functions

- [Row](#) (const vector< string > &header)
- [~Row](#) (void)
- unsigned int [size](#) (void) const
- void [push](#) (const string &value)
- bool [set](#) (const string &key, const string &value)
- template<typename T >
const T [getValue](#) (unsigned int pos) const
- const string [operator\[\]](#) (unsigned int i) const
- const string [operator\[\]](#) (const string &valueName) const

Private Attributes

- const vector< string > [_header](#)
- vector< string > [_values](#)

Friends

- ostream & [operator<<](#) (ostream &os, const [Row](#) &row)
- ofstream & [operator<<](#) (ofstream &os, const [Row](#) &row)

6.21.1 Detailed Description

This class is used to represent a line from a .csv file. It is a container of the values from a line of text.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 Row()

```
Row::Row (
    const vector< string > & header )
```

Constructor. It takes a a line of text representing the header of the csv file.

Parameters

<i>header</i>	the header line
---------------	-----------------

6.21.2.2 ~Row()

```
Row::~Row (
    void )
```

Destructor.

6.21.3 Member Function Documentation

6.21.3.1 `getValue()`

```
template<typename T >
const T Row::getValue (
    unsigned int pos ) const [inline]
```

Returns the value of an element from a row

Parameters

<i>pos</i>	the number of the element in row
------------	----------------------------------

Returns

the value

6.21.3.2 `operator[]()` [1/2]

```
const string Row::operator[] (
    unsigned int i ) const
```

Overloaded operator

Parameters

<i>i</i>	the position of the value that we want to extract
----------	---

Returns

the value from position *i*

6.21.3.3 `operator[]()` [2/2]

```
const string Row::operator[] (
    const string & valueName ) const
```

Overloaded operator

Parameters

<i>valueName</i>	the name of the column from which we want to extract a the value
------------------	--

Returns

the value from the column specified by its name

6.21.3.4 push()

```
void Row::push (
    const string & value )
```

Add a new value on a row

Parameters

<i>value</i>	to be added
--------------	-------------

6.21.3.5 set()

```
bool Row::set (
    const string & key,
    const string & value )
```

Sets the value of a specific element in a row. The element is specified by the column name.

Parameters

<i>key</i>	the name of the column (given in the header of the file)
<i>value</i>	the value to be set

Returns**6.21.3.6 size()**

```
unsigned int Row::size (
    void ) const
```

Returns the number of values on a row.

Returns

the number of values on a row.

6.21.4 Friends And Related Function Documentation**6.21.4.1 operator<< [1/2]**

```
ostream& operator<< (
    ostream & os,
    const Row & row ) [friend]
```

Overloaded operator outputs the content of a row in a stream

Parameters

<i>os</i>	the stream where we want to send the content of the row.
<i>row</i>	the row that we want to output

Returns

the same output stream

6.21.4.2 operator<< [2/2]

```
ofstream& operator<< (
    ofstream & os,
    const Row & row ) [friend]
```

Overloaded operator outputs the content of a row in a file stream

Parameters

<i>os</i>	the file stream where we want to send the content of the row.
<i>row</i>	the row that we want to output

Returns

the same output file stream

6.21.5 Member Data Documentation

6.21.5.1 `_header`

```
const vector<string> Row::_header [private]
```

6.21.5.2 `_values`

```
vector<string> Row::_values [private]
```

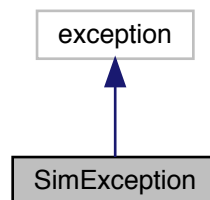
The documentation for this class was generated from the following file:

- [include/CSVparser.hpp](#)

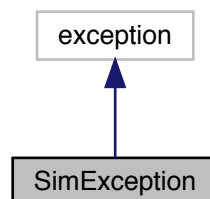
6.22 SimException Class Reference

```
#include <SimException.h>
```

Inheritance diagram for SimException:



Collaboration diagram for SimException:



Public Member Functions

- [SimException](#) (const char *[msg](#))
- virtual const char * [what](#) () const throw ()
- virtual [~SimException](#) ()

Private Attributes

- const char * [msg](#)

6.22.1 Constructor & Destructor Documentation

6.22.1.1 SimException()

```
SimException::SimException (  
    const char * msg )
```

6.22.1.2 ~SimException()

```
virtual SimException::~SimException ( ) [virtual]
```

6.22.2 Member Function Documentation

6.22.2.1 what()

```
virtual const char* SimException::what ( ) const throw ( ) [virtual]
```

6.22.3 Member Data Documentation

6.22.3.1 msg

```
const char* SimException::msg [private]
```

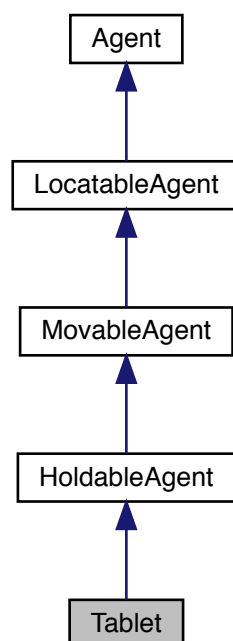
The documentation for this class was generated from the following file:

- include/[SimException.h](#)

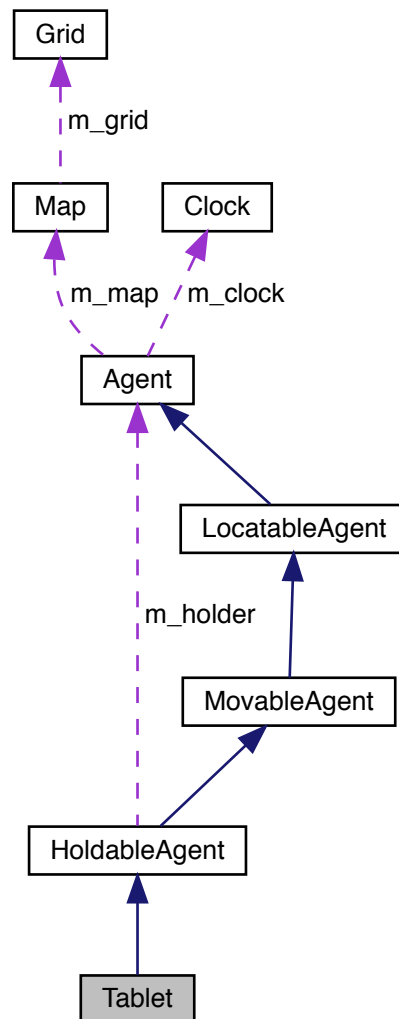
6.23 Tablet Class Reference

```
#include <Tablet.h>
```

Inheritance diagram for Tablet:



Collaboration diagram for Tablet:



Public Member Functions

- **Tablet** (const **Map** *m, const unsigned long id, Point *initPosition, const **Clock** *clock)
- virtual **~Tablet** ()
- const string **getName** () const override
- const string **toString** () const override
- Point * **move** (**MovementType** type) override
- bool **tryConnect** () override

Additional Inherited Members

6.23.1 Constructor & Destructor Documentation

6.23.1.1 Tablet()

```
Tablet::Tablet (
    const Map * m,
    const unsigned long id,
    Point * initPosition,
    const Clock * clock ) [explicit]
```

Builds a new [Tablet](#) object with the parameters provided by the user.

Parameters

<i>m</i>	a pointer to a Map object used for simulation.
<i>id</i>	the id of the tablet.
<i>initPosition</i>	the initial location on map.
<i>clock</i>	a pointer to a Clock object used for simulation..

6.23.1.2 ~Tablet()

```
virtual Tablet::~~Tablet ( ) [virtual]
```

The default destructor.

6.23.2 Member Function Documentation

6.23.2.1 getName()

```
const string Tablet::getName ( ) const [override], [virtual]
```

Returns the name of this class.

Returns

the name of this class.

Reimplemented from [HoldableAgent](#).

6.23.2.2 move()

```
Point* Tablet::move (
    MovementType type ) [inline], [override], [virtual]
```

This method is called to move the tablet (actually the person who own this tablet move) to another location on the map.

Parameters

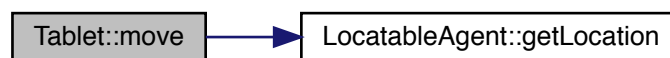
<i>type</i>	the method used to compute the new location. For details see MovableAgent::move() and Person::move() .
-------------	--

Returns

a pointer to a Point object representing the new location on the map.

Implements [MovableAgent](#).

Here is the call graph for this function:

**6.23.2.3 toString()**

```
const string Tablet::toString ( ) const [override], [virtual]
```

Builds a human readable representation of this class.

Returns

a human readable representation of this class.

Reimplemented from [HoldableAgent](#).

6.23.2.4 tryConnect()

```
bool Tablet::tryConnect ( ) [override], [virtual]
```

Called after the tablet changes location on the map, tries to connect to an antenna.

Returns

true if the connection succeeds, false otherwise.

Implements [HoldableAgent](#).

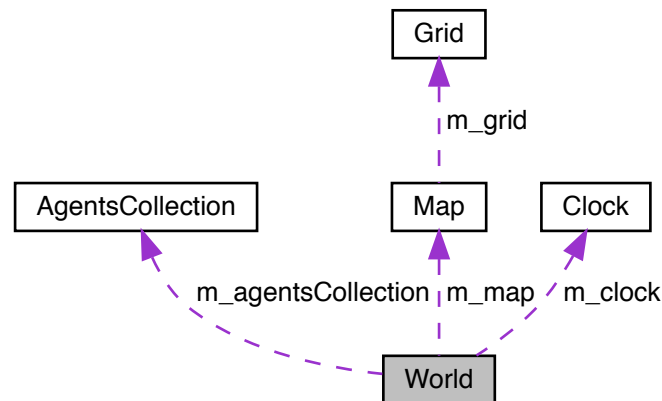
The documentation for this class was generated from the following file:

- [include/Tablet.h](#)

6.24 World Class Reference

```
#include <World.h>
```

Collaboration diagram for World:



Public Member Functions

- [World](#) ([Map](#) *map, int numPersons, int numAntennas, int numMobilePhones)
- [World](#) ([Map](#) *map, const string &configPersonsFileName, const string &configAntennasFileName, const string &configSimulationFileName, const string &probabilitiesFileName) noexcept(false)
- virtual [~World](#) ()
- void [runSimulation](#) () noexcept(false)
- [AgentsCollection](#) * [getAgents](#) () const
- void [setAgents](#) ([AgentsCollection](#) *agents)
- [Clock](#) * [getClock](#) () const
- void [setClock](#) ([Clock](#) *clock)
- const [Map](#) * [getMap](#) () const
- void [setMap](#) ([Map](#) *map)
- const string & [getGridFilename](#) () const
- map< const unsigned long, const string > [getProbFileNames](#) () const
- const string & [getAntennasFilename](#) () const
- const string & [getPersonsFilename](#) () const
- double [getGridDimTileX](#) () const
- double [getGridDimTileY](#) () const
- [PriorType](#) [getPrior](#) () const
- [HoldableAgent::CONNECTION_TYPE](#) [getConnectionType](#) () const

Private Member Functions

- `vector< Person * > generatePopulation` (unsigned long numPersons, double percentHome)
- `vector< Person * > generatePopulation` (const unsigned long numPersons, vector< double > params, [Person::AgeDistributions](#) age_distribution, double male_share, vector< [MobileOperator](#) * > mnos, double speed_walk, double speed_car, double percentHome)
- `vector< Antenna * > generateAntennas` (unsigned long numAntennas)
- `vector< Antenna * > parseAntennas` (const string &configAntennasFile, vector< [MobileOperator](#) * > mnos) noexcept(false)
- `vector< Person * > parsePersons` (const string &personsFileName, vector< [MobileOperator](#) * > mnos) noexcept(false)
- `vector< MobilePhone * > generateMobilePhones` (int numMobilePhones, [HoldableAgent::CONNECTION_TYPE](#) connType)
- `vector< MobileOperator * > parseSimulationFile` (const string &configSimulationFileName) noexcept(false)
- `int whichMNO` (vector< pair< string, double >> probs, vector< [MobileOperator](#) * > mnos)
- `string parseProbabilities` (const string &probabilitiesFileName)
- `double getDefaultConnectionThreshold` ([HoldableAgent::CONNECTION_TYPE](#) connType)

Private Attributes

- `Map * m_map`
- `AgentsCollection * m_agentsCollection`
- `Clock * m_clock`
- unsigned long `m_startTime`
- unsigned long `m_endTime`
- unsigned long `m_timeIncrement`
- double `m_GridDimTileX`
- double `m_GridDimTileY`
- `PriorType m_prior`
- unsigned `m_seed`
- unsigned long `m_stay`
- unsigned `m_intervalBetweenStays`
- double `m_connThreshold`
- `HoldableAgent::CONNECTION_TYPE m_connType`
- `MovementType m_mvType`
- string `m_gridFilename`
- `map< const unsigned long, const string > m_probFilenames`
- string `m_personsFilename`
- string `m_antennasFilename`
- double `m_probSecMobilePhone`

6.24.1 Detailed Description

This is the class where the simulation process takes place. A [World](#) object has a [Map](#), a [Clock](#), a set of Agents than can be persons, mobile phones, antennas, mobile operators etc. After generating all the required objects for simulation by reading the parameters from the input configuration files [runSimulation\(\)](#) method is called to perform the actual simulation. The output of the simulation process is written in several files. [Antenna](#) objects output their registered events in a .csv file and after the simulation ends, these files are merged in a single file that is used to compute the posterior localization probabilities for each mobile device. These probabilities are computed using a [Grid](#) that is overlapped on the [Map](#), i.e. we compute the probability of a mobile phone to be in a tile of the grid. A finer the grid means more accurate localization but this come with an important computational cost.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 World() [1/2]

```
World::World (
    Map * map,
    int numPersons,
    int numAntennas,
    int numMobilePhones )
```

Builds a new [World](#) object, randomly generating a set of persons, antennas and mobile devices with the default parameters. This class is used only for testing and developing new features. For a real simulation the user should build the World object using the other constructor that takes the name of the input files as params.

