

No-MASS Documentation

Generated by Doxygen 1.8.13

Contents

1	No-MASS	1
2	Implementation	3
3	Dependencies Of NoMASS	5
4	Using No-MASS	7
5	Simulation Configuration	9
6	IDF File	17
7	Functional Mockup Unit (FMU)	19
8	Model Description	25
9	Scripts For NoMASS	27
10	EnergyPlus Examples	29
11	Class Diagram	35
12	Compiling No-MASS	37
13	Hierarchical Index	39
13.1	Class Hierarchy	39
14	Class Index	41
14.1	Class List	41

15 Class Documentation	45
15.1 Agent Class Reference	45
15.1.1 Detailed Description	47
15.1.2 Constructor & Destructor Documentation	48
15.1.2.1 Agent()	48
15.1.3 Member Function Documentation	48
15.1.3.1 getBuildingID()	48
15.1.3.2 getId()	48
15.1.3.3 postprocess()	48
15.1.3.4 postTimeStep()	48
15.1.3.5 preprocess()	49
15.1.3.6 setBuildingID()	49
15.1.3.7 setId()	49
15.1.3.8 setIdString()	50
15.1.3.9 setup()	50
15.1.3.10 step()	50
15.1.4 Member Data Documentation	50
15.1.4.1 buildingID	50
15.1.4.2 id	50
15.1.4.3 idString	51
15.2 Appliance Class Reference	51
15.2.1 Detailed Description	55
15.2.2 Constructor & Destructor Documentation	55
15.2.2.1 Appliance()	55
15.2.3 Member Function Documentation	56
15.2.3.1 beforeClear()	56
15.2.3.2 calculateHourOfDay()	56
15.2.3.3 clear()	57
15.2.3.4 getGridPower()	57
15.2.3.5 getGridReceived()	57

15.2.3.6 getGridReceivedCost()	57
15.2.3.7 getGridSupply()	57
15.2.3.8 getGridSupplyLeft()	58
15.2.3.9 getLocalPower()	58
15.2.3.10 getLocalReceived()	58
15.2.3.11 getLocalReceivedCost()	58
15.2.3.12 getLocalSupply()	58
15.2.3.13 getLocalSupplyLeft()	58
15.2.3.14 getNeighbourhoodPower()	59
15.2.3.15 getNeighbourhoodReceived()	59
15.2.3.16 getNeighbourhoodReceivedCost()	59
15.2.3.17 getNeighbourhoodSupply()	59
15.2.3.18 getNeighbourhoodSupplyLeft()	59
15.2.3.19 getPower()	60
15.2.3.20 getPriority()	60
15.2.3.21 getReceived()	60
15.2.3.22 getReceivedCost()	61
15.2.3.23 getSupply()	61
15.2.3.24 getSupplyCost()	62
15.2.3.25 getSupplyLeft()	62
15.2.3.26 hasActivities()	62
15.2.3.27 isGlobal()	63
15.2.3.28 isLocal()	63
15.2.3.29 save()	63
15.2.3.30 saveGlobal()	63
15.2.3.31 saveGlobalCalculate()	64
15.2.3.32 saveLocal()	64
15.2.3.33 saveLocalCalculate()	65
15.2.3.34 saveNeighbourhood()	65
15.2.3.35 saveNeighbourhoodCalculate()	65

15.2.3.36 setActivities()	66
15.2.3.37 setGlobal()	66
15.2.3.38 setHourlyPriority()	66
15.2.3.39 setHourlyCost()	67
15.2.3.40 setLocal()	68
15.2.3.41 setPower()	68
15.2.3.42 setReceived()	69
15.2.3.43 setReceivedCost()	69
15.2.3.44 setSupply()	69
15.2.3.45 setSupplyCost()	70
15.2.3.46 setSupplyLeft()	70
15.2.3.47 setup()	71
15.2.3.48 setupSave()	71
15.2.4 Member Data Documentation	71
15.2.4.1 Activities	72
15.2.4.2 datastoreGridIDCost	72
15.2.4.3 datastoreGridIDReceived	72
15.2.4.4 datastoreGridIDRequested	72
15.2.4.5 datastoreGridIDSupplied	72
15.2.4.6 datastoreGridIDSuppliedCost	72
15.2.4.7 datastoreIDCost	73
15.2.4.8 datastoreIDReceived	73
15.2.4.9 datastoreIDRequested	73
15.2.4.10 datastoreIDSupplied	73
15.2.4.11 datastoreIDSuppliedCost	73
15.2.4.12 datastoreLocalIDCost	73
15.2.4.13 datastoreLocalIDReceived	74
15.2.4.14 datastoreLocalIDRequested	74
15.2.4.15 datastoreLocalIDSupplied	74
15.2.4.16 datastoreLocalIDSuppliedCost	74

15.2.4.17 datastoreNeighbourhoodIDCost	74
15.2.4.18 datastoreNeighbourhoodIDReceived	74
15.2.4.19 datastoreNeighbourhoodIDRequested	75
15.2.4.20 datastoreNeighbourhoodIDSupplied	75
15.2.4.21 datastoreNeighbourhoodIDSuppliedCost	75
15.2.4.22 global	75
15.2.4.23 hourlyCost	75
15.2.4.24 hourlyPriority	75
15.2.4.25 local	76
15.2.4.26 match	76
15.2.4.27 parameters	76
15.2.4.28 parametersGrid	76
15.2.4.29 parametersLocal	76
15.2.4.30 parametersNeighbourhood	76
15.3 Appliance_Battery Class Reference	77
15.3.1 Detailed Description	80
15.3.2 Constructor & Destructor Documentation	80
15.3.2.1 Appliance_Battery()	80
15.3.3 Member Function Documentation	80
15.3.3.1 AddCost()	80
15.3.3.2 calculateDeltaE()	80
15.3.3.3 calculateStateOfCharge()	81
15.3.3.4 calculateSupply()	81
15.3.3.5 clear()	82
15.3.3.6 doAction()	83
15.3.3.7 energy_calc()	83
15.3.3.8 get_charge_delta()	84
15.3.3.9 get_new_SOC_charge()	85
15.3.3.10 get_new_SOC_discharge()	85
15.3.3.11 getStateOfCharge()	86

15.3.3.12 postprocess()	86
15.3.3.13 powerAvailable()	87
15.3.3.14 rewardFunction()	87
15.3.3.15 saveGlobalCalculate()	88
15.3.3.16 saveNeighbourhoodCalculate()	88
15.3.3.17 setBatteryNeighbourhoodCharge()	88
15.3.3.18 setBatteryNeighbourhoodDischarge()	88
15.3.3.19 setPowerShortage()	89
15.3.3.20 setup()	89
15.3.3.21 setupModel()	90
15.3.3.22 step()	90
15.3.3.23 stepNeighbourhood()	91
15.3.4 Member Data Documentation	91
15.3.4.1 action	91
15.3.4.2 BatteryDeltaT	91
15.3.4.3 batteryNeighbourhoodCharge	91
15.3.4.4 batteryNeighbourhoodDischarge	92
15.3.4.5 capacity	92
15.3.4.6 chargeRate	92
15.3.4.7 cost	92
15.3.4.8 datastoreIDstateOfCharge	92
15.3.4.9 dischargeRate	92
15.3.4.10 efficiency	93
15.3.4.11 mostShortage	93
15.3.4.12 powerShortage	93
15.3.4.13 previousHourOfDay	93
15.3.4.14 qLearning	93
15.3.4.15 stateOfCharge	93
15.3.4.16 sumShort	94
15.3.4.17 sumSupply	94

15.4 Appliance_Battery_GridCost_Reward Class Reference	94
15.4.1 Detailed Description	97
15.4.2 Constructor & Destructor Documentation	97
15.4.2.1 Appliance_Battery_GridCost_Reward()	97
15.4.3 Member Function Documentation	97
15.4.3.1 rewardFunction()	97
15.5 Appliance_FMI Class Reference	98
15.5.1 Detailed Description	100
15.5.2 Constructor & Destructor Documentation	100
15.5.2.1 Appliance_FMI()	100
15.5.3 Member Function Documentation	100
15.5.3.1 setFМИVariableName()	100
15.5.3.2 setup()	101
15.5.3.3 step()	101
15.6 Appliance_Generic_CSV Class Reference	102
15.6.1 Detailed Description	104
15.6.2 Constructor & Destructor Documentation	104
15.6.2.1 Appliance_Generic_CSV()	104
15.6.3 Member Function Documentation	104
15.6.3.1 setFileDemand()	104
15.6.3.2 setFileSupply()	105
15.6.3.3 setup()	105
15.6.3.4 step()	106
15.7 Appliance_Group< T > Class Template Reference	107
15.7.1 Detailed Description	110
15.7.2 Constructor & Destructor Documentation	110
15.7.2.1 Appliance_Group()	110
15.7.3 Member Function Documentation	110
15.7.3.1 addGlobalContactsTo()	111
15.7.3.2 clear()	111

15.7.3.3	getApplianceAt()	111
15.7.3.4	getPower()	112
15.7.3.5	getReceived()	112
15.7.3.6	getReceivedCost()	113
15.7.3.7	getSupply()	113
15.7.3.8	getSupplyCost()	113
15.7.3.9	getSupplyLeft()	114
15.7.3.10	globalNegotiation()	114
15.7.3.11	hasActivities()	114
15.7.3.12	localNegotiation()	115
15.7.3.13	negotiationApp()	115
15.7.3.14	negotiationAppGlobal()	116
15.7.3.15	neighbourhoodNegotiation()	116
15.7.3.16	postprocess()	117
15.7.3.17	sendCondition()	117
15.7.3.18	sendContractGlobal()	118
15.7.3.19	sendContractLocal()	118
15.7.3.20	setIDString()	119
15.7.3.21	setup()	119
15.7.3.22	setupSave()	119
15.7.3.23	shuffleAppliances()	120
15.7.3.24	step()	120
15.7.3.25	stepApp()	121
15.7.4	Member Data Documentation	121
15.7.4.1	appliances	121
15.7.4.2	datastoreIDCost	121
15.7.4.3	datastoreIDReceived	121
15.7.4.4	datastoreIDRequested	122
15.7.4.5	datastoreIDSupplied	122
15.7.4.6	datastoreIDSuppliedCost	122

15.7.4.7	globalContracts	122
15.7.4.8	idString	122
15.8	Appliance_Group_Battery< T > Class Template Reference	123
15.8.1	Detailed Description	125
15.8.2	Constructor & Destructor Documentation	125
15.8.2.1	Appliance_Group_Battery()	125
15.8.3	Member Function Documentation	125
15.8.3.1	getPowerShortage()	125
15.8.3.2	globalCost()	126
15.8.3.3	negotiationApp()	126
15.8.3.4	negotiationAppGlobal()	126
15.8.3.5	neighbourhoodNegotiationBattery()	127
15.8.3.6	sendCondition()	127
15.8.3.7	setPowerShortage()	127
15.8.3.8	stepApp()	128
15.9	Appliance_Large Class Reference	128
15.9.1	Detailed Description	131
15.9.2	Constructor & Destructor Documentation	131
15.9.2.1	Appliance_Large()	131
15.9.3	Member Function Documentation	131
15.9.3.1	isOn()	132
15.9.3.2	setFile()	132
15.9.3.3	setup()	133
15.9.3.4	setupModel()	133
15.9.3.5	step()	134
15.9.4	Member Data Documentation	134
15.9.4.1	file	134
15.9.4.2	model	135
15.9.4.3	profileCSV	135
15.10	Appliance_Large_CSV Class Reference	135

15.10.1 Detailed Description	138
15.10.2 Constructor & Destructor Documentation	138
15.10.2.1 Appliance_Large_CSV()	138
15.10.3 Member Function Documentation	138
15.10.3.1 setupModel()	138
15.10.3.2 step()	139
15.10.4 Member Data Documentation	139
15.10.4.1 count	139
15.10.4.2 running	139
15.11Appliance_Large_Learning Class Reference	140
15.11.1 Detailed Description	142
15.11.2 Constructor & Destructor Documentation	142
15.11.2.1 Appliance_Large_Learning()	142
15.11.3 Member Function Documentation	142
15.11.3.1 addToCost()	142
15.11.3.2 calculateProfile()	143
15.11.3.3 getPowerAt()	144
15.11.3.4 getRequiredTime()	144
15.11.3.5 isModelOn()	145
15.11.3.6 postprocess()	145
15.11.3.7 setHourlyTimeRequired()	146
15.11.3.8 setup()	146
15.11.3.9 step()	147
15.11.4 Member Data Documentation	147
15.11.4.1 powerProfile	147
15.12Appliance_Large_Learning_CSV Class Reference	148
15.12.1 Detailed Description	150
15.12.2 Constructor & Destructor Documentation	150
15.12.2.1 Appliance_Large_Learning_CSV()	150
15.12.3 Member Function Documentation	150