Parameters

<i>map</i>	a pointer to a Map object where the simulation takes place.
<i>numPersons</i>	the number of persons to be generated.
<i>numAntennas</i>	the number of antennas to be generated.
<i>numMobilePhones</i>	the number of mobile phones to be generated. These phones are randomly given to persons.

6.24.2.2 World() [2/2]

```
World::World (
    Map * map,
    const string & configPersonsFileName,
    const string & configAntennasFileName,
    const string & configSimulationFileName,
    const string & probabilitiesFileName ) [noexcept]
```

Builds a new [World](#) object, reading the parameters for the Persons, Antennas and Mobile Phones from configuration files. The configuration files are in XML format and they should be provided as command line parameters. The general parameters of the simulation (duration, how people move around the map, how mobile phone try to connect to antennas, etc. are also read from a configuration file:

- the persons configuration file is provided through the -p parameter in the command line.
- the antennas configuration file is provided through the -a parameter in the command line.
- the simulation configuration file is provided through the -s parameter in the command line.
- the posterior probabilities configuration file is provided through the -pb parameter in the command line.

Parameters

<i>map</i>	a pointer to a Map object where the simulation takes place
<i>configPersonsFileName</i>	the configuration file name for the persons objects.
<i>configAntennasFileName</i>	the configuration file name for antenna objects.
<i>configSimulationFileName</i>	the general configuration file for a simulation.
<i>probabilitiesFileName</i>	the config file for the posterior location probabilities.

6.24.2.3 ~World()

```
virtual World::~~World ( ) [virtual]
```

Destructor It releases the memory allocated for the agents collection and the [Clock](#) object.

6.24.3 Member Function Documentation

6.24.3.1 generateAntennas()

```
vector<Antenna*> World::generateAntennas (
    unsigned long numAntennas ) [private]
```

6.24.3.2 generateMobilePhones()

```
vector<MobilePhone*> World::generateMobilePhones (
    int numMobilePhones,
    HoldableAgent::CONNECTION_TYPE connType ) [private]
```

6.24.3.3 generatePopulation() [1/2]

```
vector<Person*> World::generatePopulation (
    unsigned long numPersons,
    double percentHome ) [private]
```

6.24.3.4 generatePopulation() [2/2]

```
vector<Person*> World::generatePopulation (
    const unsigned long numPersons,
    vector< double > params,
    Person::AgeDistributions age_distribution,
    double male_share,
    vector< MobileOperator * > mnos,
    double speed_walk,
    double speed_car,
    double percentHome ) [private]
```

6.24.3.5 getAgents()

```
AgentsCollection* World::getAgents ( ) const
```

Returns the [AgentsCollection](#) used in simulation.

Returns

a pointer to [AgentsCollection](#) object.

6.24.3.6 getAntennasFilename()

```
const string& World::getAntennasFilename ( ) const
```

Returns the name of the file where the antennas exact locations are saved for later analysis.

Returns

the name of the file where the antennas exact locations are saved for later analysis.

6.24.3.7 getClock()

```
Clock* World::getClock ( ) const
```

Returns a pointer to a [Clock](#) object used for simulation.

Returns

a pointer to a [Clock](#) object used for simulation.

6.24.3.8 getConnectionType()

```
HoldableAgent::CONNECTION_TYPE World::getConnectionType ( ) const
```

Returns the type of the handover mechanism

Returns

the type of the handover mechanism: `HoldableAgent::CONNECTION_TYPE::USING_SIGNAL_QUALITY`, `HoldableAgent::CONNECTION_TYPE::USING_SIGNAL_STRENGTH`, `HoldableAgent::CONNECTION_TYPE::USING_POWER`

6.24.3.9 getDefaultConnectionThreshold()

```
double World::getDefaultConnectionThreshold (
    HoldableAgent::CONNECTION_TYPE connType ) [private]
```

6.24.3.10 getGridDimTileX()

```
double World::getGridDimTileX ( ) const
```

Returns the dimension of tiles on OX, this number is read from simulation configuration file.

Returns

the dimension of tiles on OX, this number is read from simulation configuration file.

6.24.3.11 getGridDimTileY()

```
double World::getGridDimTileY ( ) const
```

Returns the dimension of tiles on OY, this number is read from simulation configuration file.

Returns

the dimension of tiles on OY, this number is read from simulation configuration file.

6.24.3.12 getGridFilename()

```
const string& World::getGridFilename ( ) const
```

Returns the file name where the grid parameters are saved. They are needed for the visualization software.

Returns

the file name where the grid parameters are saved.

6.24.3.13 getMap()

```
const Map* World::getMap ( ) const
```

Returns a pointer to a [Map](#) object where the simulation takes place.

Returns

a pointer to a [Map](#) object where the simulation takes place.

6.24.3.14 getPersonsFilename()

```
const string& World::getPersonsFilename ( ) const
```

Returns the name of the file where the persons exact locations are saved for later analysis.

Returns

the name of the file where the persons exact locations are saved for later analysis.

6.24.3.15 getPrior()

```
PriorType World::getPrior ( ) const
```

Returns the type of the prior probability used to compute the posterior localization probability.

Returns

the type of the prior probability used to compute the posterior localization probability.

6.24.3.16 getProbFilenames()

```
map<const unsigned long, const string> World::getProbFilenames ( ) const
```

Returns the name of the file where the probabilities of mobile phones locations are saved.

Returns

the name of the file where the probabilities of mobile phones locations are saved.

6.24.3.17 parseAntennas()

```
vector<Antenna*> World::parseAntennas (
    const string & configAntennasFile,
    vector< MobileOperator * > mnos ) [private], [noexcept]
```

6.24.3.18 parsePersons()

```
vector<Person*> World::parsePersons (
    const string & personsFileName,
    vector< MobileOperator * > mnos ) [private], [noexcept]
```

6.24.3.19 parseProbabilities()

```
string World::parseProbabilities (
    const string & probabilitiesFileName ) [private]
```

6.24.3.20 parseSimulationFile()

```
vector<MobileOperator*> World::parseSimulationFile (
    const string & configSimulationFileName ) [private], [noexcept]
```

6.24.3.21 runSimulation()

```
void World::runSimulation ( ) [noexcept]
```

This method is called to perform the actual simulation. During the simulation it outputs the exact positions of all persons in a .csv file and the positions of antennas at the starting time of the simulation. A simulation means a number of time steps, at each step every person move to another position and after arriving at their new positions the mobile phones that they carry try to connect to one of the available antennas. The antennas record these events and output them in a file.

6.24.3.22 setAgents()

```
void World::setAgents (
    AgentsCollection * agents )
```

Sets the [AgentsCollection](#) to be used for simulation.

Parameters

<i>agents</i>	a pointer to AgentsCollection object.
---------------	---

6.24.3.23 setClock()

```
void World::setClock (
    Clock * clock )
```

Sets the [Clock](#) of the simulation.

Parameters

<i>clock</i>	a pointer to a Clock object used for simulation.
--------------	--

6.24.3.24 setMap()

```
void World::setMap (
    Map * map )
```

Sets the map where the simulation takes place.

Parameters

<i>map</i>	a pointer to a Map object where the simulation takes place.
------------	---

6.24.3.25 whichMNO()

```
int World::whichMNO (
    vector< pair< string, double >> probs,
    vector< MobileOperator * > mnos ) [private]
```

6.24.4 Member Data Documentation

6.24.4.1 m_agentsCollection

```
AgentsCollection* World::m_agentsCollection [private]
```

6.24.4.2 m_antennasFilename

```
string World::m_antennasFilename [private]
```

6.24.4.3 m_clock

```
Clock* World::m_clock [private]
```

6.24.4.4 m_connThreshold

```
double World::m_connThreshold [private]
```

6.24.4.5 m_connType

```
HoldableAgent::CONNECTION_TYPE World::m_connType [private]
```

6.24.4.6 m_endTime

```
unsigned long World::m_endTime [private]
```

6.24.4.7 m_GridDimTileX

```
double World::m_GridDimTileX [private]
```

6.24.4.8 m_GridDimTileY

```
double World::m_GridDimTileY [private]
```

6.24.4.9 m_gridFilename

```
string World::m_gridFilename [private]
```

6.24.4.10 m_intervalBetweenStays

```
unsigned World::m_intervalBetweenStays [private]
```

6.24.4.11 m_map

```
Map* World::m_map [private]
```

6.24.4.12 m_mvType

```
MovementType World::m_mvType [private]
```

6.24.4.13 m_personsFilename

```
string World::m_personsFilename [private]
```

6.24.4.14 m_prior

```
PriorType World::m_prior [private]
```

6.24.4.15 m_probFileNames

```
map<const unsigned long, const string> World::m_probFileNames [private]
```

6.24.4.16 m_probSecMobilePhone

```
double World::m_probSecMobilePhone [private]
```

6.24.4.17 m_seed

```
unsigned World::m_seed [private]
```

6.24.4.18 m_startTime

```
unsigned long World::m_startTime [private]
```

6.24.4.19 m_stay

```
unsigned long World::m_stay [private]
```

6.24.4.20 m_timeIncrement

```
unsigned long World::m_timeIncrement [private]
```

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

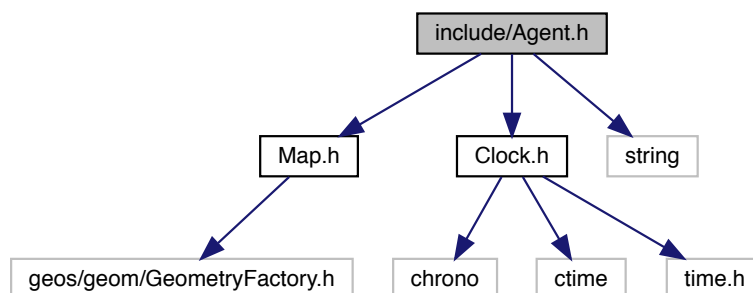
- [include/World.h](#)

Chapter 7

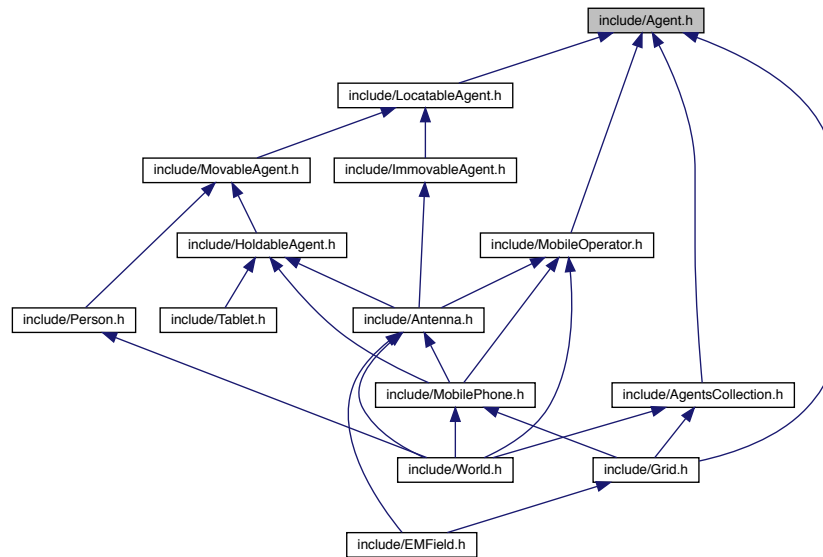
File Documentation

7.1 include/Agent.h File Reference

```
#include <Map.h>  
#include <Clock.h>  
#include <string>  
Include dependency graph for Agent.h:
```



This graph shows which files directly or indirectly include this file:



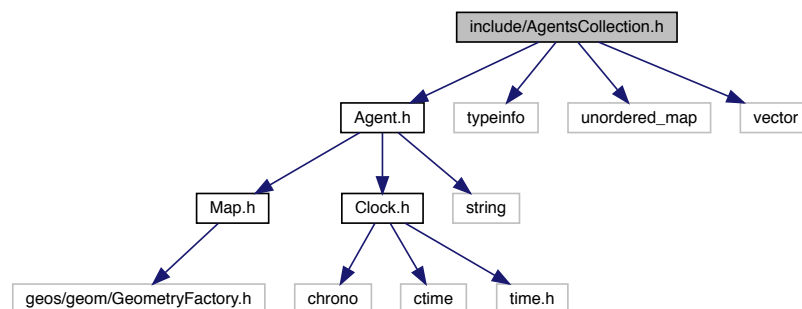
Classes

- class [Agent](#)

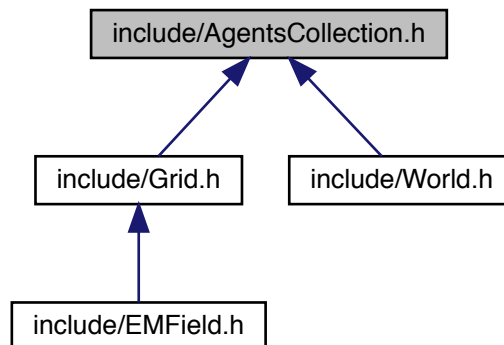
7.2 include/AgentsCollection.h File Reference

```
#include <Agent.h>
#include <typeinfo>
#include <unordered_map>
#include <vector>
```

Include dependency graph for AgentsCollection.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [AgentsCollection](#)

Typedefs

- typedef unordered_multimap< string, [Agent](#) * >::iterator [um_iterator](#)

7.2.1 Typedef Documentation

7.2.1.1 um_iterator

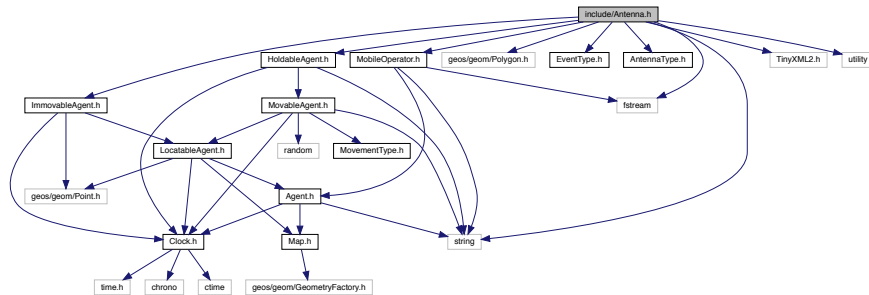
```
typedef unordered_multimap<string, Agent*>::iterator um\_iterator
```

7.3 include/Antenna.h File Reference

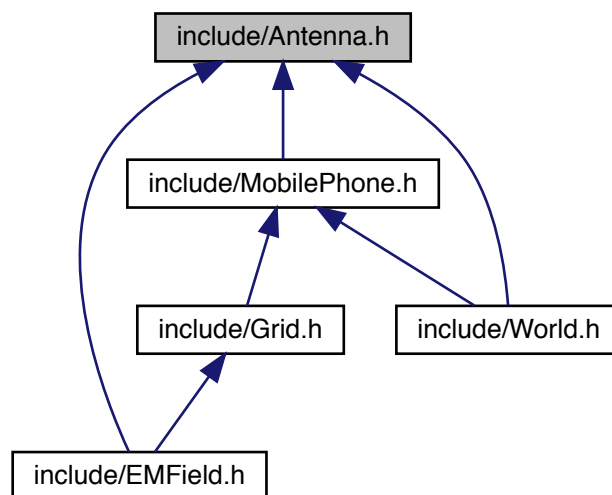
```
#include <ImmovableAgent.h>
#include <HoldableAgent.h>
#include <geos/geom/Polygon.h>
#include <EventType.h>
#include <AntennaType.h>
#include <MobileOperator.h>
#include <TinyXML2.h>
#include <string>
#include <fstream>
```

```
#include <utility>
```

Include dependency graph for Antenna.h:



This graph shows which files directly or indirectly include this file:



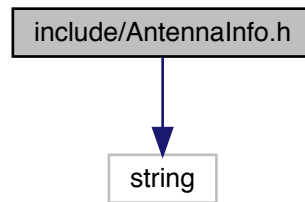
Classes

- class [Antenna](#)

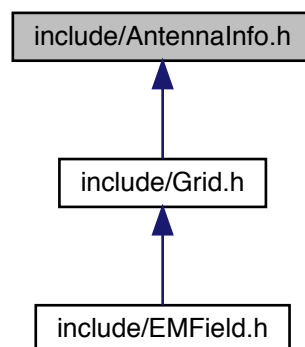
7.4 include/AntennaInfo.h File Reference

```
#include <string>
```


Include dependency graph for AntennaInfo.h:



This graph shows which files directly or indirectly include this file:

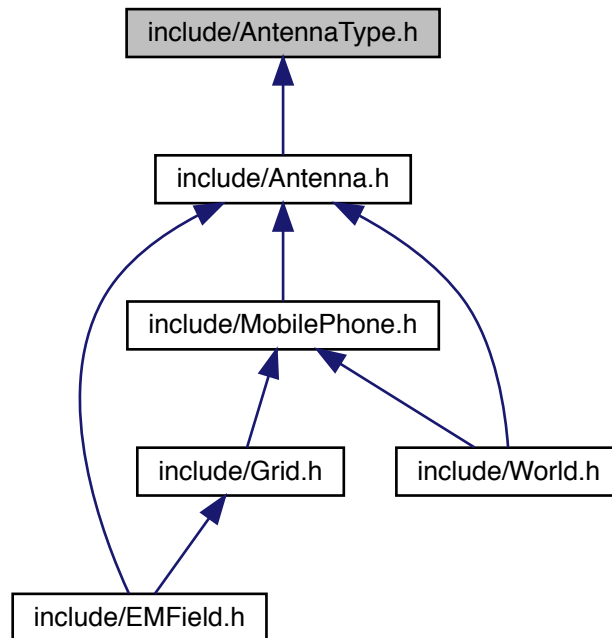


Classes

- class [AntennaInfo](#)

7.5 include/AntennaType.h File Reference

This graph shows which files directly or indirectly include this file:



Enumerations

- enum [AntennaType](#) { [AntennaType::OMNIDIRECTIONAL](#), [AntennaType::DIRECTIONAL](#) }

7.5.1 Enumeration Type Documentation

7.5.1.1 AntennaType

```
enum AntennaType [strong]
```

An enum class that is used to represent the type of an antenna. There are two types of antennas supported: omnidirectional and directional.

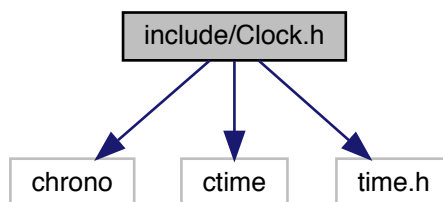
Enumerator

OMNIDIRECTIONAL	
DIRECTIONAL	

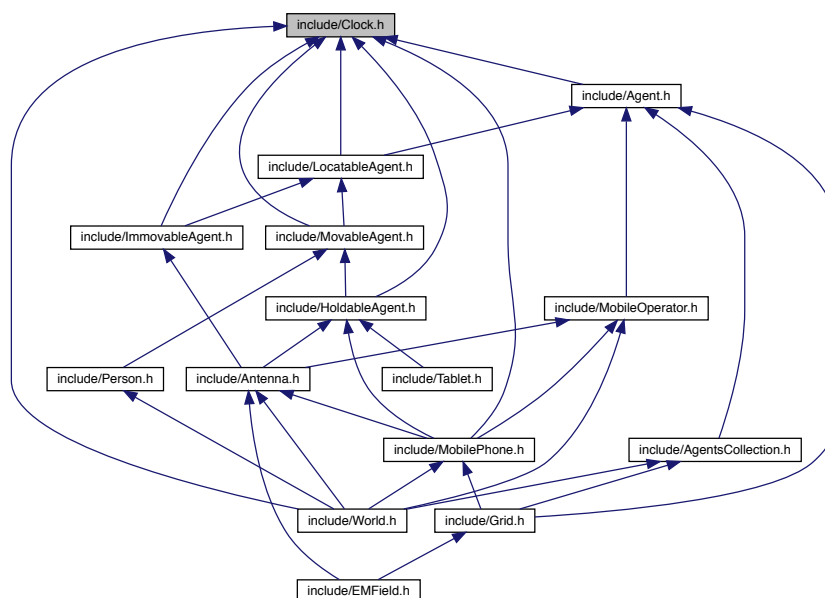
7.6 include/Clock.h File Reference

```
#include <chrono>
#include <ctime>
#include <time.h>
```

Include dependency graph for Clock.h:



This graph shows which files directly or indirectly include this file:



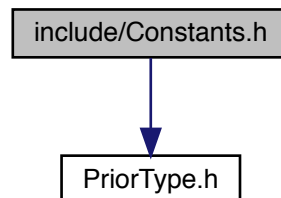
Classes

- class [Clock](#)

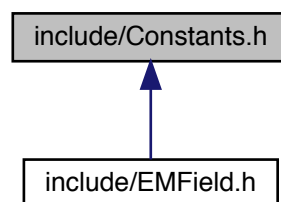
7.7 include/Constants.h File Reference

```
#include <PriorType.h>
```

Include dependency graph for Constants.h:



This graph shows which files directly or indirectly include this file:



Classes

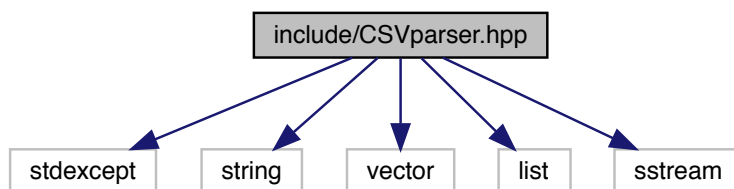
- class [Constants](#)

7.8 include/CSVparser.hpp File Reference

```
#include <stdexcept>
#include <string>
#include <vector>
#include <list>
```

```
#include <sstream>
```

Include dependency graph for CSVparser.hpp:



Classes

- class [Row](#)
- class [CSVParser](#)

Enumerations

- enum [DataType](#) { [eFILE](#) = 0, [ePURE](#) = 1 }

7.8.1 Enumeration Type Documentation

7.8.1.1 DataType

```
enum DataType
```

Enumerator

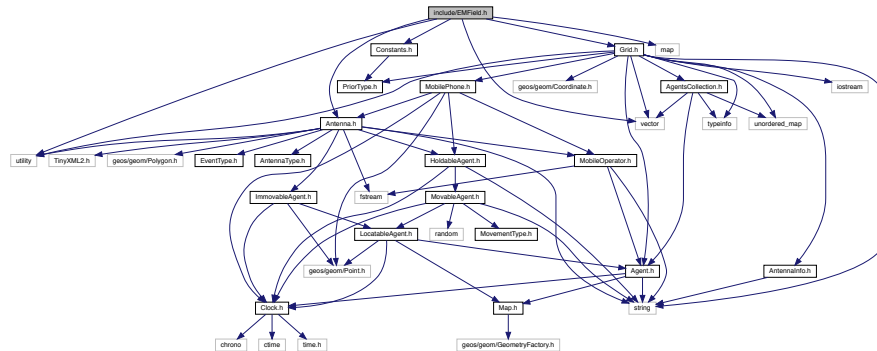
eFILE	
ePURE	

7.9 include/EMField.h File Reference

```
#include <Antenna.h>
#include <Constants.h>
#include <Grid.h>
#include <utility>
#include <vector>
```

```
#include <map>
```

Include dependency graph for EMField.h:

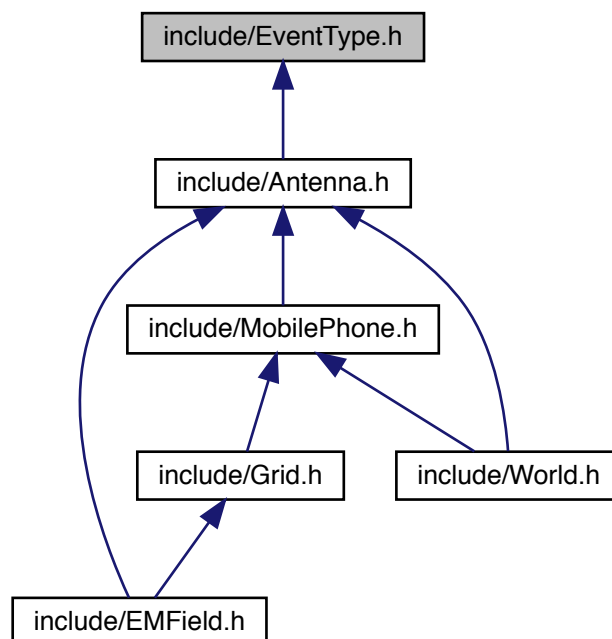


Classes

- class [EMField](#)

7.10 include/EventType.h File Reference

This graph shows which files directly or indirectly include this file:



Enumerations

- enum [EventType](#) { [EventType::ATTACH_DEVICE](#), [EventType::DETACH_DEVICE](#), [EventType::ALREADY_ATTACHED_DEVICE](#), [EventType::IN_RANGE_NOT_ATTACHED_DEVICE](#) }

7.10.1 Enumeration Type Documentation

7.10.1.1 EventType

```
enum EventType [strong]
```

This class is an enumeration of network events currently registered: [ATTACH_DEVICE](#) - means that an antenna connects a new mobile device. [DETACH_DEVICE](#) - means that an antenna disconnects a mobile device, i.e. this mobile device is no longer connected to that antenna. [ALREADY_ATTACHED_DEVICE](#) - means that an antenna is connected with a mobile device and that mobile was connected to the same antenna in the previous time step too. [IN_RANGE_NOT_ATTACHED_DEVICE](#) - means that a mobile device tried to connect to an antenna, the antenna provided enough signal power/quality but the connection was not successful (for example antenna is at its maximum capacity and cannot connect any new devices).