15.12.3.1 calculateProfile()	150
15.12.3.2 setupModel()	151
15.13Appliance_Small Class Reference	151
15.13.1 Detailed Description	154
15.13.2 Constructor & Destructor Documentation	154
15.13.2.1 Appliance_Small()	154
15.13.3 Member Function Documentation	154
15.13.3.1 setup()	154
15.13.3.2 step()	155
15.14ApplianceParameters Struct Reference	156
15.14.1 Detailed Description	156
15.14.2 Member Data Documentation	156
15.14.2.1 power	157
15.14.2.2 received	157
15.14.2.3 receivedCost	157
15.14.2.4 suppliedLeft	157
15.14.2.5 supply	157
15.14.2.6 supplyCost	158
15.15Building Class Reference	158
15.15.1 Detailed Description	159
15.15.2 Constructor & Destructor Documentation	159
15.15.2.1 Building()	159
15.15.3 Member Function Documentation	159
15.15.3.1 addContactsTo()	160
15.15.3.2 decisionBoolean()	160
15.15.3.3 decisionDoubleVec()	161
15.15.3.4 getID()	162
15.15.3.5 getPower()	162
15.15.3.6 hasZone()	162
15.15.3.7 postprocess()	163

15.15.3.8 postTimeStep()	163
15.15.3.9 preprocess()	164
15.15.3.10 setup()	164
15.15.3.11 step()	165
15.15.3.12 stepAppliancesNegotiation()	165
15.15.3.13 stepAppliancesNegotiationNeighbourhood()	166
15.15.3.14 stepAppliancesUse()	166
15.15.3.15 stepAppliancesUseBatteries()	167
15.16 Building_Appliances Class Reference	167
15.16.1 Detailed Description	168
15.16.2 Constructor & Destructor Documentation	168
15.16.2.1 Building_Appliances()	168
15.16.3 Member Function Documentation	168
15.16.3.1 addContactsTo()	169
15.16.3.2 addCurrentStates()	169
15.16.3.3 getTotalPower()	170
15.16.3.4 postprocess()	170
15.16.3.5 postTimeStep()	171
15.16.3.6 preprocess()	171
15.16.3.7 setup()	171
15.16.3.8 stepAppliancesUseBatteries()	172
15.16.3.9 stepGlobalNegotiation()	173
15.16.3.10 stepLocal()	174
15.16.3.11 stepLocalNegotiation()	174
15.16.3.12 stepNeighbourhoodNegotiation()	175
15.17 Building_Zone Class Reference	176
15.17.1 Detailed Description	177
15.17.2 Constructor & Destructor Documentation	177
15.17.2.1 Building_Zone()	177
15.17.3 Member Function Documentation	177

15.17.3.1 getActivities()	178
15.17.3.2 getAirRelativeHumidity()	178
15.17.3.3 getAirSystemSensibleHeatingRate()	179
15.17.3.4 getBlindState()	179
15.17.3.5 getCurrentOccupantCount()	179
15.17.3.6 getCurrentOccupantGains()	179
15.17.3.7 getDaylightingReferencePoint1Illuminance()	180
15.17.3.8 getGroundFloor()	180
15.17.3.9 getHeatingState()	181
15.17.3.10getId()	181
15.17.3.11getLightState()	182
15.17.3.12getMeanAirTemperature()	183
15.17.3.13getMeanRadiantTemperature()	183
15.17.3.14getNumberOfActivities()	184
15.17.3.15getOccupantFraction()	184
15.17.3.16getWindowDurationOpen()	184
15.17.3.17getWindowState()	184
15.17.3.18hasActivity()	185
15.17.3.19sActive()	185
15.17.3.20sNamed()	185
15.17.3.21 setActive()	185
15.17.3.22setAppFraction()	186
15.17.3.23setBlindState()	186
15.17.3.24setCurrentOccupantGains()	186
15.17.3.25setGroundFloor()	186
15.17.3.26setHeatingState()	187
15.17.3.27setIdString()	187
15.17.3.28setLightState()	187
15.17.3.29setName()	187
15.17.3.30setOccupantFraction()	187

15.17.3.31setup()	187
15.17.3.32setWindowDurationOpen()	188
15.17.3.33setWindowState()	188
15.17.3.34step()	188
15.18ConfigStructAgent Struct Reference	189
15.18.1 Detailed Description	190
15.18.2 Member Data Documentation	190
15.18.2.1 age	190
15.18.2.2 ApplianceDuringDay	190
15.18.2.3 bedroom	191
15.18.2.4 civstat	191
15.18.2.5 computer	191
15.18.2.6 edtry	191
15.18.2.7 famstat	191
15.18.2.8 LightOffDuringAudioVisual	191
15.18.2.9 LightOffDuringSleep	192
15.18.2.10name	192
15.18.2.11office	192
15.18.2.12power	192
15.18.2.13profile	192
15.18.2.14retired	192
15.18.2.15sex	193
15.18.2.16ShadeClosedDuringSleep	193
15.18.2.17ShadeClosedDuringWashing	193
15.18.2.18ShadeDuringAudioVisual	193
15.18.2.19ShadeDuringNight	193
15.18.2.20shadeld	193
15.18.2.21unemp	194
15.18.2.22windowld	194
15.18.2.23WindowOpenDuringCooking	194

15.18.2.24WindowOpenDuringSleeping	194
15.18.2.25WindowOpenDuringWashing	194
15.19ConfigStructAppliance Struct Reference	195
15.19.1 Detailed Description	195
15.19.2 Member Data Documentation	196
15.19.2.1 activities	196
15.19.2.2 alpha	196
15.19.2.3 batteryNeighbourhoodCharge	196
15.19.2.4 batteryNeighbourhoodDischarge	196
15.19.2.5 cost	196
15.19.2.6 costVector	197
15.19.2.7 epsilon	197
15.19.2.8 fileDemand	197
15.19.2.9 fileProfile	197
15.19.2.10fileSupply	197
15.19.2.11Fractions	197
15.19.2.12gamma	198
15.19.2.13d	198
15.19.2.14name	198
15.19.2.15priority	198
15.19.2.16StateProbabilities	198
15.19.2.17SumRatedPowers	198
15.19.2.18timeRequired	199
15.19.2.19update	199
15.19.2.20variableName	199
15.19.2.21WeibullParameters	199
15.20ConfigStructBuilding Struct Reference	200
15.20.1 Detailed Description	200
15.20.2 Member Data Documentation	201
15.20.2.1 agents	201

15.20.2.2 AppliancesBattery	201
15.20.2.3 AppliancesBatteryGrid	201
15.20.2.4 AppliancesCSV	201
15.20.2.5 AppliancesFMI	201
15.20.2.6 AppliancesGrid	202
15.20.2.7 AppliancesLarge	202
15.20.2.8 AppliancesLargeCSV	202
15.20.2.9 AppliancesLargeLearning	202
15.20.2.10 AppliancesLargeLearningCSV	202
15.20.2.11 AppliancesSmall	202
15.20.2.12d	203
15.20.2.13name	203
15.20.2.14zones	203
15.21 ConfigStructLVNode Struct Reference	203
15.21.1 Detailed Description	204
15.21.2 Member Data Documentation	204
15.21.2.1 children	204
15.21.2.2 id	204
15.21.2.3 impedance	204
15.21.2.4 parent	204
15.22 ConfigStructShade Struct Reference	205
15.22.1 Detailed Description	205
15.22.2 Member Data Documentation	206
15.22.2.1 a01arr	206
15.22.2.2 a01int	206
15.22.2.3 a10arr	206
15.22.2.4 a10int	206
15.22.2.5 afulllower	206
15.22.2.6 afullraise	207
15.22.2.7 aSFlower	207

15.22.2.8 b01inarr	207
15.22.2.9 b01inint	207
15.22.2.10b01sarr	207
15.22.2.11b01sint	207
15.22.2.12b10inarr	208
15.22.2.13b10inint	208
15.22.2.14b10sarr	208
15.22.2.15b10sint	208
15.22.2.16boutfulllower	208
15.22.2.17boutfullraise	208
15.22.2.18bSFlower	209
15.22.2.19bsfulllower	209
15.22.2.20bsfullraise	209
15.22.2.21shapelower	209
15.23ConfigStructSimulation Struct Reference	209
15.23.1 Detailed Description	210
15.23.2 Member Data Documentation	210
15.23.2.1 agentHeatGains	210
15.23.2.2 caseOrder	210
15.23.2.3 endDay	211
15.23.2.4 endMonth	211
15.23.2.5 GridCost	211
15.23.2.6 heating	211
15.23.2.7 learn	211
15.23.2.8 learnep	211
15.23.2.9 learnupdate	212
15.23.2.10ights	212
15.23.2.11precision	212
15.23.2.12presencePage	212
15.23.2.13save	212

15.23.2.14	ShadeClosedDuringNight	212
15.23.2.15	shading	213
15.23.2.16	startDay	213
15.23.2.17	startDayOfWeek	213
15.23.2.18	startMonth	213
15.23.2.19	timeSteps	213
15.23.2.20	timeStepsPerHour	213
15.23.2.21	windows	214
15.23.2.22	windowsLearn	214
15.24	ConfigStructWindow Struct Reference	214
15.24.1	Detailed Description	215
15.24.2	Member Data Documentation	215
15.24.2.1	a01arr	215
15.24.2.2	a01dep	215
15.24.2.3	a01int	216
15.24.2.4	a10dep	216
15.24.2.5	aop	216
15.24.2.6	b01absdep	216
15.24.2.7	b01absprevarr	216
15.24.2.8	b01gddep	216
15.24.2.9	b01inarr	217
15.24.2.10	b01int	217
15.24.2.11	b01outarr	217
15.24.2.12	b01outdep	217
15.24.2.13	b01outint	217
15.24.2.14	b01presint	217
15.24.2.15	b01rnarr	218
15.24.2.16	b01rnint	218
15.24.2.17	b10absdep	218
15.24.2.18	b10gddep	218

15.24.2.19b10indep	218
15.24.2.20b10outdep	218
15.24.2.21bopout	219
15.24.2.22shapeop	219
15.25ConfigStructZone Struct Reference	219
15.25.1 Detailed Description	220
15.25.2 Member Data Documentation	220
15.25.2.1 active	220
15.25.2.2 activities	220
15.25.2.3 groundFloor	220
15.25.2.4 id	220
15.25.2.5 name	220
15.25.2.6 windowCount	221
15.26Configuration Class Reference	221
15.26.1 Detailed Description	222
15.26.2 Member Function Documentation	222
15.26.2.1 getStepCount()	223
15.26.2.2 getZone()	223
15.26.2.3 isZoneGroundFloor()	224
15.26.2.4 lengthOfTimestep()	224
15.26.2.5 parseConfiguration()	224
15.26.2.6 reset()	225
15.26.2.7 setStepCount()	226
15.26.2.8 step()	226
15.26.3 Member Data Documentation	226
15.26.3.1 buildings	226
15.26.3.2 FileActivity	226
15.26.3.3 FileLargeAppliance	226
15.26.3.4 FolderSmallAppliance	227
15.26.3.5 info	227

15.26.3.6 <code>lvn</code>	227
15.26.3.7 <code>outputRegExs</code>	227
15.26.3.8 <code>RunLocation</code>	227
15.26.3.9 <code>shades</code>	227
15.26.3.10 <code>windows</code>	228
15.27 Contract Struct Reference	228
15.27.1 Detailed Description	229
15.27.2 Member Data Documentation	229
15.27.2.1 <code>buildingID</code>	229
15.27.2.2 <code>id</code>	229
15.27.2.3 <code>priority</code>	229
15.27.2.4 <code>received</code>	229
15.27.2.5 <code>receivedCost</code>	230
15.27.2.6 <code>requested</code>	230
15.27.2.7 <code>supplied</code>	230
15.27.2.8 <code>suppliedCost</code>	230
15.27.2.9 <code>suppliedLeft</code>	230
15.28 Contract_Negotiation Class Reference	231
15.28.1 Detailed Description	231
15.28.2 Constructor & Destructor Documentation	231
15.28.2.1 <code>Contract_Negotiation()</code>	232
15.28.3 Member Function Documentation	232
15.28.3.1 <code>clear()</code>	232
15.28.3.2 <code>getContract()</code>	233
15.28.3.3 <code>getCostOfPowerForContract()</code>	233
15.28.3.4 <code>getDifference()</code>	234
15.28.3.5 <code>getReceivedPowerForContract()</code>	234
15.28.3.6 <code>process()</code>	235
15.28.3.7 <code>submit()</code>	235
15.29 Contract_Node_Priority Class Reference	236