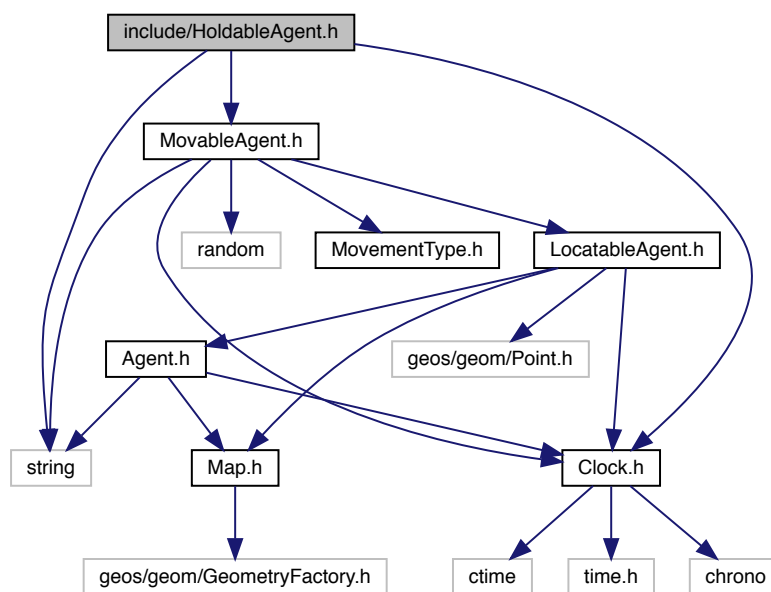
Enumerator

ATTACH_DEVICE	
DETACH_DEVICE	
ALREADY_ATTACHED_DEVICE	
IN_RANGE_NOT_ATTACHED_DEVICE	

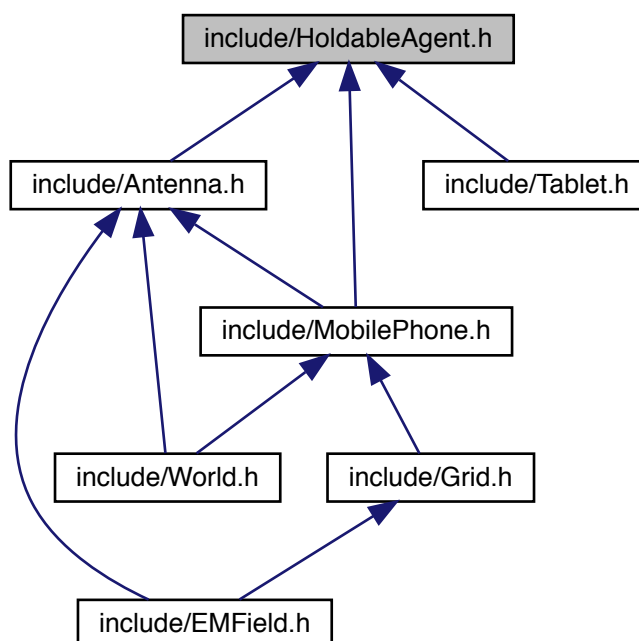
7.11 include/Grid.h File Reference

```
#include <string>
#include <iostream>
#include <vector>
#include <geos/geom/Coordinate.h>
#include <MobilePhone.h>
#include <AntennaInfo.h>
#include <Agent.h>
#include <AgentsCollection.h>
#include <typeinfo>
#include <unordered_map>
#include <utility>
#include <PriorType.h>
```


Include dependency graph for HoldableAgent.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [HoldableAgent](#)

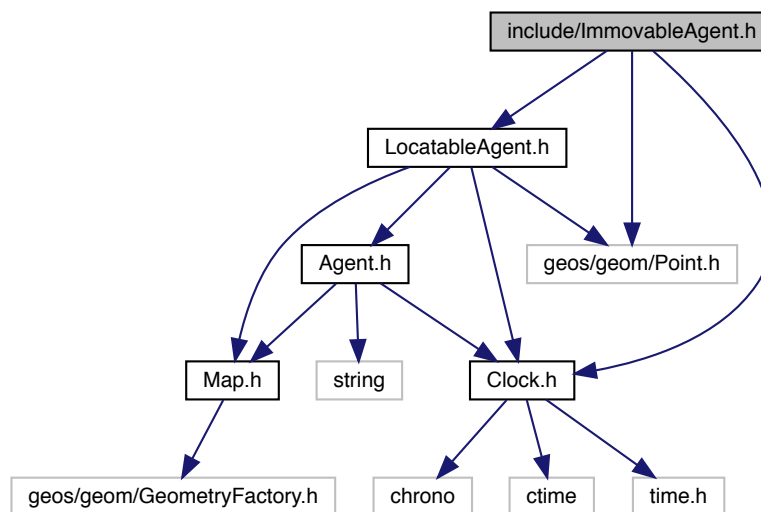
7.13 include/IDGenerator.h File Reference

Classes

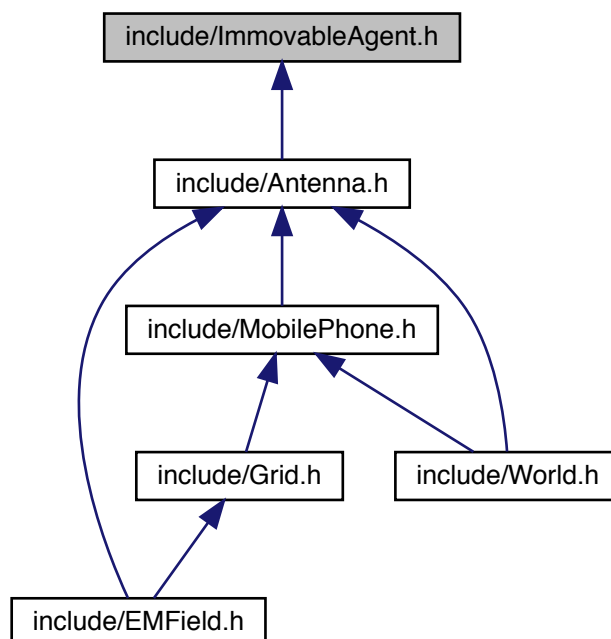
- class [IDGenerator](#)

7.14 include/ImmovableAgent.h File Reference

```
#include "LocatableAgent.h"  
#include <Clock.h>  
#include <geos/geom/Point.h>  
Include dependency graph for ImmovableAgent.h:
```



This graph shows which files directly or indirectly include this file:



Classes

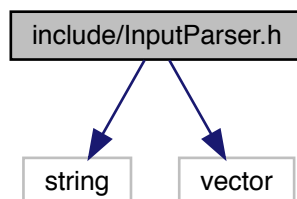
- class `ImmovableAgent`

7.15 include/InputParser.h File Reference

```
#include <string>
```

```
#include <vector>
```

Include dependency graph for `InputParser.h`:



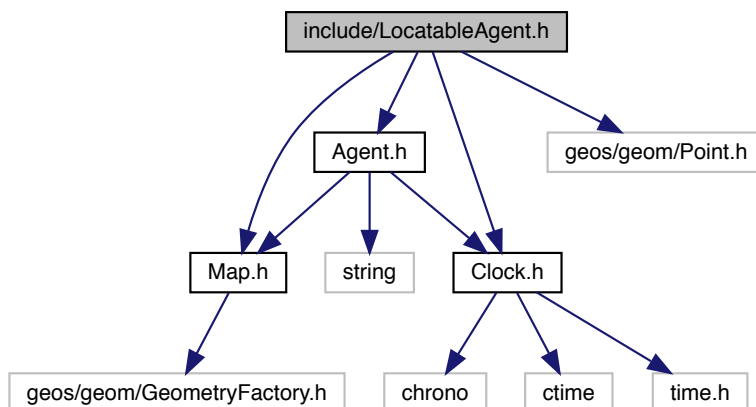
Classes

- class [InputParser](#)

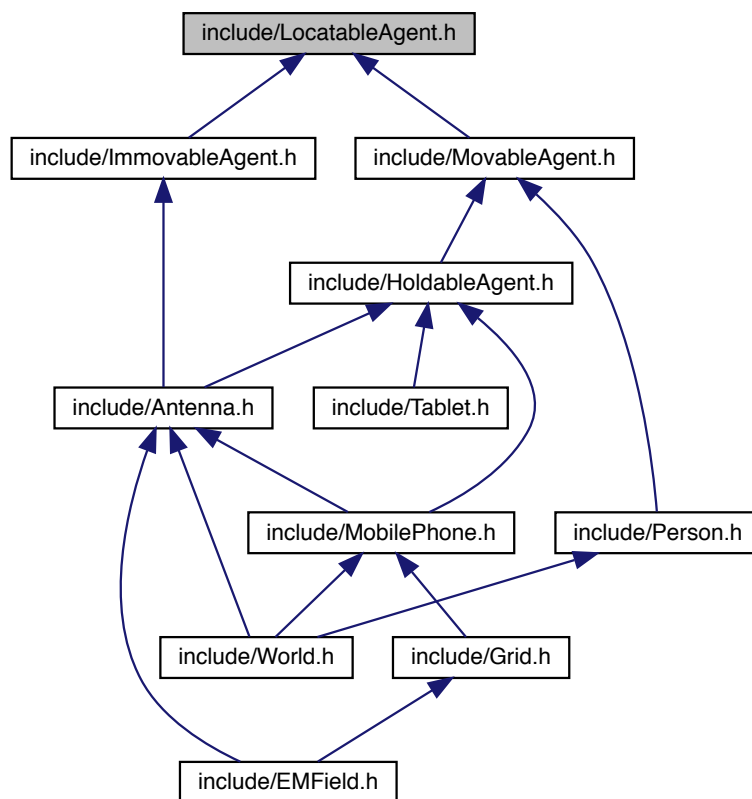
7.16 include/LocatableAgent.h File Reference

```
#include <Agent.h>
#include <Clock.h>
#include <Map.h>
#include <geos/geom/Point.h>
```

Include dependency graph for LocatableAgent.h:



This graph shows which files directly or indirectly include this file:



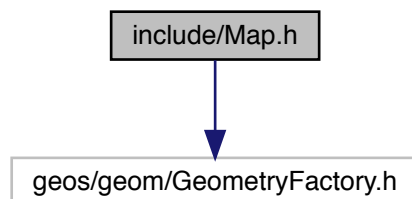
Classes

- class [LocatableAgent](#)

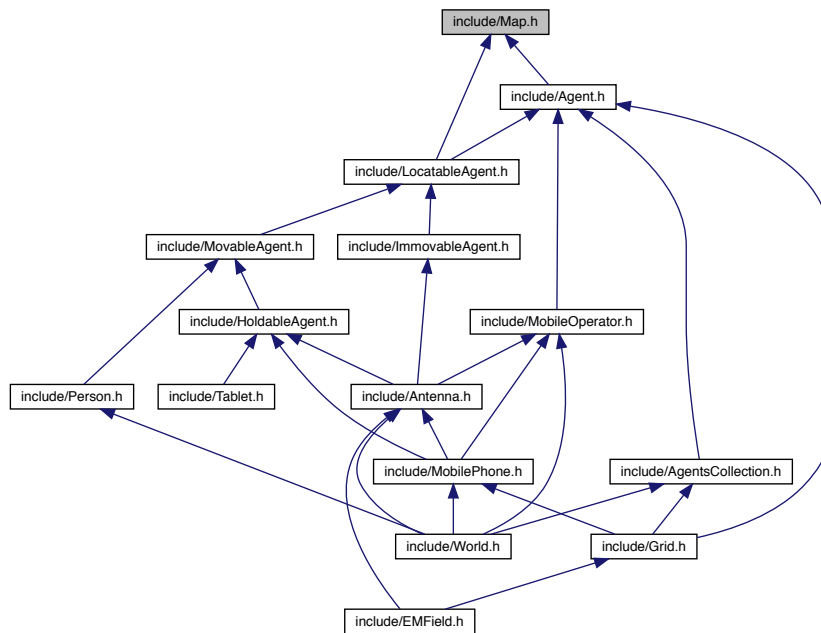
7.17 include/Map.h File Reference

```
#include <geos/geom/GeometryFactory.h>
```

Include dependency graph for Map.h:



This graph shows which files directly or indirectly include this file:



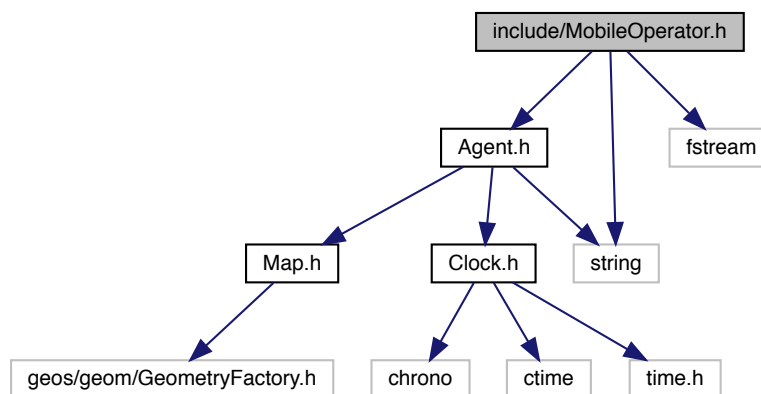
Classes

- class [Map](#)

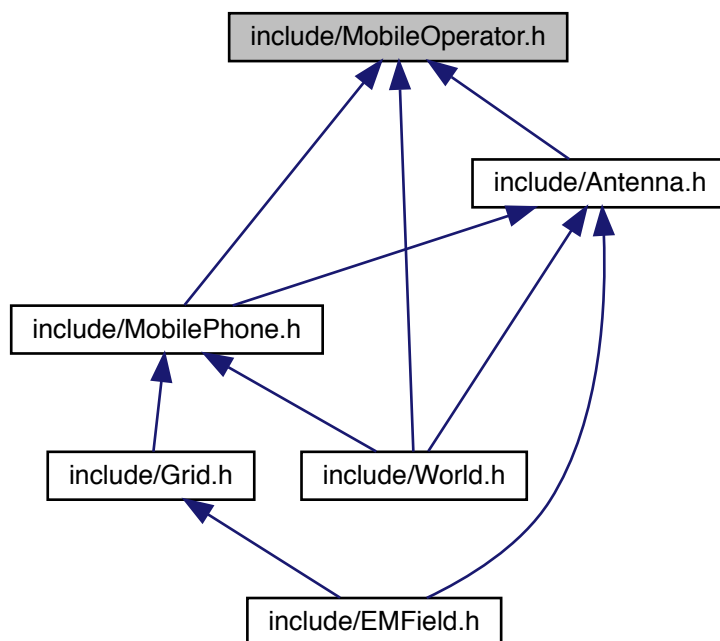
7.18 include/MobileOperator.h File Reference

```
#include <Agent.h>
#include <string>
#include <fstream>
```

Include dependency graph for MobileOperator.h:



This graph shows which files directly or indirectly include this file:



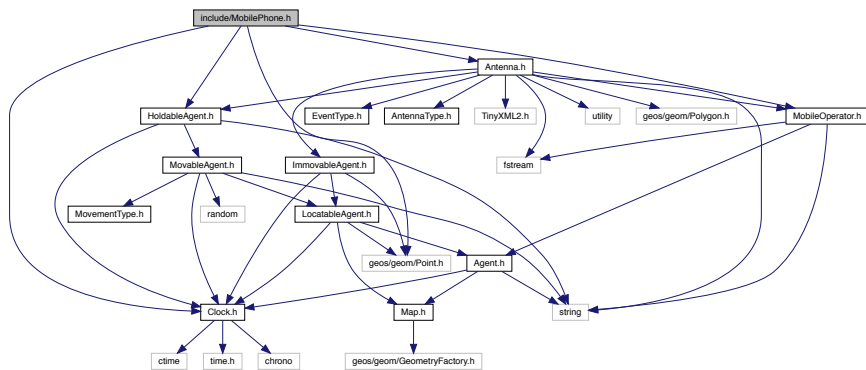
Classes

- class [MobileOperator](#)

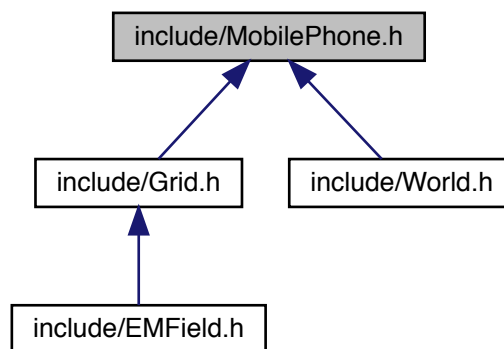
7.19 include/MobilePhone.h File Reference

```
#include <HoldableAgent.h>
#include <MobileOperator.h>
#include <Antenna.h>
#include <Clock.h>
#include <geos/geom/Point.h>
```

Include dependency graph for MobilePhone.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [MobilePhone](#)

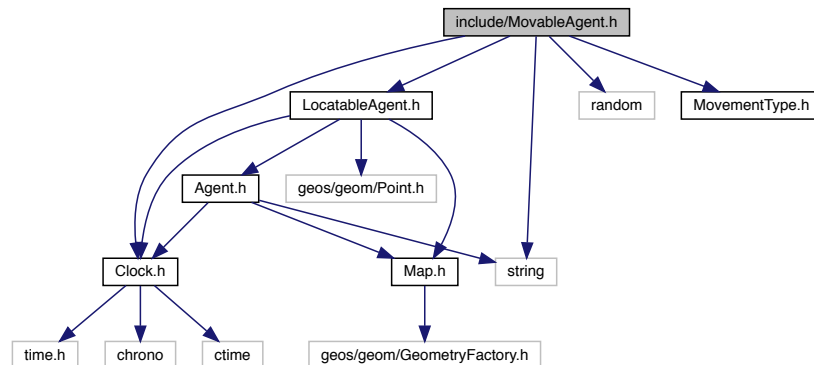
7.20 include/MovableAgent.h File Reference

```
#include <LocatableAgent.h>
#include <Clock.h>
#include <random>
#include <MovementType.h>
```

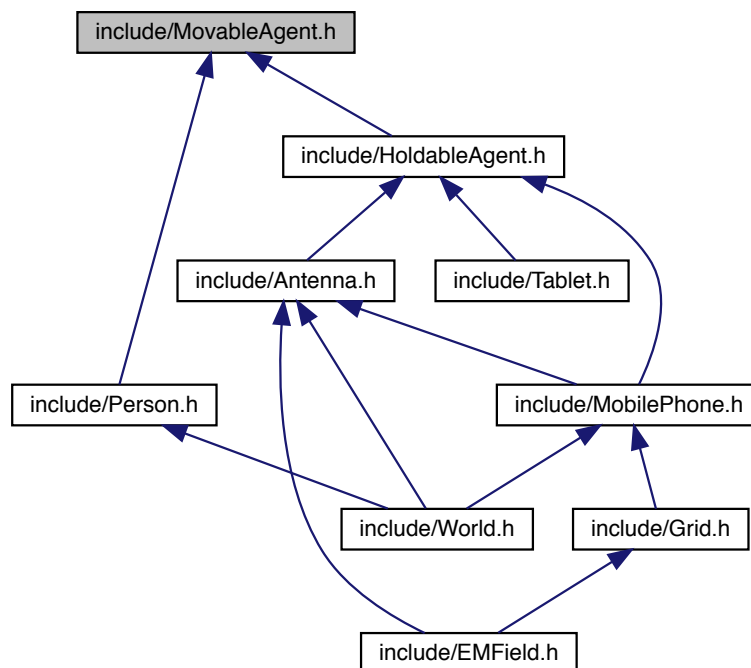


```
#include <string>
```

Include dependency graph for MovableAgent.h:



This graph shows which files directly or indirectly include this file:

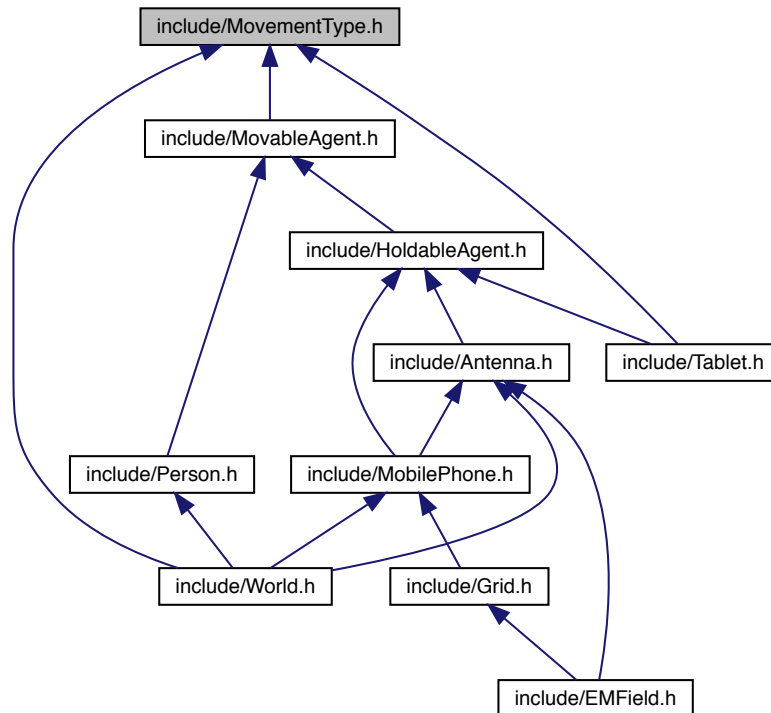


Classes

- class [MovableAgent](#)

7.21 include/MovementType.h File Reference

This graph shows which files directly or indirectly include this file:



Enumerations

- enum `MovementType` { `MovementType::RANDOM_WALK_CLOSED_MAP`, `MovementType::RANDOM_WALK_CLOSED_MAP`, `MovementType::UNKNOWN` }

7.21.1 Enumeration Type Documentation

7.21.1.1 MovementType

```
enum MovementType [strong]
```

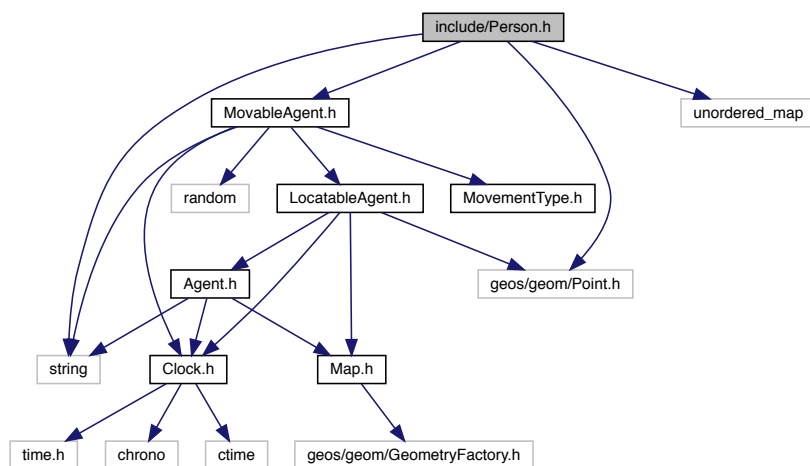
An enum class that enumerates the types of the methods used to move the people on the map. `RANDOM_WALK_CLOSED_MAP` - the agent moves randomly inside the map boundary. The direction is generated as a random value at each time step and the step length is computed multiplying the speed with the time interval. `RANDOM_WALK_CLOSED_MAP_WITH_DRIFT`: the agent moves in a preferential direction. There are two constants defining these directions: `SIM_TREND_ANGLE_1` and `SIM_TREND_ANGLE_2` ($3\pi/4$ and $5\pi/4$). The actual direction is generated as a normally distributed random value with means equals to these constants and `sad` = 0.1.

Enumerator

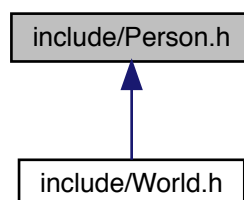
RANDOM_WALK_CLOSED_MAP	
RANDOM_WALK_CLOSED_MAP_WITH_DRIFT	
UNKNOWN	

7.22 include/Person.h File Reference

```
#include <MovableAgent.h>
#include <geos/geom/Point.h>
#include <string>
#include <unordered_map>
Include dependency graph for Person.h:
```



This graph shows which files directly or indirectly include this file:

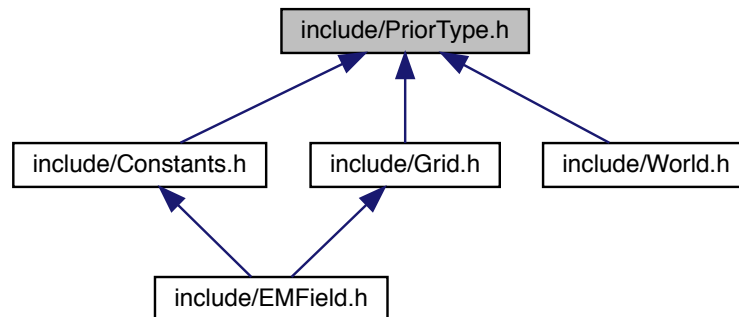


Classes

- class [Person](#)

7.23 include/PriorType.h File Reference

This graph shows which files directly or indirectly include this file:



Enumerations

- enum `PriorType` { `PriorType::UNIFORM`, `PriorType::REGISTER`, `PriorType::NETWORK` }

7.23.1 Enumeration Type Documentation

7.23.1.1 PriorType

```
enum PriorType [strong]
```

An enum class that enumerates the types of the prior used to computed the posterior localization probability. `UNIFORM`: the prior is an uniform probability, i.e. each object is equally located in each tile of the map. `REGISTER`: the prior probability is given by an administrative register. `NETWORK`: the prior probability is given by the mobile network - it is computed as the ratio between the signal quality given by [Antenna](#) `a` in tile `t` and the sum of the signal quality given by all antennas in all tiles of the grid.

Enumerator

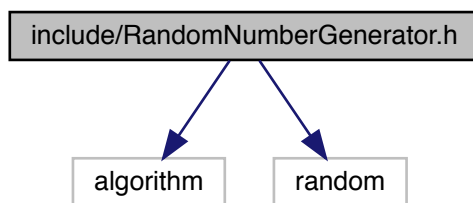
UNIFORM	
REGISTER	
NETWORK	

7.24 include/RandomNumberGenerator.h File Reference

```
#include <algorithm>
```

```
#include <random>
```

Include dependency graph for RandomNumberGenerator.h:



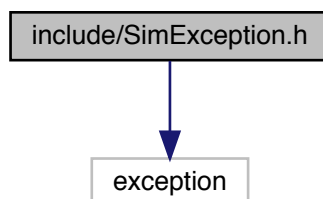
Classes

- class [RandomNumberGenerator](#)

7.25 include/SimException.h File Reference

```
#include <exception>
```

Include dependency graph for SimException.h:

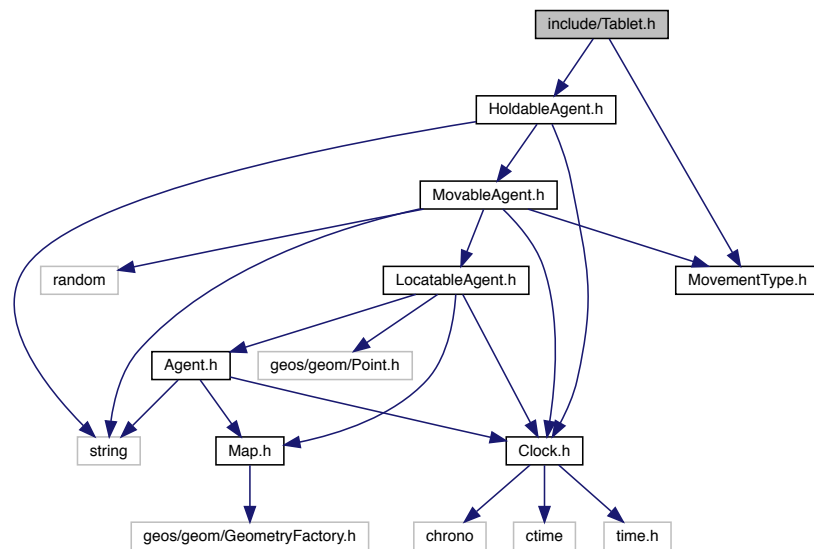


Classes

- class [SimException](#)