15.29.1 Detailed Description	238
15.29.2 Constructor & Destructor Documentation	239
15.29.2.1 Contract_Node_Priority()	239
15.29.3 Member Function Documentation	239
15.29.3.1 compare()	239
15.29.3.2 isNodeRemoveable()	239
15.29.3.3 isRemoveable()	239
15.29.3.4 makeLeft()	240
15.29.3.5 makeRight()	240
15.30Contract_Node_Supply Class Reference	240
15.30.1 Detailed Description	242
15.30.2 Constructor & Destructor Documentation	243
15.30.2.1 Contract_Node_Supply()	243
15.30.3 Member Function Documentation	243
15.30.3.1 compare()	243
15.30.3.2 isNodeRemoveable()	243
15.30.3.3 isRemoveable()	243
15.30.3.4 makeLeft()	244
15.30.3.5 makeRight()	244
15.31Contract_Node_Tree< T > Class Template Reference	244
15.31.1 Detailed Description	246
15.31.2 Constructor & Destructor Documentation	246
15.31.2.1 Contract_Node_Tree()	246
15.31.3 Member Function Documentation	246
15.31.3.1 clear()	246
15.31.3.2 compare()	247
15.31.3.3 findLeftEdge()	247
15.31.3.4 findRightEdge()	247
15.31.3.5 getNodeObject()	247
15.31.3.6 insert()	248

15.31.3.7 isLeftNull()	248
15.31.3.8 isNodeRemoveable()	248
15.31.3.9 isRemoveable()	249
15.31.3.10 isRightNull()	249
15.31.3.11 makeLeft()	249
15.31.3.12 makeRight()	250
15.31.3.13 popLeftEdge()	250
15.31.4 Member Data Documentation	250
15.31.4.1 nodeObject	250
15.31.4.2 pLeft	251
15.31.4.3 pRight	251
15.32 DataStore Class Reference	251
15.32.1 Detailed Description	252
15.32.2 Member Function Documentation	252
15.32.2.1 addValue()	252
15.32.2.2 addValueS()	253
15.32.2.3 addVariable()	254
15.32.2.4 clear()	254
15.32.2.5 clearValues()	255
15.32.2.6 getID()	255
15.32.2.7 getValue()	256
15.32.2.8 getValueForZone()	256
15.32.2.9 getValueS()	257
15.32.2.10 print()	257
15.33 Environment Class Reference	258
15.33.1 Detailed Description	258
15.33.2 Member Function Documentation	258
15.33.2.1 calculateDailyMeanTemperature()	259
15.33.2.2 getDailyMeanTemperature()	259
15.33.2.3 getEVG()	260

15.33.2.4 getOutdoorAirDrybulbTemperature()	260
15.33.3 Member Data Documentation	261
15.33.3.1 dailyMeanTemperature	261
15.34fmiCallbackFunctions Struct Reference	261
15.34.1 Detailed Description	261
15.34.2 Member Data Documentation	262
15.34.2.1 allocateMemory	262
15.34.2.2 freeMemory	262
15.34.2.3 logger	262
15.34.2.4 stepFinished	262
15.35fmiEventInfo Struct Reference	263
15.35.1 Detailed Description	263
15.35.2 Member Data Documentation	263
15.35.2.1 iterationConverged	263
15.35.2.2 nextEventTime	264
15.35.2.3 stateValueReferencesChanged	264
15.35.2.4 stateValuesChanged	264
15.35.2.5 terminateSimulation	264
15.35.2.6 upcomingTimeEvent	264
15.36Log Class Reference	265
15.36.1 Detailed Description	265
15.36.2 Constructor & Destructor Documentation	265
15.36.2.1 Log()	266
15.36.3 Member Function Documentation	266
15.36.3.1 error()	266
15.36.3.2 getError()	267
15.36.3.3 operator<<()	267
15.36.3.4 printLog()	268
15.36.3.5 reset()	268
15.37LVN Class Reference	269

15.37.1 Detailed Description	269
15.37.2 Constructor & Destructor Documentation	269
15.37.2.1 LVN()	269
15.37.3 Member Function Documentation	270
15.37.3.1 postTimeStep()	270
15.37.3.2 setPowerForID()	270
15.37.3.3 setup()	271
15.38LVN_Node Class Reference	272
15.38.1 Detailed Description	273
15.38.2 Constructor & Destructor Documentation	273
15.38.2.1 LVN_Node()	273
15.38.3 Member Function Documentation	273
15.38.3.1 addChildren()	274
15.38.3.2 addNode()	274
15.38.3.3 backwardSweep()	274
15.38.3.4 checkTolerance()	275
15.38.3.5 forwardSweep()	276
15.38.3.6 getID()	276
15.38.3.7 resetIterations()	276
15.38.3.8 runUntilConvergence()	276
15.38.3.9 save()	277
15.38.3.10setId()	278
15.38.3.11setImpedance()	278
15.38.3.12setNodeLoad()	279
15.38.3.13setNominalVoltage()	279
15.38.3.14setPowerForID()	279
15.38.3.15setup()	280
15.39Model_Activity Class Reference	281
15.39.1 Detailed Description	283
15.39.2 Constructor & Destructor Documentation	283

15.39.2.1 Model_Activity()	284
15.39.3 Member Function Documentation	284
15.39.3.1 disaggregate()	284
15.39.3.2 getDay()	285
15.39.3.3 getSeasonInt()	285
15.39.3.4 getSeasonString()	285
15.39.3.5 multinomial()	286
15.39.3.6 multinomialActivity()	286
15.39.3.7 multinomialP()	287
15.39.3.8 parseConfiguration()	287
15.39.3.9 parseOther()	288
15.39.3.10 preProcessActivities()	288
15.39.3.11 setAge()	289
15.39.3.12 setCivstat()	289
15.39.3.13 setComputer()	290
15.39.3.14 setEdtry()	290
15.39.3.15 setFamstat()	290
15.39.3.16 setProbMap()	291
15.39.3.17 setRetired()	291
15.39.3.18 setSex()	292
15.39.3.19 setUnemp()	292
15.39.4 Member Data Documentation	292
15.39.4.1 age	292
15.39.4.2 civstat	293
15.39.4.3 computer	293
15.39.4.4 dictionary	293
15.39.4.5 edtry	293
15.39.4.6 famstat	293
15.39.4.7 probMap	293
15.39.4.8 retired	294

15.39.4.9 sex	294
15.39.4.10unemp	294
15.40Model_Activity_Survival Class Reference	294
15.40.1 Detailed Description	297
15.40.2 Constructor & Destructor Documentation	297
15.40.2.1 Model_Activity_Survival()	297
15.40.3 Member Function Documentation	297
15.40.3.1 multinominalActivity()	297
15.41Model_Appliance_Large_Usage Class Reference	298
15.41.1 Detailed Description	300
15.41.2 Constructor & Destructor Documentation	300
15.41.2.1 Model_Appliance_Large_Usage()	300
15.41.3 Member Function Documentation	301
15.41.3.1 as_vector()	301
15.41.3.2 as_vector_vector()	301
15.41.3.3 consumption()	301
15.41.3.4 getCountry()	302
15.41.3.5 getMeanFraction()	302
15.41.3.6 getPower()	303
15.41.3.7 isOn()	303
15.41.3.8 onAt()	304
15.41.3.9 parseConfiguration()	304
15.41.3.10parseShapeScale()	305
15.41.3.11probOn()	305
15.41.3.12setCountry()	306
15.41.3.13setID()	306
15.41.4 Member Data Documentation	306
15.41.4.1 country	306
15.41.4.2 id	306
15.41.4.3 maxPower	307

15.41.4.4 meanF	307
15.41.4.5 name	307
15.41.4.6 on	307
15.41.4.7 onProbabilities	307
15.41.4.8 onProbabilities10	307
15.41.4.9 state	308
15.41.4.10transitions	308
15.42Model_Appliance_Large_Usage_Survival Class Reference	308
15.42.1 Detailed Description	311
15.42.2 Constructor & Destructor Documentation	311
15.42.2.1 Model_Appliance_Large_Usage_Survival()	311
15.42.3 Member Function Documentation	311
15.42.3.1 decreaseDuration()	312
15.42.3.2 onAt()	312
15.42.3.3 setDuration()	313
15.42.3.4 setScale()	313
15.42.3.5 setShape()	314
15.43Model_Appliance_Ownership Class Reference	314
15.43.1 Detailed Description	315
15.43.2 Constructor & Destructor Documentation	315
15.43.2.1 Model_Appliance_Ownership()	315
15.43.3 Member Function Documentation	315
15.43.3.1 cooker()	315
15.43.3.2 dishwasher()	315
15.43.3.3 fridge()	316
15.43.3.4 microware()	316
15.43.3.5 tvless21()	316
15.43.3.6 tvmore21()	316
15.43.3.7 washingMachine()	317
15.44Model_Appliance_Power_CSV Class Reference	317

15.44.1 Detailed Description	317
15.44.2 Constructor & Destructor Documentation	318
15.44.2.1 Model_Appliance_Power_CSV()	318
15.44.3 Member Function Documentation	318
15.44.3.1 parseConfiguration()	318
15.44.3.2 power()	318
15.45 Model_Appliance_Small_Usage Class Reference	319
15.45.1 Detailed Description	321
15.45.2 Constructor & Destructor Documentation	321
15.45.2.1 Model_Appliance_Small_Usage()	321
15.45.3 Member Function Documentation	321
15.45.3.1 calculateStateAtTenMin()	321
15.45.3.2 consumption()	322
15.45.3.3 durationAtState()	322
15.45.3.4 getFractionalPowerAtState()	323
15.45.3.5 getStateProbabilities()	323
15.45.3.6 readFractions()	323
15.45.3.7 readStateProbabilities()	324
15.45.3.8 readSumRatedPowers()	324
15.45.3.9 readWeibullParameters()	324
15.45.3.10 setFolderLocation()	325
15.45.3.11 setRatedPowerAt()	325
15.45.3.12 weibullInvCdf()	326
15.46 Model_ExternalShading Class Reference	326
15.46.1 Detailed Description	328
15.46.2 Constructor & Destructor Documentation	329
15.46.2.1 Model_ExternalShading()	329
15.46.3 Member Function Documentation	329
15.46.3.1 arrival()	329
15.46.3.2 departure()	330

15.46.3.3 intermediate()	330
15.46.3.4 setArrivalVars()	331
15.46.3.5 setDurationVars()	331
15.46.3.6 setFullVars()	332
15.46.3.7 setInterVars()	332
15.47Model_HeatGains Class Reference	333
15.47.1 Detailed Description	333
15.47.2 Constructor & Destructor Documentation	334
15.47.2.1 Model_HeatGains()	334
15.47.3 Member Function Documentation	334
15.47.3.1 calculate()	334
15.47.3.2 getAllHeatGains()	335
15.47.3.3 getConvectiveHeatGains()	335
15.47.3.4 getDryRespiration()	335
15.47.3.5 getLatentRespirationHeatGains()	335
15.47.3.6 getPmv()	336
15.47.3.7 getPpd()	336
15.47.3.8 getRadiantHeatGains()	336
15.47.3.9 getSweatEvaporation()	336
15.48Model_Lights Class Reference	337
15.48.1 Detailed Description	337
15.48.2 Constructor & Destructor Documentation	337
15.48.2.1 Model_Lights()	337
15.48.3 Member Function Documentation	338
15.48.3.1 arrival()	338
15.48.3.2 departure()	338
15.48.3.3 intermediate()	339
15.49Model_Presence Class Reference	340
15.49.1 Detailed Description	340
15.49.2 Constructor & Destructor Documentation	340