7.26 include/Tablet.h File Reference

```
#include <HoldableAgent.h>
#include <MovementType.h>
Include dependency graph for Tablet.h:
```

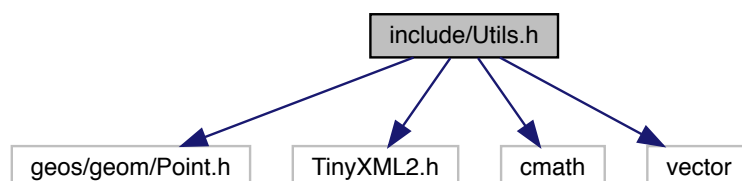


Classes

- class [Tablet](#)

7.27 include/Utils.h File Reference

```
#include <geos/geom/Point.h>
#include <TinyXML2.h>
#include <cmath>
#include <vector>
Include dependency graph for Utils.h:
```



- `utils`

- vector< Point * > [utils::generatePoints](#) (const Map *m, unsigned long n, double percentHome, unsigned seed)
- vector< Point * > [utils::generateFixedPoints](#) (const Map *m, unsigned long n, unsigned seed)
- void [utils::printPersonHeader](#) ()
- void [utils::printAntennaHeader](#) ()
- void [utils::printPhoneHeader](#) ()
- void [utils::printMobileOperatorHeader](#) ()
- double [utils::r2d](#) (double x)
- double [utils::d2r](#) (double x)
- XMLNode * [utils::getNode](#) (XMLElement *el, const char *name)
- XMLElement * [utils::getFirstChildElement](#) (XMLElement *el, const char *name) noexcept(false)
- double [utils::getValue](#) (XMLElement *el, const char *name, double default_value)
- int [utils::getValue](#) (XMLElement *el, const char *name, int default_value)
- unsigned long [utils::getValue](#) (XMLElement *el, const char *name, unsigned long default_value)
- const char * [utils::getValue](#) (XMLElement *el, const char *name, const char *default_value)
- double [utils::getValue](#) (XMLElement *el, const char *name)

- `const double utils::PI = std::atan(1.0) * 4`

```
#include <Antenna.h>
#include <Person.h>
#include <random>
#include <vector>
#include <map>
#include <AgentsCollection.h>
#include <Clock.h>
#include <MobilePhone.h>
#include <MovementType.h>
#include <TinyXML2.h>
#include <PriorType.h>
#include <MobileOperator.h>
```

Classes

- class [World](#)

Index

- CSVParser, [67](#)
- CSVParser, [67](#)
- CSVParser, [67](#)
 - Row, [144](#)
- CSVParser, [67](#)
- CSVParser, [68](#)
- CSVParser, [68](#)
- Row, [145](#)
- Agent, [19](#)
- AgentsCollection, [22](#)
- Antenna, [30](#)
- CSVParser, [64](#)
- Clock, [51](#)
- EMField, [69](#)
- Grid, [77](#)
- HoldableAgent, [87](#)
- ImmovableAgent, [95](#)
- LocatableAgent, [99](#)
- Map, [103](#)
- MobileOperator, [107](#)
- MobilePhone, [113](#)
- MovableAgent, [118](#)
- Person, [124](#)
- Row, [141](#)
- SimException, [146](#)
- Tablet, [149](#)
- World, [154](#)
- addAgent
 - AgentsCollection, [23](#)
- addAntenna
 - EMField, [70](#)
- addDevice
 - Person, [124](#)
- addGrid
 - Map, [103](#)
- addRow
 - CSVParser, [64](#)
- AgeDistributions
 - Person, [123](#)
- Agent, [17](#)
 - ~Agent, [19](#)
 - Agent, [18](#)
 - getClock, [19](#)
 - getId, [19](#)
 - getMap, [19](#)
 - getName, [20](#)
 - m_clock, [21](#)
 - m_id, [21](#)
 - m_map, [21](#)
 - operator==, [20](#)
 - setMap, [20](#)
 - toString, [21](#)
- AgentsCollection, [22](#)
 - ~AgentsCollection, [22](#)
 - addAgent, [23](#)
 - AgentsCollection, [22](#)
 - begin, [24](#)
 - deleteAgent, [24](#)
 - end, [24](#)
 - getAgent, [24](#)
 - getAgentListByType, [25](#)
 - m_agents, [26](#)
 - printAgents, [25](#)
 - size, [25](#)
- AgentsCollection.h
 - um_iterator, [165](#)
- ALREADY_ATTACHED_DEVICE
 - EventType.h, [173](#)
- alreadyRegistered
 - Antenna, [30](#)
- Antenna, [26](#)
 - ~Antenna, [30](#)

- alreadyRegistered, 30
- Antenna, 29, 30
- attachDevice, 31
- computePower, 31
- computeSignalMeasure, 32
- computeSignalQuality, 32
- computeSignalQualityDirectional, 33
- computeSignalQualityOmnidirectional, 33
- computeSignalStrength, 33
- computeSignalStrengthDirectional, 34
- computeSignalStrengthOmnidirectional, 34
- createMapping, 34
- dettachDevice, 34
- dumpCell, 34
- findSD, 35
- getAntennaOutputFileName, 35
- getCoverageArea, 35
- getCoverageAreaDirectional, 35
- getCoverageAreaOmnidirectional, 36
- getHandoverMechanism, 36
- getMaxConnections, 36
- getMin3db, 36
- getMNO, 36
- getName, 36
- getNumActiveConections, 37
- getPLE, 37
- getPower, 37
- getRmax, 37
- getSmid, 38
- getSmin, 38
- getSSteep, 38
- getType, 38
- m_azim_dB_Back, 43
- m_beam_H, 43
- m_beam_V, 43
- m_cell, 43
- m_devices, 43
- m_direction, 43
- m_elev_dB_Back, 44
- m_file, 44
- m_handoverMechanism, 44
- m_height, 44
- m_mapping_azim, 44
- m_mapping_elev, 44
- m_maxConnections, 44
- m_MNO, 44
- m_ple, 45
- m_power, 45
- m_Qmin, 45
- m_rmax, 45
- m_S0, 45
- m_sd_azim, 45
- m_sd_elev, 45
- m_Smid, 45
- m_Smin, 46
- m_SSSteep, 46
- m_tilt, 46
- m_type, 46
- norm_dBloss, 39
- normalizeAngle, 39
- projectToEPlane, 39
- registerEvent, 39
- S, 39
- S0, 40
- SDist, 40
- searchMin, 40
- setHandoverMechanism, 40
- setLocationWithElevation, 40
- setMaxConnections, 40
- setPLE, 41
- setPower, 41
- setSmid, 41
- setSSteep, 41
- setType, 42
- toString, 42
- tryRegisterDevice, 42
- ANTENNA_AZIM_DB_BACK
 - Constants, 56
- ANTENNA_BEAM_H
 - Constants, 56
- ANTENNA_BEAM_V
 - Constants, 56
- ANTENNA_DIRECTION
 - Constants, 56
- ANTENNA_ELEV_DB_BACK
 - Constants, 57
- ANTENNA_HEIGHT
 - Constants, 57
- ANTENNA_MAPPING_N
 - Constants, 57
- ANTENNA_MAX_CONNECTIONS
 - Constants, 57
- ANTENNA_MIN_3_DB
 - Constants, 57
- ANTENNA_POWER
 - Constants, 57
- ANTENNA_QMIN
 - Constants, 57
- ANTENNA_S_MID
 - Constants, 57
- ANTENNA_S_STEEP
 - Constants, 58
- ANTENNA_SMIN
 - Constants, 58
- ANTENNA_TILT
 - Constants, 58
- AntennaInfo, 46
 - AntennaInfo, 47
 - getAntennald, 47
 - getDeviceld, 48
 - getEventCode, 48
 - getTime, 48
 - getX, 48
 - getY, 48
 - m_antennald, 49
 - m_deviceld, 49

- m_eventCode, [50](#)
 - m_time, [50](#)
 - m_x, [50](#)
 - m_y, [50](#)
 - operator<, [49](#)
 - toString, [49](#)
- ANTENNAS_FILE_NAME
 - Constants, [58](#)
- AntennaType
 - AntennaType.h, [168](#)
- AntennaType.h
 - AntennaType, [168](#)
 - DIRECTIONAL, [168](#)
 - OMNIDIRECTIONAL, [168](#)
- ATT_FACTOR
 - Constants, [58](#)
- ATTACH_DEVICE
 - EventType.h, [173](#)
- attachDevice
 - Antenna, [31](#)
- begin
 - AgentsCollection, [24](#)
- Clock, [50](#)
 - ~Clock, [51](#)
 - Clock, [51](#)
 - getCurrentTime, [52](#)
 - getFinalTime, [52](#)
 - getIncrement, [52](#)
 - getInitialTime, [52](#)
 - m_currentTime, [54](#)
 - m_finalTime, [54](#)
 - m_increment, [54](#)
 - m_initialTime, [55](#)
 - realTime, [52](#)
 - reset, [53](#)
 - setCurrentTime, [53](#)
 - setFinalTime, [53](#)
 - setIncrement, [53](#)
 - setInitialTime, [54](#)
 - tick, [54](#)
- cmdOptionExists
 - InputParser, [96](#)
- columnCount
 - CSVParser, [64](#)
- computeMaxPower
 - EMField, [70](#)
- computeMaxQuality
 - EMField, [71](#)
- computeMaxStrength
 - EMField, [71](#)
- computePower
 - Antenna, [31](#)
- computeProbability
 - Grid, [77](#)
- computeSignalMeasure
 - Antenna, [32](#)
- computeSignalQuality
 - Antenna, [32](#)
- computeSignalQualityDirectional
 - Antenna, [33](#)
- computeSignalQualityOmnidirectional
 - Antenna, [33](#)
- computeSignalStrength
 - Antenna, [33](#)
- computeSignalStrengthDirectional
 - Antenna, [34](#)
- computeSignalStrengthOmnidirectional
 - Antenna, [34](#)
- computeTileCenters
 - Grid, [78](#)
- CONNECTION_TYPE
 - HoldableAgent, [86](#)
- connectionLikelihood
 - EMField, [72](#)
- connectionLikelihoodGrid
 - EMField, [72](#)
- Constants, [55](#)
 - ANTENNA_AZIM_DB_BACK, [56](#)
 - ANTENNA_BEAM_H, [56](#)
 - ANTENNA_BEAM_V, [56](#)
 - ANTENNA_DIRECTION, [56](#)
 - ANTENNA_ELEV_DB_BACK, [57](#)
 - ANTENNA_HEIGHT, [57](#)
 - ANTENNA_MAPPING_N, [57](#)
 - ANTENNA_MAX_CONNECTIONS, [57](#)
 - ANTENNA_MIN_3_DB, [57](#)
 - ANTENNA_POWER, [57](#)
 - ANTENNA_QMIN, [57](#)
 - ANTENNA_S_MID, [57](#)
 - ANTENNA_S_STEEP, [58](#)
 - ANTENNA_SMIN, [58](#)
 - ANTENNA_TILT, [58](#)
 - ANTENNAS_FILE_NAME, [58](#)
 - ATT_FACTOR, [58](#)
 - GRID_DIM_TILE_X, [58](#)
 - GRID_DIM_TILE_Y, [58](#)
 - GRID_FILE_NAME, [59](#)
 - GRID_X_ORIG, [59](#)
 - GRID_Y_ORIG, [59](#)
 - PERSONS_FILE_NAME, [59](#)
 - PHONE_CONNECTION_THRESHOLD, [59](#)
 - PHONE_POWER_THRESHOLD, [59](#)
 - PHONE_QUALITY_THRESHOLD, [60](#)
 - PHONE_STRENGTH_THRESHOLD, [60](#)
 - PRIOR_PROBABILITY, [60](#)
 - PROB_FILE_NAME_PREFIX, [60](#)
 - RANDOM_SEED, [60](#)
 - sep, [60](#)
 - SIM_DEFAULT_MNO_NAME, [60](#)
 - SIM_END_TIME, [61](#)
 - SIM_INCREMENT_TIME, [61](#)
 - SIM_INTERVAL_BETWEEN_STAYS, [61](#)
 - SIM_NO_ANTENNAS, [61](#)
 - SIM_NO_MOBILE_PHONES, [61](#)
 - SIM_NO_PERSONS, [61](#)