15.49.2.1 Model_Presence()	341
15.49.3 Member Function Documentation	341
15.49.3.1 at()	341
15.49.3.2 calculatePresenceFromPage()	341
15.49.3.3 presentForFutureSteps()	342
15.49.3.4 setProbMap()	342
15.49.3.5 size()	343
15.50 Model_RandomWeibull Class Reference	343
15.50.1 Detailed Description	344
15.50.2 Constructor & Destructor Documentation	344
15.50.2.1 Model_RandomWeibull()	344
15.50.3 Member Function Documentation	345
15.50.3.1 probability()	345
15.50.3.2 randomDouble() [1/2]	345
15.50.3.3 randomDouble() [2/2]	346
15.50.3.4 randomWeibull()	346
15.51 Model_Windows Class Reference	347
15.51.1 Detailed Description	349
15.51.2 Constructor & Destructor Documentation	349
15.51.2.1 Model_Windows()	349
15.51.3 Member Function Documentation	349
15.51.3.1 arrival()	350
15.51.3.2 departure()	350
15.51.3.3 getA01arr()	351
15.51.3.4 getA01dep()	351
15.51.3.5 getA01int()	351
15.51.3.6 getA10dep()	351
15.51.3.7 getAop()	351
15.51.3.8 getB01absdep()	352
15.51.3.9 getB01absprevarr()	352

15.51.3.10getB01gddep()	352
15.51.3.11getB01inarr()	352
15.51.3.12getB01init()	352
15.51.3.13getB01outarr()	352
15.51.3.14getB01outdep()	353
15.51.3.15getB01outint()	353
15.51.3.16getB01presint()	353
15.51.3.17getB01rnarr()	353
15.51.3.18getB01rnint()	353
15.51.3.19getB10absdep()	353
15.51.3.20getB10gddep()	354
15.51.3.21getB10indep()	354
15.51.3.22getB10outdep()	354
15.51.3.23getBopout()	354
15.51.3.24getDurationOpen()	354
15.51.3.25getshapeop()	355
15.51.3.26getWindowState()	355
15.51.3.27intermediate()	355
15.51.3.28setArrivalVars()	356
15.51.3.29setDepartureVars()	356
15.51.3.30setDurationOpen()	357
15.51.3.31setDurationVars()	357
15.51.3.32setInterVars()	357
15.51.3.33setWindowState()	358
15.52ModellInstance Struct Reference	358
15.52.1 Detailed Description	359
15.52.2 Member Data Documentation	359
15.52.2.1 b	359
15.52.2.2 eventInfo	359
15.52.2.3 functions	359

15.52.2.4 GUID	360
15.52.2.5 i	360
15.52.2.6 instanceName	360
15.52.2.7 isPositive	360
15.52.2.8 loggingOn	360
15.52.2.9 r	360
15.52.2.10s	361
15.52.2.11sim	361
15.52.2.12state	361
15.52.2.13time	361
15.53 Occupant Class Reference	362
15.53.1 Detailed Description	364
15.53.2 Constructor & Destructor Documentation	364
15.53.2.1 Occupant()	364
15.53.3 Member Function Documentation	364
15.53.3.1 currentlyInZone()	365
15.53.3.2 getCurrentRadientGains()	365
15.53.3.3 getDesiredAppliance()	366
15.53.3.4 getDesiredHeatState()	366
15.53.3.5 getDesiredLightState()	367
15.53.3.6 getDesiredShadeState()	367
15.53.3.7 getDesiredWindowState()	368
15.53.3.8 getPower()	368
15.53.3.9 getStateID()	368
15.53.3.10InteractionOnZone()	369
15.53.3.11isActionAppliance()	369
15.53.3.12sActionHeatGains()	369
15.53.3.13sActionLearning()	370
15.53.3.14sActionLights()	370
15.53.3.15sActionShades()	371

15.53.3.16sActionWindow()	371
15.53.3.17postprocess()	371
15.53.3.18postTimeStep()	372
15.53.3.19previouslyInZone()	372
15.53.3.20setBuildingName()	372
15.53.3.21setState()	373
15.53.3.22setup()	373
15.53.3.23step()	374
15.53.3.24zoneInteractions()	375
15.54Occupant_Action Class Reference	375
15.54.1 Detailed Description	376
15.54.2 Constructor & Destructor Documentation	377
15.54.2.1 Occupant_Action()	377
15.54.3 Member Function Documentation	377
15.54.3.1 activityAvailable()	377
15.54.3.2 getCurrentDurationOfPresenceState()	377
15.54.3.3 getFutureDurationOfAbsenceState()	378
15.54.3.4 getPreviousDurationOfAbsenceState()	379
15.54.3.5 getResult()	379
15.54.3.6 setAvailableActivities()	380
15.54.3.7 setReward()	380
15.54.3.8 setZoneld()	380
15.54.4 Member Data Documentation	381
15.54.4.1 availableActivities	381
15.54.4.2 result	381
15.54.4.3 reward	381
15.54.4.4 zoneld	381
15.55Occupant_Action_Appliance Class Reference	382
15.55.1 Detailed Description	383
15.55.2 Constructor & Destructor Documentation	383

15.55.2.1 Occupant_Action_Appliance()	384
15.56Occupant_Action_Appliance_BDI Class Reference	384
15.56.1 Detailed Description	385
15.56.2 Constructor & Destructor Documentation	386
15.56.2.1 Occupant_Action_Appliance_BDI()	386
15.56.3 Member Function Documentation	386
15.56.3.1 doRecipe()	386
15.56.3.2 setApplianceDuringDay()	387
15.57Occupant_Action_Heat_Gains Class Reference	387
15.57.1 Detailed Description	389
15.57.2 Constructor & Destructor Documentation	390
15.57.2.1 Occupant_Action_Heat_Gains()	390
15.57.3 Member Function Documentation	390
15.57.3.1 getPMV()	390
15.57.3.2 getPPD()	390
15.57.3.3 prestep()	391
15.57.3.4 setup()	391
15.57.3.5 step()	392
15.58Occupant_Action_HeatingSetPoints_Learning Class Reference	393
15.58.1 Detailed Description	395
15.58.2 Constructor & Destructor Documentation	396
15.58.2.1 Occupant_Action_HeatingSetPoints_Learning()	396
15.58.3 Member Function Documentation	396
15.58.3.1 print()	396
15.58.3.2 reset()	397
15.58.3.3 setFile()	397
15.58.3.4 setup()	397
15.58.3.5 step()	398
15.59Occupant_Action_Lights Class Reference	399
15.59.1 Detailed Description	401

15.59.2 Constructor & Destructor Documentation	401
15.59.2.1 Occupant_Action_Lights()	402
15.59.3 Member Function Documentation	402
15.59.3.1 step()	402
15.60 Occupant_Action_Lights_BDI Class Reference	403
15.60.1 Detailed Description	405
15.60.2 Constructor & Destructor Documentation	405
15.60.2.1 Occupant_Action_Lights_BDI()	405
15.60.3 Member Function Documentation	405
15.60.3.1 doRecipe()	405
15.60.3.2 setOffDuringAudioVisual()	406
15.60.3.3 setOffDuringSleep()	406
15.61 Occupant_Action_Shades Class Reference	407
15.61.1 Detailed Description	408
15.61.2 Constructor & Destructor Documentation	409
15.61.2.1 Occupant_Action_Shades()	409
15.61.3 Member Function Documentation	409
15.61.3.1 setIndoorIlluminance()	409
15.61.3.2 setup()	409
15.61.3.3 step()	410
15.61.4 Member Data Documentation	411
15.61.4.1 Lumint	411
15.62 Occupant_Action_Shades_BDI Class Reference	411
15.62.1 Detailed Description	414
15.62.2 Constructor & Destructor Documentation	414
15.62.2.1 Occupant_Action_Shades_BDI()	414
15.62.3 Member Function Documentation	414
15.62.3.1 doRecipe()	414
15.62.3.2 setClosedDuringAudioVisual()	415
15.62.3.3 setClosedDuringNight()	415

15.62.3.4 setClosedDuringSleep()	416
15.62.3.5 setClosedDuringWashing()	416
15.63Occupant_Action_Window Class Reference	417
15.63.1 Detailed Description	419
15.63.2 Constructor & Destructor Documentation	419
15.63.2.1 Occupant_Action_Window()	419
15.63.3 Member Function Documentation	419
15.63.3.1 durationOpen()	419
15.63.3.2 saveResult()	420
15.63.3.3 setDailyMeanTemperature()	420
15.63.3.4 setup()	421
15.63.4 Member Data Documentation	421
15.63.4.1 dailyMeanTemperature	421
15.63.4.2 m_window	421
15.63.4.3 variableNameWindowDesire	421
15.64Occupant_Action_Window_Learning Class Reference	422
15.64.1 Detailed Description	424
15.64.2 Constructor & Destructor Documentation	424
15.64.2.1 Occupant_Action_Window_Learning()	424
15.64.3 Member Function Documentation	424
15.64.3.1 print()	424
15.64.3.2 reset()	425
15.64.3.3 setup()	425
15.64.3.4 step()	426
15.65Occupant_Action_Window_Stochastic Class Reference	427
15.65.1 Detailed Description	429
15.65.2 Constructor & Destructor Documentation	429
15.65.2.1 Occupant_Action_Window_Stochastic()	429
15.65.3 Member Function Documentation	429
15.65.3.1 setup()	429

15.65.3.2 step()	430
15.66Occupant_Action_Window_Stochastic_BDI Class Reference	431
15.66.1 Detailed Description	434
15.66.2 Constructor & Destructor Documentation	434
15.66.2.1 Occupant_Action_Window_Stochastic_BDI()	434
15.66.3 Member Function Documentation	434
15.66.3.1 doRecipe()	434
15.66.3.2 enabled()	435
15.66.3.3 setDailyMeanTemperature()	435
15.66.3.4 setOpenDuringCooking()	435
15.66.3.5 setOpenDuringSleeping()	436
15.66.3.6 setOpenDuringWashing()	436
15.67Occupant_Zone Class Reference	437
15.67.1 Detailed Description	438
15.67.2 Constructor & Destructor Documentation	438
15.67.2.1 Occupant_Zone()	438
15.67.3 Member Function Documentation	438
15.67.3.1 actionStep()	439
15.67.3.2 getDesiredAppliance()	440
15.67.3.3 getDesiredHeatingSetPoint()	440
15.67.3.4 getDesiredLightState()	440
15.67.3.5 getDesiredShadeState()	440
15.67.3.6 getDesiredWindowDuration()	441
15.67.3.7 getDesiredWindowState()	441
15.67.3.8 getHeatgains()	441
15.67.3.9 getId()	441
15.67.3.10getPMV()	441
15.67.3.11isActionAppliance()	442
15.67.3.12sActionHeatGains()	442
15.67.3.13sActionLearning()	442

15.67.3.14sActionLights()	442
15.67.3.15sActionShades()	443
15.67.3.16sActionWindow()	443
15.67.3.17postprocess()	443
15.67.3.18postTimeStep()	444
15.67.3.19setClo()	444
15.67.3.20setMetabolicRate()	444
15.67.3.21setup()	444
15.67.3.22step()	445
15.67.3.23stepPre()	446
15.68profileStruct Struct Reference	447
15.68.1 Detailed Description	448
15.68.2 Member Data Documentation	448
15.68.2.1 cost	448
15.68.2.2 isLearningPeriod	449
15.68.2.3 learningStep	449
15.68.2.4 maxTimeRequired	449
15.68.2.5 nonLearningStep	449
15.68.2.6 power	449
15.68.2.7 requestedTime	449
15.68.2.8 startTime	450
15.69QLearning Class Reference	450
15.69.1 Detailed Description	451
15.69.2 Constructor & Destructor Documentation	451
15.69.2.1 QLearning()	451
15.69.3 Member Function Documentation	452
15.69.3.1 getAction()	452
15.69.3.2 greedySelection()	452
15.69.3.3 learn()	453
15.69.3.4 printQ()	454

15.69.3.5 reset()	454
15.69.3.6 setAction()	455
15.69.3.7 setActions()	455
15.69.3.8 setAlpha()	456
15.69.3.9 setEpsilon()	456
15.69.3.10 setFilename()	457
15.69.3.11 setGamma()	457
15.69.3.12 setId()	458
15.69.3.13 setReward()	458
15.69.3.14 setState()	459
15.69.3.15 setStates()	459
15.69.3.16 setup()	460
15.69.3.17 setUpdate()	460
15.69.3.18 updateQ()	461
15.69.4 Member Data Documentation	461
15.69.4.1 action	461
15.69.4.2 actions	461
15.69.4.3 filename	461
15.69.4.4 id	462
15.69.4.5 learnNext	462
15.69.4.6 previous_reward	462
15.69.4.7 previous_state	462
15.69.4.8 reward	462
15.69.4.9 state	462
15.69.4.10 states	463
15.70 Simulation Class Reference	463
15.70.1 Detailed Description	464
15.70.2 Constructor & Destructor Documentation	464
15.70.2.1 Simulation()	464
15.70.3 Member Function Documentation	464