- SIM_NUM_MNO, 61
- SIM_PROB_MOBILE_PHONE, 61
- SIM_PROB_SECOND_MOBILE_PHONE, 62
- SIM_START_TIME, 62
- SIM_STAY_TIME, 62
- SIM_TREND_ANGLE_1, 62
- SIM_TREND_ANGLE_2, 62
- create_rectangle
 - Map, 104
- createMapping
 - Antenna, 34
- CSVParser, 62
 - _content, 67
 - _file, 67
 - _header, 67
 - _originalFile, 67
 - _sep, 68
 - _type, 68
 - ~CSVParser, 64
 - addRow, 64
 - columnCount, 64
 - CSVParser, 63
 - deleteRow, 65
 - getFileName, 65
 - getHeader, 65
 - getHeaderElement, 65
 - getRow, 66
 - m_header, 68
 - operator[], 66
 - parseContent, 66
 - parseHeader, 66
 - rowCount, 67
 - sync, 67
- CSVparser.hpp
 - DataType, 171
 - eFILE, 171
 - ePURE, 171
- d2r
 - utils, 9
- DataType
 - CSVparser.hpp, 171
- deleteAgent
 - AgentsCollection, 24
- deleteRow
 - CSVParser, 65
- DETACH_DEVICE
 - EventType.h, 173
- dettachDevice
 - Antenna, 34
- DIRECTIONAL
 - AntennaType.h, 168
- dumpCell
 - Antenna, 34
- dumpDevices
 - Person, 125
- dumpGrid
 - Grid, 78
- dumpLocation
 - LocatableAgent, 100
- eFILE
 - CSVparser.hpp, 171
- EMField, 68
 - ~EMField, 69
 - addAntenna, 70
 - computeMaxPower, 70
 - computeMaxQuality, 71
 - computeMaxStrength, 71
 - connectionLikelihood, 72
 - connectionLikelihoodGrid, 72
 - EMField, 70
 - getAntennaMin3DbArray, 72
 - getInRangeAntennas, 73
 - getSd, 73
 - instance, 73
 - isAntennaInRange, 73
 - m_antennaMin3DbArray, 75
 - m_antennas, 75
 - m_instance, 75
 - m_sd, 75
 - m_sumQuality, 75
 - operator=, 74
 - sumSignalQuality, 74
- end
 - AgentsCollection, 24
- ePURE
 - CSVparser.hpp, 171
- EventType
 - EventType.h, 173
- EventType.h
 - ALREADY_ATTACHED_DEVICE, 173
 - ATTACH_DEVICE, 173
 - DETACH_DEVICE, 173
 - EventType, 173
 - IN_RANGE_NOT_ATTACHED_DEVICE, 173
- FEMALE
 - Person, 124
- findSD
 - Antenna, 35
- Gender
 - Person, 123
- generateAntennas
 - World, 154
- generateBernoulliInt
 - RandomNumberGenerator, 132, 133
- generateBinomialInt
 - RandomNumberGenerator, 133
- generateExponentialDouble
 - RandomNumberGenerator, 134
- generateFixedPoints
 - utils, 10
- generateMobilePhones
 - World, 154
- generateNewLocation
 - Person, 125

- generateNormal2Double
 - RandomNumberGenerator, [134](#)
- generateNormalDouble
 - RandomNumberGenerator, [135](#)
- generatePoints
 - utils, [10](#)
- generatePopulation
 - World, [154](#)
- generateTruncatedNormalDouble
 - RandomNumberGenerator, [136](#)
- generateUniformDouble
 - RandomNumberGenerator, [136](#)
- generateUniformInt
 - RandomNumberGenerator, [137](#)
- getAge
 - Person, [125](#)
- getAgent
 - AgentsCollection, [24](#)
- getAgentListByType
 - AgentsCollection, [25](#)
- getAgents
 - World, [154](#)
- getAntennaCellsFile
 - MobileOperator, [107](#)
- getAntennald
 - AntennaInfo, [47](#)
- getAntennaMin3DbArray
 - EMField, [72](#)
- getAntennaOutputFileName
 - Antenna, [35](#)
- getAntennas
 - HoldableAgent, [87](#)
- getAntennasFilename
 - World, [155](#)
- getAvgIntervalBetweenStays
 - Person, [125](#)
- getAvgTimeStay
 - Person, [126](#)
- getBoundary
 - Map, [104](#)
- getClock
 - Agent, [19](#)
 - World, [155](#)
- getCmdOption
 - InputParser, [97](#)
- getConnectionThreshold
 - MobilePhone, [113](#)
- getConnectionType
 - World, [155](#)
- getCoverageArea
 - Antenna, [35](#)
- getCoverageAreaDirectional
 - Antenna, [35](#)
- getCoverageAreaOmnidirectional
 - Antenna, [36](#)
- getCurrentTime
 - Clock, [52](#)
- getDefaultConnectionThreshold
 - World, [155](#)
- getDeviceId
 - AntennaInfo, [48](#)
- getEventCode
 - AntennaInfo, [48](#)
- getFileName
 - CSVParser, [65](#)
- getFinalTime
 - Clock, [52](#)
- getFirstChildElement
 - utils, [11](#)
- getGender
 - Person, [126](#)
- getGlobalFactory
 - Map, [104](#)
- getGrid
 - Map, [104](#)
- getGridDimTileX
 - World, [156](#)
- getGridDimTileY
 - World, [156](#)
- getGridFilename
 - World, [156](#)
- getHandoverMechanism
 - Antenna, [36](#)
- getHeader
 - CSVParser, [65](#)
- getHeaderElement
 - CSVParser, [65](#)
- getHolder
 - HoldableAgent, [87](#)
- getId
 - Agent, [19](#)
- getIncrement
 - Clock, [52](#)
- getInitialTime
 - Clock, [52](#)
- getInRangeAntennas
 - EMField, [73](#)
- getLocation
 - LocatableAgent, [100](#)
- getMap
 - Agent, [19](#)
 - World, [156](#)
- getMaxConnections
 - Antenna, [36](#)
- getMin3db
 - Antenna, [36](#)
- getMNO
 - Antenna, [36](#)
- getMNOName
 - MobileOperator, [107](#)
- getMobileOperator
 - MobilePhone, [113](#)
- getName
 - Agent, [20](#)
 - Antenna, [36](#)
 - HoldableAgent, [88](#)

- ImmovableAgent, 95
- LocatableAgent, 100
- MobileOperator, 108
- MobilePhone, 113
- MovableAgent, 118
- Person, 126
- Tablet, 149
- getNode
 - utils, 11
- getNoTiles
 - Grid, 78
- getNoTilesX
 - Grid, 78
- getNoTilesY
 - Grid, 79
- getNumActiveConections
 - Antenna, 37
- getPersonsFilename
 - World, 157
- getPLE
 - Antenna, 37
- getPower
 - Antenna, 37
- getPrior
 - World, 157
- getProbFileNames
 - World, 157
- getProbMobilePhone
 - MobileOperator, 108
- getRmax
 - Antenna, 37
- getRow
 - CSVParser, 66
- getSd
 - EMField, 73
- getSignalFile
 - MobileOperator, 108
- getSmid
 - Antenna, 38
- getSmin
 - Antenna, 38
- getSpeed
 - MovableAgent, 118
- getSSteep
 - Antenna, 38
- getTileCenter
 - Grid, 79
- getTileCenters
 - Grid, 79
- getTileIndexX
 - Grid, 79, 80
- getTileIndexY
 - Grid, 80
- getTileNo
 - Grid, 80, 81
- getTime
 - AntennaInfo, 48
- getType
 - Antenna, 38
- getValue
 - Row, 142
 - utils, 11–13
- getX
 - AntennaInfo, 48
- getXOrigin
 - Grid, 81
- getXTileDim
 - Grid, 81
- getY
 - AntennaInfo, 48
- getYOrigin
 - Grid, 82
- getYTileDim
 - Grid, 82
- Grid, 75
 - ~Grid, 77
 - computeProbability, 77
 - computeTileCenters, 78
 - dumpGrid, 78
 - getNoTiles, 78
 - getNoTilesX, 78
 - getNoTilesY, 79
 - getTileCenter, 79
 - getTileCenters, 79
 - getTileIndexX, 79, 80
 - getTileIndexY, 80
 - getTileNo, 80, 81
 - getXOrigin, 81
 - getXTileDim, 81
 - getYOrigin, 82
 - getYTileDim, 82
 - Grid, 77
 - m_noTilesX, 83
 - m_noTilesY, 83
 - m_tileCenters, 83
 - m_xOrigin, 83
 - m_xTileDim, 83
 - m_yOrigin, 83
 - m_yTileDim, 83
 - toString, 82
 - useNetworkPrior, 82
 - useUniformPrior, 82
- GRID_DIM_TILE_X
 - Constants, 58
- GRID_DIM_TILE_Y
 - Constants, 58
- GRID_FILE_NAME
 - Constants, 59
- GRID_X_ORIG
 - Constants, 59
- GRID_Y_ORIG
 - Constants, 59
- hasDevices
 - Person, 126
- HoldableAgent, 84
 - ~HoldableAgent, 87

- CONNECTION_TYPE, 86
- getAntennas, 87
- getHolder, 87
- getName, 88
- HoldableAgent, 86, 87
- isConnected, 88
- m_antennas, 90
- m_holder, 90
- setHolder, 88
- setLocation, 89
- toString, 89
- tryConnect, 89
- UNKNOWN, 86
- USING_POWER, 86
- USING_SIGNAL_QUALITY, 86
- USING_SIGNAL_STRENGTH, 86
- IDGenerator, 90
 - IDGenerator, 91
 - instance, 91
 - m_id, 92
 - m_instance, 92
 - next, 91
- ImmovableAgent, 92
 - ~ImmovableAgent, 95
 - getName, 95
 - ImmovableAgent, 93
 - toString, 95
- IN_RANGE_NOT_ATTACHED_DEVICE
 - EventType.h, 173
- include/Agent.h, 163
- include/AgentsCollection.h, 164
- include/Antenna.h, 165
- include/AntennaInfo.h, 166
- include/AntennaType.h, 168
- include/Clock.h, 169
- include/Constants.h, 170
- include/CSVparser.hpp, 170
- include/EMField.h, 171
- include/EventType.h, 172
- include/Grid.h, 173
- include/HoldableAgent.h, 174
- include/IDGenerator.h, 176
- include/ImmovableAgent.h, 176
- include/InputParser.h, 177
- include/LocatableAgent.h, 178
- include/Map.h, 179
- include/MobileOperator.h, 180
- include/MobilePhone.h, 181
- include/MovableAgent.h, 182
- include/MovementType.h, 184
- include/Person.h, 185
- include/PriorType.h, 186
- include/RandomNumberGenerator.h, 186
- include/SimException.h, 187
- include/Tablet.h, 188
- include/Utils.h, 188
- include/World.h, 189
- InputParser, 96
 - cmdOptionExists, 96
 - getCmdOption, 97
 - InputParser, 96
 - tokens, 97
- instance
 - EMField, 73
 - IDGenerator, 91
 - RandomNumberGenerator, 138
- isAntennaInRange
 - EMField, 73
- isConnected
 - HoldableAgent, 88
- LocatableAgent, 97
 - ~LocatableAgent, 99
 - dumpLocation, 100
 - getLocation, 100
 - getName, 100
 - LocatableAgent, 99
 - m_location, 101
 - setLocation, 100
 - toString, 101
- m_age
 - Person, 129
- m_agents
 - AgentsCollection, 26
- m_agentsCollection
 - World, 159
- m_antennaCellsFileName
 - MobileOperator, 109
- m_antennald
 - AntennaInfo, 49
- m_antennaMin3DbArray
 - EMField, 75
- m_antennas
 - EMField, 75
 - HoldableAgent, 90
- m_antennasFilename
 - World, 159
- m_avgIntervalBetweenStays
 - Person, 129
- m_avgTimeStay
 - Person, 129
- m_azim_dB_Back
 - Antenna, 43
- m_beam_H
 - Antenna, 43
- m_beam_V
 - Antenna, 43
- m_bernoulli_distribution
 - RandomNumberGenerator, 139
- m_binomial_distribution
 - RandomNumberGenerator, 139
- m_boundary
 - Map, 105
- m_cell
 - Antenna, 43
- m_changeDirection

- Person, 129
- m_clock
 - Agent, 21
 - World, 160
- m_connectedTo
 - MobilePhone, 115
- m_connThreshold
 - World, 160
- m_connType
 - MobilePhone, 115
 - World, 160
- m_currentTime
 - Clock, 54
- m_deviceId
 - AntennaInfo, 49
- m_devices
 - Antenna, 43
- m_direction
 - Antenna, 43
- m_elev_dB_Back
 - Antenna, 44
- m_endTime
 - World, 160
- m_eventCode
 - AntennaInfo, 50
- m_exponential_double_distribution
 - RandomNumberGenerator, 139
- m_file
 - Antenna, 44
- m_finalTime
 - Clock, 54
- m_gender
 - Person, 129
- m_generator
 - RandomNumberGenerator, 140
- m_globalFactory
 - Map, 105
- m_grid
 - Map, 105
- m_GridDimTileX
 - World, 160
- m_GridDimTileY
 - World, 160
- m_gridFilename
 - World, 160
- m_handoverMechanism
 - Antenna, 44
- m_header
 - CSVParser, 68
- m_height
 - Antenna, 44
- m_holder
 - HoldableAgent, 90
- m_id
 - Agent, 21
 - IDGenerator, 92
- m_idDevices
 - Person, 129
- m_increment
 - Clock, 54
- m_initialTime
 - Clock, 55
- m_instance
 - EMField, 75
 - IDGenerator, 92
 - RandomNumberGenerator, 140
- m_intervalBetweenStays
 - World, 160
- m_location
 - LocatableAgent, 101
- m_map
 - Agent, 21
 - World, 161
- m_mapping_azim
 - Antenna, 44
- m_mapping_elev
 - Antenna, 44
- m_maxConnections
 - Antenna, 44
- m_MNO
 - Antenna, 44
- m_mno
 - MobilePhone, 115
- m_mvType
 - World, 161
- m_name
 - MobileOperator, 109
- m_nextStay
 - Person, 129
- m_normal_double_distribution
 - RandomNumberGenerator, 140
- m_noTilesX
 - Grid, 83
- m_noTilesY
 - Grid, 83
- m_personsFilename
 - World, 161
- m_ple
 - Antenna, 45
- m_power
 - Antenna, 45
- m_prior
 - World, 161
- m_probFileNames
 - World, 161
- m_probMobilePhone
 - MobileOperator, 109
- m_probSecMobilePhone
 - World, 161
- m_Qmin
 - Antenna, 45
- m_rmax
 - Antenna, 45
- m_S0
 - Antenna, 45
- m_sd