15.70.3.1 <code>getGridCost()</code>	464
15.70.3.2 <code>parseConfiguration()</code>	465
15.70.3.3 <code>postprocess()</code>	465
15.70.3.4 <code>postTimeStep()</code>	466
15.70.3.5 <code>preprocess()</code>	466
15.70.3.6 <code>preTimeStep()</code>	467
15.70.3.7 <code>setConfigurationFile()</code>	467
15.70.3.8 <code>setupSimulationModel()</code>	467
15.70.3.9 <code>timeStep()</code>	468
15.71 <code>SimulationTime</code> Class Reference	468
15.71.1 Detailed Description	470
15.71.2 Member Function Documentation	470
15.71.2.1 <code>preprocess()</code>	470
15.71.2.2 <code>reset()</code>	471
15.71.2.3 <code>trackTime()</code>	471
15.71.3 Member Data Documentation	472
15.71.3.1 <code>databaselDay</code>	472
15.71.3.2 <code>databaselHour</code>	472
15.71.3.3 <code>databaselHourOfDay</code>	472
15.71.3.4 <code>databaselMinute</code>	472
15.71.3.5 <code>databaselMinuteOfDay</code>	472
15.71.3.6 <code>databaselMonth</code>	473
15.71.3.7 <code>databaselStepCount</code>	473
15.71.3.8 <code>day</code>	473
15.71.3.9 <code>hour</code>	473
15.71.3.10 <code>hourOfDay</code>	473
15.71.3.11 <code>minute</code>	473
15.71.3.12 <code>minuteOfDay</code>	474
15.71.3.13 <code>month</code>	474
15.71.3.14 <code>stepCount</code>	474

15.72State Class Reference	474
15.72.1 Detailed Description	475
15.72.2 Constructor & Destructor Documentation	475
15.72.2.1 State() [1/2]	475
15.72.2.2 State() [2/2]	476
15.72.2.3 ~State()	476
15.72.3 Member Function Documentation	476
15.72.3.1 addState()	476
15.72.3.2 getClo()	477
15.72.3.3 getId()	477
15.72.3.4 getMetabolicRate()	477
15.72.3.5 getState()	478
15.72.3.6 getZonePtr()	478
15.72.3.7 hasState()	478
15.72.3.8 isInActivity()	479
15.72.3.9 numberOfSubStates()	479
15.72.3.10setActivity()	479
15.72.3.11setClo()	479
15.72.3.12setId()	479
15.72.3.13setMetabolicRate()	479
15.72.3.14setZonePtr()	480
15.72.4 Member Data Documentation	480
15.72.4.1 activity	480
15.72.4.2 clo	480
15.72.4.3 id	480
15.72.4.4 metabolicRate	481
15.72.4.5 states	481
15.72.4.6 zone	481
15.73StateMachine Class Reference	481
15.73.1 Detailed Description	482

15.73.2 Constructor & Destructor Documentation	482
15.73.2.1 StateMachine()	482
15.73.3 Member Function Documentation	482
15.73.3.1 addState()	482
15.73.3.2 hasState()	483
15.73.3.3 numberOfStates()	483
15.73.3.4 transitionTo()	484
15.74 Utility Class Reference	484
15.74.1 Detailed Description	486
15.74.2 Member Typedef Documentation	486
15.74.2.1 uTable	486
15.74.3 Member Function Documentation	486
15.74.3.1 calculateNumberOfDays()	486
15.74.3.2 csv.ToDouble()	487
15.74.3.3 csvToInt()	487
15.74.3.4 csvToTable() [1/2]	488
15.74.3.5 csvToTable() [2/2]	488
15.74.3.6 csvToTableHead()	489
15.74.3.7 cumulativeProbability() [1/2]	489
15.74.3.8 cumulativeProbability() [2/2]	490
15.74.3.9 randomDouble()	490
15.74.3.10 randomInt()	491
15.74.3.11 randomIntVect()	491
15.74.3.12 setSeed()	492
15.74.3.13 splitCSV()	492
15.74.3.14 tossACoin()	493
15.74.4 Member Data Documentation	493
15.74.4.1 engine	493
Index	495

Chapter 1

No-MASS

Introduction

The No-MASS framework integrates existing models of occupant interaction and appliance usage into a tool that can be coupled with building or urban energy performance simulation tools, or run independently.

Pages

- [Implementation](#)
- [Dependencies Of NoMASS](#)
- [Compiling No-MASS](#)
- [Using No-MASS](#)
- [Scripts For NoMASS](#)
- [Class Diagram](#)
- [Functional Mockup Unit \(FMU\)](#)
- [EnergyPlus Examples](#)

What is No-MASS?

A multi-agent stochastic simulation of occupants and appliances for use with building or urban scale simulations tools. No-MASS using the generic FMI interface so that it can interface with any tool that meets the standard.

What kind of data can I get from No-MASS?

- Stochastic window openings predictions
- Stochastic external shade fractions
- Stochastic lighting predictions
- Stochastic occupant activity predictions
- Stochastic occupant presence predictions
- Stochastic large appliance use
- Stochastic small appliance use
- Machine learning appliance profile shifting
- Machine learning heating/ cooling setpoint predictions
- Belief-Desire-Intent rules of occupant interactions

Background

The resources needed to sustain the world's ever expanding population are reaching new levels. It is therefore important to reduce global energy use arising from the contribution of fossil fuels and the associated emission of greenhouse gasses. Building performance simulation is used to model the flows of energy in buildings and their systems at the design/ retrofit stage, to ensure they not only meet the demands of the occupants but also allow designers to test strategies for improving building performance.

Although a powerful building (re-)design decision support tool, building performance simulations can be subject to limitations. Studies have found that buildings may use twice as much energy as predicted at design stage. Predicted building performance continues to deviate – sometimes considerably – from that which is observed post-build. With the objective of addressing these limitations, and following the suggestions of Robinson (2011), our approach is to use multi-agent stochastic simulation for modelling occupant behaviour; to combine stochastic models into a single package that can be used to support building and urban performance simulation using a range of software.

A multi-agent simulation framework has been developed, which we have named Nottingham Multi-Agent Stochastic Simulation (No-MASS). The No-MASS framework integrates existing stochastic models of occupant interaction into a tool that can be coupled with building or urban energy performance simulation tools. The coupling allows for simulated occupants to make changes within the simulated building environment and receive responses arising from the effects of the interactions which may stimulate future interactions. Current stochastic models do not cover all of the energy related behaviours of occupants, therefore a belief-desire-intent (BDI) rule system is used to model other interactions, supplementing the data-driven stochastic models. For example switching off the light during sleep. Agents can have unique desires causing changes within the environment that may be in conflict with the desires of other agents. To solve this problem an agent social interaction model is developed to govern the interactions between agents. For more complex interactions where BDI rules would be difficult to design, agent machine learning techniques are used, allowing the agent to learn how to respond to different stimuli.

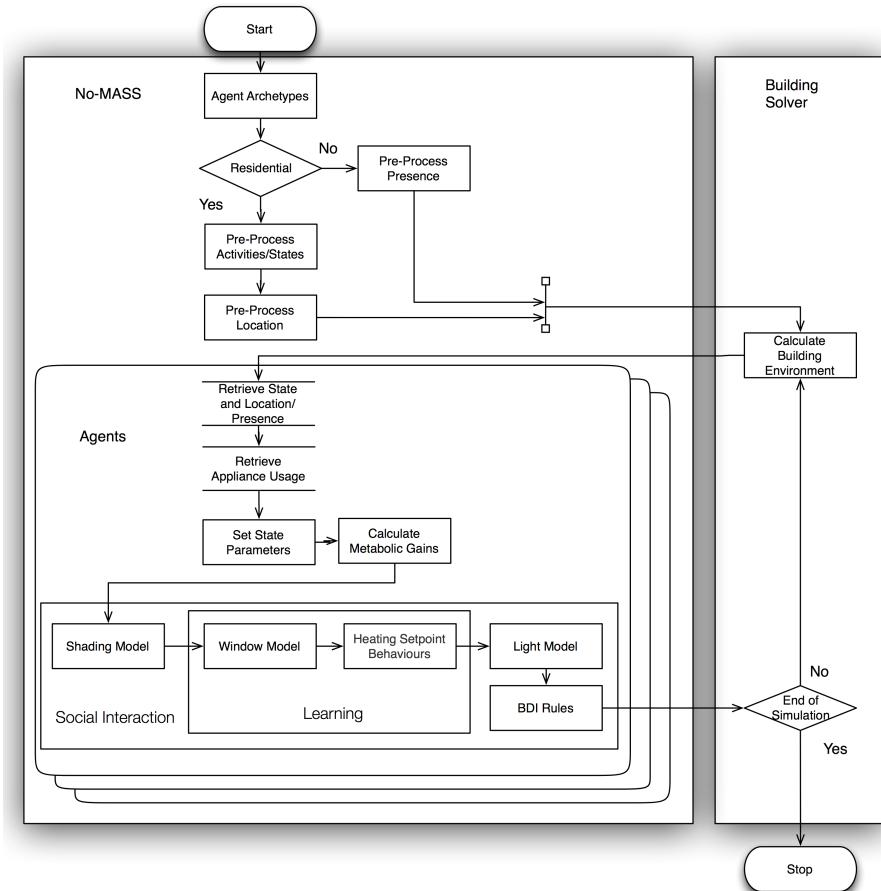
Chapter 2

Implementation

When launched, No-MASS first builds an agent population, assigning a profile to each member dependent on the input parameters supplied in a No-MASS configuration file. These profiles influence the models predicting activities (e.g. sleeping, bathing, watching TV) and dependent behaviours (e.g. opening windows or lowering shading devices). Once the agent profiles are assigned No-MASS proceeds according to one of two scenarios. If the building is non-residential, chains of presence and absence are calculated; otherwise for residential buildings chains of activities are calculated, and the corresponding activity locations are assigned to each activity. Communication with the building solver is then performed. The building solver that we have coupled with No-MASS is EnergyPlus. To achieve this we use the generic Functional Mockup Interface (FMI) co-simulation standard, so that we can also couple No-MASS with any other FMI compliant BPS software. The building solver calculates the environmental conditions within the zones that the agents have been allocated to and parses the results to No-MASS via the FMI. Each agent then retrieves its pre-processed location or presence (Page 2008). State parameters are then set based on the activity (Jaboob 2015) that is performed, affecting the agents' clothing level, metabolic rate etc. We then calculate the metabolic gains of each individual occupant. Next we call models predicting the agents' use of shades (Haldi 2010), windows (Haldi 2009), lights (Reinhart 2004) and heating system setpoints. To facilitate the modelling of agent social interactions we wrap these models in a social interaction framework; emulating negotiations through a vote casting mechanism. The prediction of agents' heating setpoint choices is based on a reinforcement learning model, allowing agents to learn over a number of simulation replicates the setpoints that best maintain their PMV within acceptable bounds. BDI rules are included to model relatively straight forward interactions for which data is scarce, such as closing shades for privacy in the home while showering. Finally the results are parsed back to the building solver which calculates the environmental conditions arising from the modelled interactions at the next time step.

No-MASS was built from the ground up, C++ was chosen as the development language as it is simple to integrate with EnergyPlus (Crawley 2001), our chosen building simulation tool is also developed in C++. Using the same language allows for easy communication between the two tools. EnergyPlus developed by the US Department of Energy, is well tested, well documented and open source; allowing us to readily understand how to connect to it. There are also two interfaces that allow other tools to interact with it, without altering the EnergyPlus source code. The first is through the building controls virtual test bed (BCVTB) and the second is through the Functional Mockup Interface (FMI) (Nouidui 2014). The No-MASS platform connects to EnergyPlus using FMI, which is an open standard so that No-MASS could in principle be integrated with any other FMI compliant simulation tool. This is chosen over the BCVTB as it allows direct communication through C++ double precision arrays using predefined calling points, whereas BCVTB requires calls over sockets adding complexity and slowing the processing time. The calling points are well documented, with No-MASS only using the initialise function, the receive an array of doubles function for the environmental variables and the send an array of doubles function for the occupant interactions. The array of values that No-MASS receives at each time step is defined in the XML file ModelDescription.xml. At the beginning of the time step the following environmental variables are received: horizontal sky illuminance, rain status, outdoor air dry-bulb temperature, zone air temperature, zone humidity, indoor radiant temperature and indoor illuminance. Returned to EnergyPlus are the number of occupants in a zone, their metabolic gains, appliance gains, the window status, the blind shading fraction, the lighting status and the heating setpoint. Due to the window, shading and location/presence models used within No-MASS a sub-hourly timestep is recommended (ie. 5 minutes), as longer timesteps may overestimate the implication of the occupant interactions. For example the response time to

an agent opening a window may be short with the room cooling in just a few minutes. An agent can only respond at the next time step, if the timesteps are not sufficiently short in length, the open window may over cool the room.



Chapter 3

Dependencies Of NoMASS

No-MASS depends on two external libraries, Google Tests and RapidXML. Google test is only needed to run the No-MASS tests.

These should be downloaded automatically with the command:

```
git submodule update --init --recursive
```

However if this fails then they can be manually downloaded.

- [Google Tests](#)
- [RapidXML](#)

Chapter 4

Using No-MASS

There are number of simulation files that need to be considered depending on what you wish to achieve with No-MASS:

- The main file is the [Simulation Configuration](#)
- For interfacing with FMI the [Model Description](#)
- For interfacing with EnergyPlus the [IDF File](#)

Chapter 5

Simulation Configuration

This is an XML document that is used to inform No-MASS how to run.

It contains the information for the simulation period, buildings, zones, agents and appliances.

The initial tag to define a number simulation

```
<simulation>
...
</simulation>
```

Seed

As we use stochastic models we define a seed value. This can be removed and No-MASS will generate its own. However as it is often required for results can be repeated, doing it manually will allow the random values to be the same.

```
<seed>0</seed>
```

Time Period

Here we define the time period information.

```
<timeStepsPerHour>60</timeStepsPerHour>
<beginMonth>1</beginMonth>
<endMonth>1</endMonth>
<beginDay>1</beginDay>
<endDay>7</endDay>
```

Saving Results

To save the results enable the save tag with the value 1. This writes out the No-MASS results to a NoMASS.out file. However this can slow the simulation down and can therefore be disabled with a 0 value.

```
<save>1</save>
```

The results file can be further filtered using regular expressions like that of below. Any output names that matches one of the options below is saved. If an output name does not match below it is discarded.

```
<output>
<regex>nsm</regex>
<regex>TimeStep</regex>
<regex>hour</regex>
<regex>day</regex>
<regex>Building[0-9]\d?_Appliance[0-9]\d?_received.*</regex>
<regex>Building[0-9]\d?_Appliance1[0-9]\d?_received.*</regex>
<regex>Building[0-9]\d?_Appliance10_supplied.*</regex>
<regex>Building[0-9]\d?_Appliance10_supplied</regex>
<regex>Building[0-9]\d?_Appliance1[0-9]\d?_supplied.*</regex>
<regex>Building[0-9]\d?_Appliance1[0-9]\d?_received.*</regex>
<regex>Building[0-9]\d?_Appliance[0-9]\d?_previous_state</regex>
<regex>Building[0-9]\d?_Appliance[0-9]\d?_action</regex>
<regex>Building[0-9]\d?_Appliance[0-9]\d?_reward</regex>
<regex>Building[0-9]\d?_Appliance[0-9]\d?_state</regex>
<regex>Building[0-9]\d?_Appliance1_cost</regex>
<regex>Building[0-9]\d?_Appliance4_cost</regex>
<regex>grid_power</regex>
<regex>grid_cost</regex>
</output>
```

Q-Learning for Occupants

Q-Learning occupant model specific however there will be times when you want to read from the learning data but not update the data with new values, for example once the training period is over. Use learnupdate to turn this off. The learn ep value is the epsilon value of Q-Learning and can be altered here with a double value.

```
<learnupdate>1</learnupdate>
<learnep>0.01</learnep>
```

The cost of using the grid

Any appliance that uses energy from the grid will use a tariff that is multiplied by the power used. This is specified per day/hour/halfhour using an array with a length of 1/24/44 respectively. Each value is the cost of one unit of power.

```
<GridCost>0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.9, 0.9, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.9, 0.9, 0.9, 0.9, 0.9, 0.5, 0.5, 0.1 </GridCost>
```

Buildings

Buildings are defined within the buildings tag, like so.

```
<buildings>
  <building>
    ...
  </building>
  ...
  <building>
    ...
  </building>
</buildings>
```

Building

A Building is define with a set of zones, agents and appliances.

```
<building>
  <zone>...</zone>
  <zone>...</zone>
  <Appliances>...</Appliances>
  <agents>...</agents>
</building>
```

Zone (Rooms)

A zone is a section of building often there is one per room, although a room can be sub divide into individual zones. Each zone is given a name, a set of activities and the number of windows in the the zone.

There are 10 activities as defined by the activity model:

- Cooking
- Cleaning
- AudioVisual
- Metabolic
- Passive
- Washing
- WashingAppliance
- Sleep
- IT
- Out

These are given to a zone and are comma seperated like below, however Out does not need to be defined.

The activities assign an agent to a zone, for example if Cooking is defined as an activity for that zone when an agent is in the state cooking they are also defined to this zone.

```
<zone>
  <name>Block1:Kitchen</name>
  <activities>Cooking,Cleaning</activities>
  <>windowCount>1</windowCount>
</zone>
<zone>
  <name>Block1:LivingRoom</name>
  <activities>AudioVisual,Metabolic,Passive</activities>
  <>windowCount>1</windowCount>
</zone>
```

Appliances

There are the following types of appliance:

- Large
 - LargeCSV
 - LargeLearning
 - LargeLearningCSV
 - Small
 - PV
 - CSV
 - FMI
 - Grid
 - Battery
 - BatteryGridCost

```
<Appliances>
  <Large>...</Large>
  <LargeCSV>...</LargeCSV>
  <LargeLearning>...</LargeLearning>
  <LargeLearningCSV>...</LargeLearningCSV>
  <Small>...</Small>
  <pv>...</pv>
  <FMI>...</FMI>
  <Grid>...</Grid>
  <Battery>...</Battery>
  <BatteryGridCost>...</BatteryGridCost>
</Appliances>
```

Appliance Priority

Each Appliance is given a priority, which is a value that defines the order in which the device will receive its demanded energy, higher priority energy receive there demand first.

Large Appliances

The large appliance has a id that relates to the id of the large appliances defined in the ApplianceLarge.xml file.

```
<Large>
  <id>0</id>
  <priority>0</priority>
</Large>
```

The large appliance can also be given a set of activities meaning that the appliance will only turn on when an agent is in the activity specified.

```
<Large>
  <id>0</id>
  <priority>0</priority>
  <activities>Cooking,Cleaning</activities>
</Large>
```

If required it is possible to override the large appliance predicted power profile with that of a profile specified through a csv file.

Large Appliances Q-Learning

Define in the same way as a large appliance only now we use the tag LargeLearning instead. This allows the appliance to shift its profile over time. Using the Q-Learning algorithm hopefully to a time when the supplied power is cheap.

If required it is possible to override the large learning appliance predicted power profile with that of a profile specified through a csv file.

Small Appliances

The small appliance are defined through csv files included with NoMASS

```
<Small>
  <id></id>
  <priority>6</priority>
  <WeibullParameters>weibull_parameters_audiovisual.csv</WeibullParameters>
  <StateProbabilities>state_probability_audiovisual.csv</StateProbabilities>
  <Fractions>mean_fractional_audiovisual.csv</Fractions>
  <SumRatedPowers>sum_rate_av.txt</SumRatedPowers>
</Small>
```

PV

Filename is the CSV profile used to generate the supply. Cost is the how much the supplied power is.

```
<pv>
  <id>10</id>
  <priority>0</priority>
  <filename>PVBowler2013_365.csv</filename>
  <cost>0.02</cost>
</pv>
```

CSV Appliance

Filename is the CSV profile used to generate the supply/demand. Cost is the how much the supplied power is.

```
<csv>
  <id>11</id>
  <priority>100</priority>
  <demand>HeatingPower.csv</demand>
  <supply>turbineSupply.csv</supply>

  <cost>0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.0,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.2,0.2</cost>
</csv>
```

FMI Appliance

Only used for appliances from the FMI interface (such as EnergyPlus sending the power), the variable name needs to correspond with the name in the modeldescription.xml file.

```
<FMI>
  <id>11</id>
  <priority>0</priority>
  <variablename>HVACPower</variablename>
</FMI>
```

Grid

Calculate the supply needed based on supply of the other appliances minus the demand of the other appliances. Cost is the how much the supplied power is.

```
<Grid>
  <id>1000</id>
  <priority>0</priority>
  <cost>1.00</cost>
</Grid>
```

Battery

The Battery can learn when discharge it stored energy. This is achieved through the q-learning algorithm. With the battery it is possible to define when it can charge and discharge to the local neighbourhood.

```
<battery>
  <id>12</id>
  <priority>0</priority>
  <epsilon>0.1</epsilon>
  <alpha>0.3</alpha>
  <gamma>0.1</gamma>
  <updateQTable>1</updateQTable>
  <batteryneighbourhooddischarge>0</batteryneighbourhooddischarge>
  <batteryneighbourhoodcharge>1</batteryneighbourhoodcharge>
</battery>
```

Agents

The definition of an agent. The window and shade parameter is the id of the window and shade profile that the agent uses. Office and bedroom are the corresponding locations for activities IT and Sleep. Profile is the activity profile to use, in the example we specify the file used. The other parameters correspond to the activity model.

```
<agents>
...
<agent>
  <shade>1</shade>
  <window>1</window>
  <office>Block1:Zone1</office>
  <power>0.5</power>
  <age>age2</age>
  <computer>computer0</computer>
  <civstat>civstat1</civstat>
  <unemp>unemp0</unemp>
  <retired>retired1</retired>
  <edtry>edtry1</edtry>
  <famstat>famstat3</famstat>
  <sex>sex2</sex>
  <bedroom>Block2:MasterBedroom</bedroom>
  <profile>
    <file>Activity.xml</file>
  </profile>
</agent>
...
</agents>
```

models

External to the building tag are the model tags these specify the possible window or shading models to use. As defined on a per agent basis using ids.

```
<models>
...
<shades>
  <enabled>0</enabled>
  <shade>
    <id>1</id>
  ...
</shade>
...
</shades>
<windows>
  <enabled>0</enabled>
  <window>
    <id>1</id>
  ...
</window>
...
</windows>
...
</models>
```

shades

These are the coefficients as defined by the shading model

```
<shade>
  <id>23</id>
  <name>204-10</name>
  <a01arr>-6.17</a01arr>
  <b01inarr>0.00114</b01inarr>
  <b01sarr>2.17</b01sarr>
  <a10arr>1.2</a10arr>
  <b10inarr>-0.00674</b10inarr>
  <b10sarr>-3.139</b10sarr>
  <a01int>-7.67</a01int>
```

```

<b01inint>0.00088</b01inint>
<b01sint>1.27</b01sint>
<a10int>-2.61</a10int>
<b10inint>-0.0051</b10inint>
<b10sint>-2.683</b10sint>
<afullraise>0.435</afullraise>
<boutfullraise>1.95</boutfullraise>
<bsfullraise>-0.0000231</bsfullraise>
<bsfulllower>0.0000091</bsfulllower>
<boutfulllower>-2.23</boutfulllower>
<afulllower>-0.27</afulllower>
<aSFlower>-2.294</aSFlower>
<bSFlower>1.522</bSFlower>
<shapelower>1.708</shapelower>
</shade>

```

shades

These are the coefficients as defined by the window model.

```

<window>
  <id>1</id>
  <aop>2.151</aop>
  <bopout>0.172</bopout>
  <shapeop>0.418</shapeop>
  <a01arr>-13.88</a01arr>
  <b01inarr>0.312</b01inarr>
  <b01outarr>0.0433</b01outarr>
  <b01absprevarr>1.862</b01absprevarr>
  <b01rnarr>-0.45</b01rnarr>
  <a01int>-12.23</a01int>
  <b01inint>0.281</b01inint>
  <b01outint>0.0271</b01outint>
  <b01presint>-0.000878</b01presint>
  <b01rnint>-0.336</b01rnint>
  <a01dep>-8.75</a01dep>
  <b01outdep>0.1371</b01outdep>
  <b01absdep>0.84</b01absdep>
  <b01gddep>0.83</b01gddep>
  <a10dep>-8.54</a10dep>
  <b10indep>0.213</b10indep>
  <b10outdep>-0.0911</b10outdep>
  <b10absdep>1.614</b10absdep>
  <b10gddep>-0.923</b10gddep>
</window>