- EMField, 75
- m_sd_azim
 - Antenna, 45
- m_sd_elev
 - Antenna, 45
- m_seed
 - World, 161
- m_signalMeasureFileName
 - MobileOperator, 109
- m_Smid
 - Antenna, 45
- m_Smin
 - Antenna, 46
- m_speed
 - MovableAgent, 120
- m_SSteep
 - Antenna, 46
- m_startTime
 - World, 161
- m_stay
 - World, 162
- m_sumQuality
 - EMField, 75
- m_threshold
 - MobilePhone, 115
- m_tileCenters
 - Grid, 83
- m_tilt
 - Antenna, 46
- m_time
 - AntennaInfo, 50
- m_timeIncrement
 - World, 162
- m_timeStay
 - Person, 129
- m_type
 - Antenna, 46
- m_unif_double_distribution
 - RandomNumberGenerator, 140
- m_unif_int_distribution
 - RandomNumberGenerator, 140
- m_x
 - AntennaInfo, 50
- m_xOrigin
 - Grid, 83
- m_xTileDim
 - Grid, 83
- m_y
 - AntennaInfo, 50
- m_yOrigin
 - Grid, 83
- m_yTileDim
 - Grid, 83
- MALE
 - Person, 124
- Map, 101
 - ~Map, 103
 - addGrid, 103
 - create_rectangle, 104
 - getBoundary, 104
 - getGlobalFactory, 104
 - getGrid, 104
 - m_boundary, 105
 - m_globalFactory, 105
 - m_grid, 105
 - Map, 102, 103
 - setBoundary, 104
- MobileOperator, 105
 - ~MobileOperator, 107
 - getAntennaCellsFile, 107
 - getMNOName, 107
 - getName, 108
 - getProbMobilePhone, 108
 - getSignalFile, 108
 - m_antennaCellsFileName, 109
 - m_name, 109
 - m_probMobilePhone, 109
 - m_signalMeasureFileName, 109
 - MobileOperator, 107
 - toString, 108
- MobilePhone, 110
 - ~MobilePhone, 113
 - getConnectionThreshold, 113
 - getMobileOperator, 113
 - getName, 113
 - m_connectedTo, 115
 - m_connType, 115
 - m_mno, 115
 - m_threshold, 115
 - MobilePhone, 112
 - move, 113
 - setMobileOperator, 114
 - toString, 114
 - tryConnect, 114
- MovableAgent, 116
 - ~MovableAgent, 118
 - getName, 118
 - getSpeed, 118
 - m_speed, 120
 - MovableAgent, 118
 - move, 119
 - setSpeed, 119
 - toString, 120
- move
 - MobilePhone, 113
 - MovableAgent, 119
 - Person, 127
 - Tablet, 149
- MovementType
 - MovementType.h, 184
- MovementType.h
 - MovementType, 184
 - RANDOM_WALK_CLOSED_MAP, 185
 - RANDOM_WALK_CLOSED_MAP_WITH_DRIFT, 185
 - UNKNOWN, 185

- msg
 - SimException, 146
- NETWORK
 - PriorType.h, 186
- next
 - IDGenerator, 91
- norm_dBLoss
 - Antenna, 39
- NORMAL
 - Person, 123
- normal_pdf
 - RandomNumberGenerator, 138
- normalizeAngle
 - Antenna, 39
- OMNIDIRECTIONAL
 - AntennaType.h, 168
- operator<
 - AntennaInfo, 49
- operator<<
 - Row, 144
- operator=
 - EMField, 74
 - RandomNumberGenerator, 139
- operator==
 - Agent, 20
- operator[]
 - CSVParser, 66
 - Row, 142
- parseAntennas
 - World, 157
- parseContent
 - CSVParser, 66
- parseHeader
 - CSVParser, 66
- parsePersons
 - World, 158
- parseProbabilities
 - World, 158
- parseSimulationFile
 - World, 158
- Person, 120
 - ~Person, 124
 - addDevice, 124
 - AgeDistributions, 123
 - dumpDevices, 125
 - FEMALE, 124
 - Gender, 123
 - generateNewLocation, 125
 - getAge, 125
 - getAvgIntervalBetweenStays, 125
 - getAvgTimeStay, 126
 - getGender, 126
 - getName, 126
 - hasDevices, 126
 - m_age, 129
 - m_avgIntervalBetweenStays, 129
 - m_avgTimeStay, 129
 - m_changeDirection, 129
 - m_gender, 129
 - m_idDevices, 129
 - m_nextStay, 129
 - m_timeStay, 129
 - MALE, 124
 - move, 127
 - NORMAL, 123
 - Person, 124
 - randomWalkClosedMap, 127
 - randomWalkClosedMapDrift, 127
 - setAge, 128
 - setLocation, 128
 - setNewLocation, 128
 - toString, 128
 - UNIFORM, 123
- PERSONS_FILE_NAME
 - Constants, 59
- PHONE_CONNECTION_THRESHOLD
 - Constants, 59
- PHONE_POWER_THRESHOLD
 - Constants, 59
- PHONE_QUALITY_THRESHOLD
 - Constants, 60
- PHONE_STRENGTH_THRESHOLD
 - Constants, 60
- PI
 - utils, 15
- printAgents
 - AgentsCollection, 25
- printAntennaHeader
 - utils, 14
- printMobileOperatorHeader
 - utils, 14
- printPersonHeader
 - utils, 14
- printPhoneHeader
 - utils, 14
- PRIOR_PROBABILITY
 - Constants, 60
- PriorType
 - PriorType.h, 186
- PriorType.h
 - NETWORK, 186
 - PriorType, 186
 - REGISTER, 186
 - UNIFORM, 186
- PROB_FILE_NAME_PREFIX
 - Constants, 60
- projectToEPlane
 - Antenna, 39
- push
 - Row, 143
- r2d
 - utils, 14
- RANDOM_SEED
 - Constants, 60

- RANDOM_WALK_CLOSED_MAP
 - MovementType.h, [185](#)
- RANDOM_WALK_CLOSED_MAP_WITH_DRIFT
 - MovementType.h, [185](#)
- RandomNumberGenerator, [130](#)
 - generateBernoulliInt, [132](#), [133](#)
 - generateBinomialInt, [133](#)
 - generateExponentialDouble, [134](#)
 - generateNormal2Double, [134](#)
 - generateNormalDouble, [135](#)
 - generateTruncatedNormalDouble, [136](#)
 - generateUniformDouble, [136](#)
 - generateUniformInt, [137](#)
 - instance, [138](#)
 - m_bernoulli_distribution, [139](#)
 - m_binomial_distribution, [139](#)
 - m_exponential_double_distribution, [139](#)
 - m_generator, [140](#)
 - m_instance, [140](#)
 - m_normal_double_distribution, [140](#)
 - m_unif_double_distribution, [140](#)
 - m_unif_int_distribution, [140](#)
 - normal_pdf, [138](#)
 - operator=, [139](#)
 - RandomNumberGenerator, [131](#)
 - setSeed, [139](#)
- randomWalkClosedMap
 - Person, [127](#)
- randomWalkClosedMapDrift
 - Person, [127](#)
- realTime
 - Clock, [52](#)
- REGISTER
 - PriorType.h, [186](#)
- registerEvent
 - Antenna, [39](#)
- reset
 - Clock, [53](#)
- Row, [140](#)
 - _header, [144](#)
 - _values, [145](#)
 - ~Row, [141](#)
 - getValue, [142](#)
 - operator<<, [144](#)
 - operator[], [142](#)
 - push, [143](#)
 - Row, [141](#)
 - set, [143](#)
 - size, [143](#)
- rowCount
 - CSVParser, [67](#)
- runSimulation
 - World, [158](#)
- S
 - Antenna, [39](#)
- S0
 - Antenna, [40](#)
- SDist
 - Antenna, [40](#)
 - searchMin
 - Antenna, [40](#)
 - sep
 - Constants, [60](#)
 - set
 - Row, [143](#)
 - setAge
 - Person, [128](#)
 - setAgents
 - World, [158](#)
 - setBoundary
 - Map, [104](#)
 - setClock
 - World, [159](#)
 - setCurrentTime
 - Clock, [53](#)
 - setFinalTime
 - Clock, [53](#)
 - setHandoverMechanism
 - Antenna, [40](#)
 - setHolder
 - HoldableAgent, [88](#)
 - setIncrement
 - Clock, [53](#)
 - setInitialTime
 - Clock, [54](#)
 - setLocation
 - HoldableAgent, [89](#)
 - LocatableAgent, [100](#)
 - Person, [128](#)
 - setLocationWithElevation
 - Antenna, [40](#)
 - setMap
 - Agent, [20](#)
 - World, [159](#)
 - setMaxConnections
 - Antenna, [40](#)
 - setMobileOperator
 - MobilePhone, [114](#)
 - setNewLocation
 - Person, [128](#)
 - setPLE
 - Antenna, [41](#)
 - setPower
 - Antenna, [41](#)
 - setSeed
 - RandomNumberGenerator, [139](#)
 - setSmid
 - Antenna, [41](#)
 - setSpeed
 - MovableAgent, [119](#)
 - setSSteep
 - Antenna, [41](#)
 - setType
 - Antenna, [42](#)
 - SIM_DEFAULT_MNO_NAME
 - Constants, [60](#)

- SIM_END_TIME
 - Constants, 61
- SIM_INCREMENT_TIME
 - Constants, 61
- SIM_INTERVAL_BETWEEN_STAYS
 - Constants, 61
- SIM_NO_ANTENNAS
 - Constants, 61
- SIM_NO_MOBILE_PHONES
 - Constants, 61
- SIM_NO_PERSONS
 - Constants, 61
- SIM_NUM_MNO
 - Constants, 61
- SIM_PROB_MOBILE_PHONE
 - Constants, 61
- SIM_PROB_SECOND_MOBILE_PHONE
 - Constants, 62
- SIM_START_TIME
 - Constants, 62
- SIM_STAY_TIME
 - Constants, 62
- SIM_TREND_ANGLE_1
 - Constants, 62
- SIM_TREND_ANGLE_2
 - Constants, 62
- SimException, 145
 - ~SimException, 146
 - msg, 146
 - SimException, 146
 - what, 146
- size
 - AgentsCollection, 25
 - Row, 143
- sumSignalQuality
 - EMField, 74
- sync
 - CSVParser, 67
- Tablet, 147
 - ~Tablet, 149
 - getName, 149
 - move, 149
 - Tablet, 148
 - toString, 150
 - tryConnect, 150
- tick
 - Clock, 54
- tokens
 - InputParser, 97
- toString
 - Agent, 21
 - Antenna, 42
 - AntennaInfo, 49
 - Grid, 82
 - HoldableAgent, 89
 - ImmovableAgent, 95
 - LocatableAgent, 101
 - MobileOperator, 108
 - MobilePhone, 114
 - MovableAgent, 120
 - Person, 128
 - Tablet, 150
- tryConnect
 - HoldableAgent, 89
 - MobilePhone, 114
 - Tablet, 150
- tryRegisterDevice
 - Antenna, 42
- um_iterator
 - AgentsCollection.h, 165
- UNIFORM
 - Person, 123
 - PriorType.h, 186
- UNKNOWN
 - HoldableAgent, 86
 - MovementType.h, 185
- useNetworkPrior
 - Grid, 82
- useUniformPrior
 - Grid, 82
- USING_POWER
 - HoldableAgent, 86
- USING_SIGNAL_QUALITY
 - HoldableAgent, 86
- USING_SIGNAL_STRENGTH
 - HoldableAgent, 86
- utils, 9
 - d2r, 9
 - generateFixedPoints, 10
 - generatePoints, 10
 - getFirstChildElement, 11
 - getNode, 11
 - getValue, 11–13
 - PI, 15
 - printAntennaHeader, 14
 - printMobileOperatorHeader, 14
 - printPersonHeader, 14
 - printPhoneHeader, 14
 - r2d, 14
- what
 - SimException, 146
- whichMNO
 - World, 159
- World, 151
 - ~World, 154
 - generateAntennas, 154
 - generateMobilePhones, 154
 - generatePopulation, 154
 - getAgents, 154
 - getAntennasFilename, 155
 - getClock, 155
 - getConnectionType, 155
 - getDefaultConnectionThreshold, 155
 - getGridDimTileX, 156
 - getGridDimTileY, 156

- getGridFilename, [156](#)
- getMap, [156](#)
- getPersonsFilename, [157](#)
- getPrior, [157](#)
- getProbFileNames, [157](#)
- m_agentsCollection, [159](#)
- m_antennasFilename, [159](#)
- m_clock, [160](#)
- m_connThreshold, [160](#)
- m_connType, [160](#)
- m_endTime, [160](#)
- m_GridDimTileX, [160](#)
- m_GridDimTileY, [160](#)
- m_gridFilename, [160](#)
- m_intervalBetweenStays, [160](#)
- m_map, [161](#)
- m_mvType, [161](#)
- m_personsFilename, [161](#)
- m_prior, [161](#)
- m_probFileNames, [161](#)
- m_probSecMobilePhone, [161](#)
- m_seed, [161](#)
- m_startTime, [161](#)
- m_stay, [162](#)
- m_timeIncrement, [162](#)
- parseAntennas, [157](#)
- parsePersons, [158](#)
- parseProbabilities, [158](#)
- parseSimulationFile, [158](#)
- runSimulation, [158](#)
- setAgents, [158](#)
- setClock, [159](#)
- setMap, [159](#)
- whichMNO, [159](#)
- World, [153](#)