```

Chapter 6

IDF File

Process flow for integrating with EnergyPlus

- EnergyPlus starts, reads in the IDF, setups up the Simulation for interfacing with No-MASS
- EnergyPlus extracts the NoMASS.fmu file
- For each run period:
 - Initialise No-MASS
 - For each timestep:
 - * Set the variables in the No-MASS by passing an array of double values and an array of the value references given in the XML file
 - * Calls the No-MASS each step, No-MASS performs its calculations calculations
 - * Return the results of the No-MASS back to EnergyPlus, sends an array of the value references given in the XML file and the value of variable
 - Terminates No-MASS
- End of simulation

Chapter 7

Functional Mockup Unit (FMU)

A Functional Mockup Unit is a program that allows EnergyPlus to couple new custom functionality to EnergyPlus. It is a generic method for allowing simulation programs to be extended. When applied information can be passed from EnergyPlus at runtime to a FMU which can then make calculations and send the results back to EnergyPlus. These results can then be used in EnergyPlus future calculations. For more information on FMUs please see the [EnergyPlus website](#)

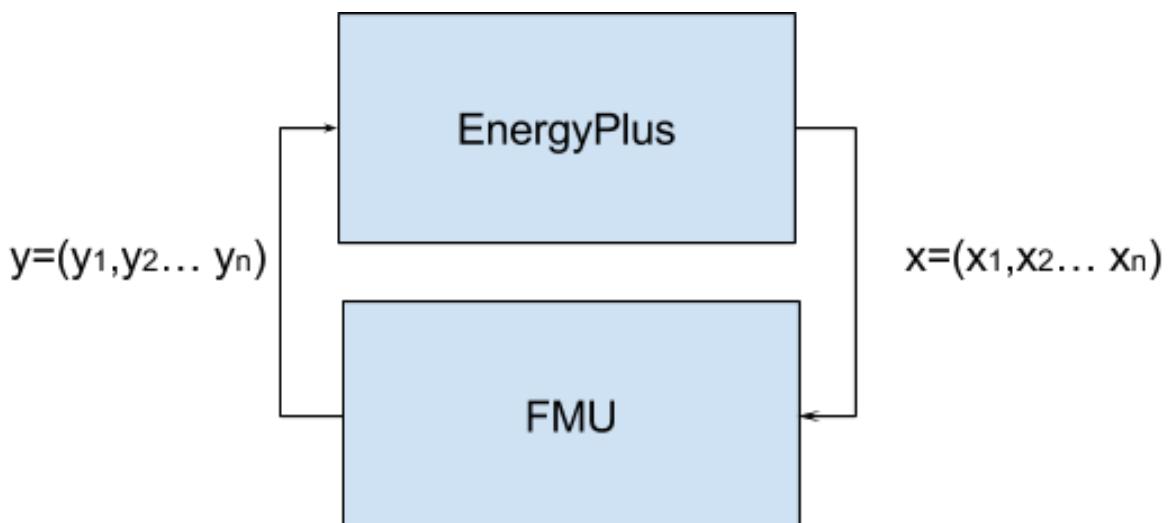
What is a FMU

An FMU consists of three main parts:

- FMU Library (Windows DLL)
- Model Description File
- IDF Code used to enable the FMU

How does an FMU Work

An FMU works using a simple data exchange methodology. At each timestep EnergyPlus sends a set of predefined variables (x) to the FMU (e.g. Zone Mean Air Temperature and/or Site Rain status...). The FMU performs a calculation on the values and returns a set of results (y) to EnergyPlus (e.g. occupant location and/or shade status...). These return values can be made to overwrite EnergyPlus values for the next timestep.



A FMU can be developed to simulate heating setpoints, HVAC controllers or extend EnergyPlus to include any missing functionality.

Linking to EnergyPlus

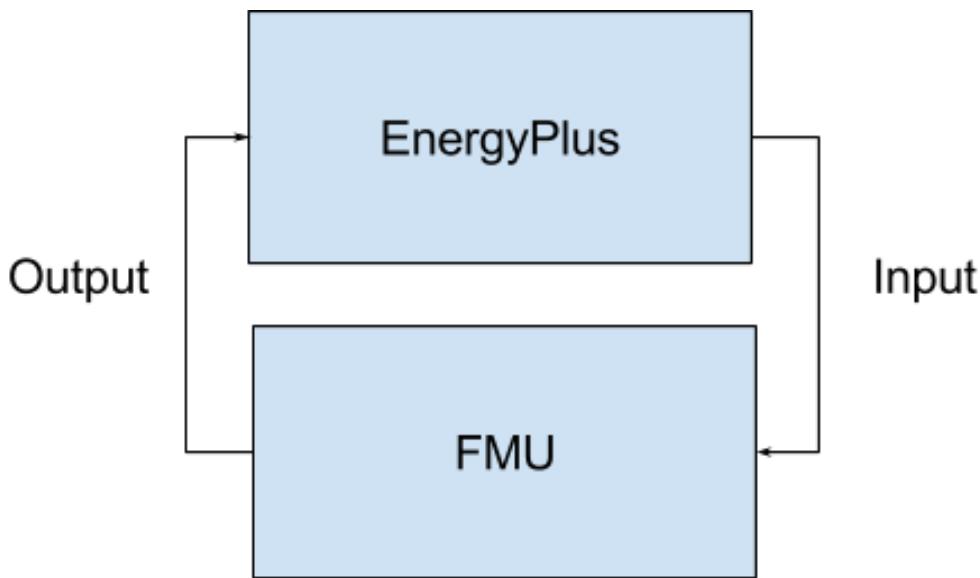
To run a simulation you will also need to run EnergyPlus with the idf file, epw file and the FMU in the same directory

Model description

The model description tab allows you to specify the variables that will be passed between the FMU and EnergyPlus. EnergyPlus and the FMU use this to know what values they should expect and what they should send in return. This is written in XML format, see below.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<fmiModelDescription fmiVersion="1.0"
modelName="ShadingController" <----- 1. Name of the FMU
modelIdentifier="ShadingController" <----- 2. Name of the model within
numberOfContinuousStates="42" <----- 3. Unique GUID
numberOfEventIndicators="42" <----- 4. Start of list of variables
guid="7b2d6d3f-ac4d-4aa8-93eb-d53357dc58ec"> <----- 5. Start of variable
<ModelVariables> <----- 6. Name of variable
  | <ScalarVariable <----- 7. Unique value reference
    name="Block1_Zone1ZoneMeanAirTemperature" <----- 8. Causality of variable
    valueReference="1" <----- 9. Type of variable
    causality="input"> <----- 10. End of variable
    <Real/>
  </ScalarVariable>
</ModelVariables> <----- 11. End of list of variables
</fmiModelDescription>
```

The name of the the FMU(1) and the name of the model(2) need to be kept consistent throughout the dialog and should be provided with the imported FMU. The GUID(2) is a unique string that will also be provided with the FMU if required. The variables to be pass are specified between items 4 and 11. A variable is defined with items 5 to 10. The start of the variable contains the the name associated to it, a unique number and direction the value will be sent. All EnergyPlus variables are of "Real" value so the type of variable will always be real. The name of the variable will correspond to the value given in IDF value under FMU Model Variable Name definition, these should be identical. The value reference should always be unique to the variable and is used by EnergyPlus and the FMU to keep track of the values passed. The causality of the variable specifies the direction the value of the variable will be sent. Shown below:



Using this XML file you will also have to define in the IDF file which values EnergyPlus needs to send.

IDF Script

The IDF script contents for FMU is explained in the EnergyPlus documentation, see [here](#)

The IDF FMU details tell EnergyPlus the values that it needs to know in order to run the FMU program. First we define a ExternalInterface:FunctionalMockupUnitImport which specifies the file name of the FMU in the EnergyPlus folder. This is the name of the FMU plus the ".fmu" file extension.

```

ExternalInterface:FunctionalMockupUnitImport,
  ShadingController.fmu,
  15,
  0;
  1. File Name
  !- FMU Filename
  !- FMU Timeout in milli-seconds
  !- FMU LoggingOn Value

ExternalInterface:FunctionalMockupUnitImport:From:Variable,
  Block1:Zone1,
  Zone Mean Air Temperature,
  ShadingController.fmu,
  ShadingController,
  Block1_Zone1ZoneMeanAirTemperature;
  2. Variable Type
  !- EnergyPlus Key Value
  !- EnergyPlus Variable Name
  !- FMU Filename
  !- FMU Model Name
  !- FMU Model Variable Name
  3. Variable Key
  4. File Name
  5. Model Name
  6. FMU Variable Name

|
ExternalInterface:FunctionalMockupUnitImport:To:Actuator,
  Block1_Zone1_Wall_2_0_0_0_0_0_Win_Shading_Deploy_Status, !- EnergyPlus Variable Name
  Block1:Zone1_Wall_2_0_0_0_0_0_Win,
  Window Shading Control,
  Control Status,
  ShadingController.fmu,
  ShadingController,
  Block1_Zone1_Wall_2_0_0_0_0_0_WinShade,
  6; !- Initial Value
  7. Variable To:
  8. Variable Name
  !- Actuated Component Unique Name
  !- Actuated Component Type
  !- Actuated Component Control Type
  !- FMU Filename
  !- FMU Model Name
  !- FMU Model Variable Name
  9. Actuator Details
  !- Initial Value
  10. Initial Value
  
```

Next we define all the variables that EnergyPlus will send to the FMU as (2). This includes the variable name and key (3), the name of the FMU(4), the model name (5), and the FMU variable name (6). The values for 5 & 6 need to be kept consistent with the values given in the corresponding variables in the model description file.

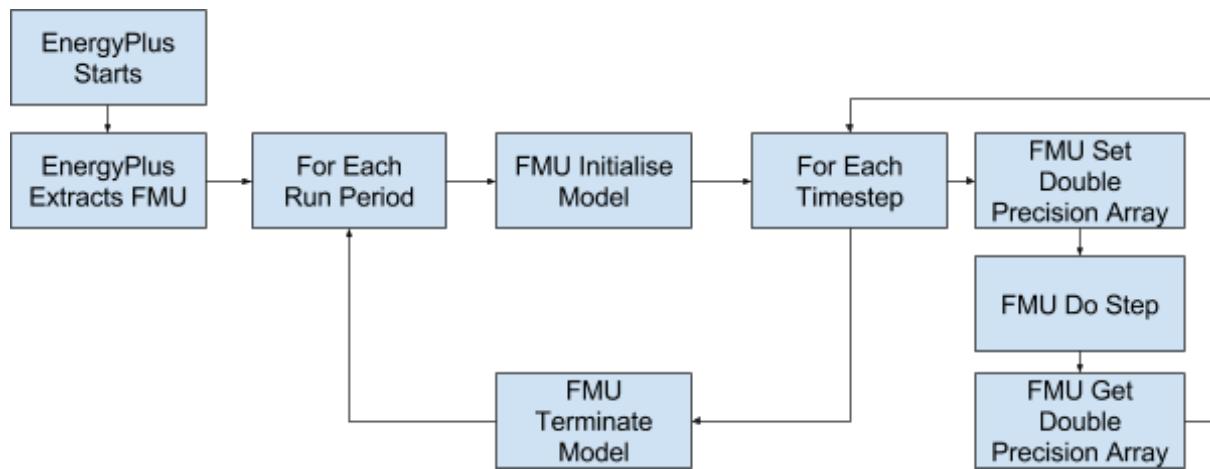
Finally there is the returned value from the FMU, in this case the shading value as an actuator. These are the available external types that can be written to:

- ExternalInterface:FunctionalMockupUnitImport:To:Variable
- ExternalInterface:FunctionalMockupUnitImport:To:Schedule
- ExternalInterface:FunctionalMockupUnitImport:To:Actuator

Developing a FMU

Developing an FMU requires some advance computer science knowledge, first how to program and second how to build that program into a library exported as a DLL in Microsoft Windows. More information about developing FMUs can be found on the [Functional Mockup Interface website](#).

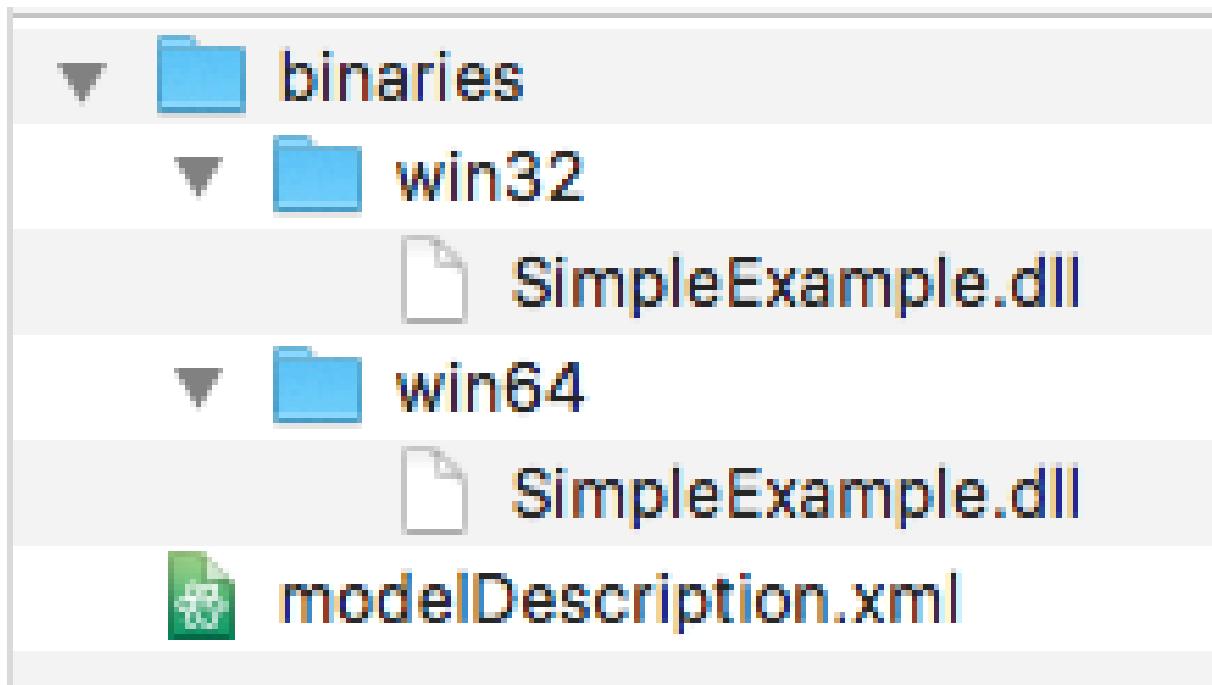
The basic flow for an FMU is as followed:



- EnergyPlus starts, reads in the IDF, setups up the Simulation for FMU usage
- EnergyPlus extracts the .fmu file
- For each run period:
 - Initialise the FMU model by calling the initialise method
 - For each timestep:
 - * Set the variables in the FMU by passing an array of double values and an array of the value references given in the XML file
 - * Calls the FMU do step method, the fmu does any calculations
 - * Return the results of the FMU back to EnergyPlus, sends a preallocated array that is written to and an array of the value references given in the XML file
 - Terminate the FMU by calling the terminate method
- End of simulation

FMU file format

A fmw file such as agentFMU.fmw is just a zipped folder renamed with the from agentFMU.zip to agentFMU.fmw. The zipped folder has the following folder format:



EnergyPlus extracts these files so the model description file can be edited to work with different buildings and configurations. To Debug the DLL with EnergyPlus for testing this format will need to be used. Copy the DLL to the correct binaries folder, zip the folder and copy the renamed fmu to the EnergyPlus folder. Have Visual Studio run EnergyPlus, If everything is correctly setup then Visual Studio will be able to stop the program at breakpoints for inspection.

Chapter 8

Model Description

The model description tab allows you to specify the variables that will be passed between the FMU and EnergyPlus. EnergyPlus and the FMU use this to know what values they should expect and what they should send in return.

This is written in XML format, see below.

```
<fmiModelDescription description="Model with interfaces for media with moist air that will be linked to the  
BCVTB which models the response of the room" fmiVersion="1.0" generationDateAndTime="2012-04-17T19:12:58Z"  
generationTool="Dymola Version 2012 FD01 (32-bit), 2011-11-22" guid="{fd719ef5-c46e-48c7-ae95-96089a69ee64}"  
modelIdentifier="FMI" modelName="FMI" numberOfContinuousStates="0" numberOfEventIndicators="0"  
variableNamingConvention="structured" version="1.2">  
<TypeDefinitions>  
    <Type name="Modelica.Blocks.Interfaces.RealInput">  
        <RealType />  
    </Type>  
    <Type name="Modelica.Blocks.Interfaces.RealOutput">  
        <RealType />  
    </Type>  
</TypeDefinitions>  
<DefaultExperiment startTime="0.0" stopTime="1.0" tolerance="1E-005" />  
<ModelVariables>  
    <ScalarVariable causality="input" name="Block1:Zone1ZoneAirRelativeHumidity" valueReference="1">  
        <Real declaredType="Modelica.Blocks.Interfaces.RealInput" start="0.0" />  
    </ScalarVariable>  
    <ScalarVariable causality="input" name="Block1:Zone1ZoneMeanRadiantTemperature" valueReference="2">  
        <Real declaredType="Modelica.Blocks.Interfaces.RealInput" start="0.0" />  
    </ScalarVariable>  
    <ScalarVariable causality="output" name="Block1:Zone1BlindFraction" valueReference="3">  
        <Real declaredType="Modelica.Blocks.Interfaces.RealInput" start="1.0" />  
    </ScalarVariable>  
    ...  
</ModelVariables>  
</fmiModelDescription>
```

ScalarVariable

A ScalarVariable declares the parameter that will be passed

Name

The name of the variable gives the program understanding as to the parameter being passed, NoMass expects these to be declared in a specific way depending on how they are used by a model.

ValueReference

The value reference should be unique and links the variable in the code to the specific value. For example, the name will be linked to a the value reference so when the variable is passed from the main program it is passed with its value reference. With this information the variable is known.

Causality

The causality of the variable specifies the direction the value of the variable will be sent.

- Input is into the FMU
- Output is out of the FMU into the main program

Real

For NoMASS the variable is always a real.

declaredType

The declared type is always Modelica.Blocks.Interfaces.RealInput.

start

Start is the initial value the variable will have.

Chapter 9

Scripts For NoMASS

There is a NoMASS.py script that implements an interface for running No-MASS as a standalone program. The NoMASS class can be called like so.

```
from NoMASS import NoMASS
nomass = NoMASS()

nomass.runLocation = "../FMU/build/Simulation/"
nomass.locationOfNoMASS = "../FMU/build/"
nomass.configurationDirectory = home +"/Dropbox/DSM_ABM/Simulations/Configurations/WinterWeek/"
nomass.resultsLocation = "../FMU/build/Results/"
nomass.printInput = True
nomass.numberOfSimulations = 1
nomass.learnUpToSimulation = 0
nomass.simulate()
nomass.deleteLearningData()
```

runLocation

NoMASS runs stochastically and often runs over multiple replicates.

Chapter 10

EnergyPlus Examples

Example 1: Shoe box configuration file

The following example enables No-MASS to run a non-residential building configuration.

- A building is defined
 - Two agents are working in the building and assigned to the only zone
 - Both agents have the same presence profile, have equal power and use the same shade and window model

XML File

```
<?xml version="1.0" encoding="UTF-8"?>
<simulation>
  <seed>0</seed>
  <timeStepsPerHour>12</timeStepsPerHour>
  <beginMonth>1</beginMonth>
  <endMonth>12</endMonth>
  <beginDay>1</beginDay>
  <endDay>31</endDay>
  <learn>0</learn>
  <buildings>
    <building>
      <zone>
        <name>Block1:Zone1</name>
        <activities>IT</activities>
        <groundFloor>1</groundFloor>
        <windowCount>1</windowCount>
      </zone>
      <agents>
        <agent>
          <shade>1</shade>
          <window>1</window>
          <office>Block1:Zone1</office>
          <power>0.5</power>
          <profile>
            <monday>0.021,0.021,0.021,0.021,0.021,0.021,0.025,0.250,0.422,0.309,0.377,0.187,0.375,0.426
,0.396,0.375,0.432,0.084,0.070,0.047,0.039,0.038,0.038</monday>
            <tuesday>0.040,0.038,0.038,0.038,0.038,0.038,0.038,0.046,0.320,0.401,0.325,0.417,0.196,0.372,0.43
5,0.402,0.334,0.435,0.100,0.053,0.044,0.044,0.042,0.042</tuesday>
            <wednesday>0.041,0.041,0.041,0.041,0.041,0.041,0.062,0.342,0.403,0.297,0.342,0.194,0.354,0.
395,0.363,0.336,0.383,0.080,0.043,0.027,0.026,0.026</wednesday>
            <thursday>0.024,0.024,0.024,0.024,0.024,0.024,0.024,0.024,0.029,0.265,0.373,0.271,0.368,0.193,0.361,0.4
02,0.357,0.324,0.367,0.094,0.067,0.032,0.032,0.030,0.030</thursday>
            <friday>0.030,0.029,0.029,0.029,0.029,0.029,0.029,0.055,0.327,0.367,0.304,0.373,0.215,0.317,0.280
,0.239,0.197,0.125,0.066,0.069,0.031,0.029,0.027,0.027</friday>
            <saturday>0.028,0.030,0.029,0.029,0.030,0.029,0.029,0.029,0.030,0.030,0.030,0.035,0.037,0.030,0.032,0.0
41,0.045,0.042,0.034,0.035,0.032,0.027,0.024,0.021,0.021</saturday>
            <sunday>0.021,0.021,0.021,0.021,0.021,0.021,0.021,0.024,0.025,0.032,0.045,0.040,0.041,0.033
,0.032,0.033,0.031,0.031,0.028,0.026,0.027,0.025,0.025</sunday>
          </profile>
        </agent>
      </agents>
    </building>
  </buildings>
</simulation>
```

```

</agent>
<agents>
  <agent>
    <shade>1</shade>
    <window>1</window>
    <office>Block1:Zone1</office>
    <power>0.5</power>
    <profile>
      <monday>0.021,0.021,0.021,0.021,0.021,0.021,0.025,0.250,0.422,0.309,0.377,0.187,0.375,0.426
      ,0.396,0.375,0.432,0.084,0.070,0.047,0.039,0.038,0.038</monday>
      <tuesday>0.040,0.038,0.038,0.038,0.038,0.038,0.046,0.320,0.401,0.325,0.417,0.196,0.372,0.43
      5,0.402,0.334,0.435,0.100,0.053,0.044,0.044,0.042,0.042</tuesday>
      <wednesday>0.041,0.041,0.041,0.041,0.041,0.041,0.041,0.062,0.342,0.403,0.297,0.342,0.194,0.354,0.
      395,0.363,0.336,0.383,0.080,0.043,0.027,0.026,0.026,0.026</wednesday>
      <thursday>0.024,0.024,0.024,0.024,0.024,0.024,0.024,0.029,0.265,0.373,0.271,0.368,0.193,0.361,0.4
      02,0.357,0.324,0.367,0.094,0.067,0.032,0.032,0.030,0.030</thursday>
      <friday>0.030,0.029,0.029,0.029,0.029,0.029,0.029,0.055,0.327,0.367,0.304,0.373,0.215,0.317,0.280
      ,0.239,0.197,0.125,0.066,0.069,0.031,0.029,0.027,0.027</friday>
      <saturday>0.028,0.030,0.029,0.029,0.030,0.029,0.029,0.030,0.030,0.030,0.035,0.037,0.030,0.032,0.0
      41,0.045,0.042,0.034,0.035,0.032,0.027,0.024,0.021,0.021</saturday>
      <sunday>0.021,0.021,0.021,0.021,0.021,0.021,0.021,0.024,0.025,0.032,0.045,0.040,0.041,0.033
      ,0.032,0.033,0.031,0.031,0.028,0.026,0.027,0.025</sunday>
    </profile>
  </agent>
</agents>
</building>
</buildings>
<models>
  <presencePage>1</presencePage>
  <shades>
    <enabled>1</enabled>
    <shade>
      <id>1</id>
      <name>001-28-X</name>
      <a0larr>-6.39</a0larr>
      <b0linarr>0.00141</b0linarr>
      <b0lsarr>2.17</b0lsarr>
      <a10arr>-2.71</a10arr>
      <b10inarr>-0.00364</b10inarr>
      <b10sarr>-3.139</b10sarr>
      <a0lint>-6.66</a0lint>
      <b0linint>0.00086</b0linint>
      <b0lsint>1.27</b0lsint>
      <a10int>-3.87</a10int>
      <b10inint>-0.00358</b10inint>
      <b10sint>-2.683</b10sint>
      <afullraise>0.435</afullraise>
      <boutfullraise>1.95</boutfullraise>
      <bsfullraise>-0.0000231</bsfullraise>
      <bsfulllower>0.00000091</bsfulllower>
      <boutfulllower>-2.23</boutfulllower>
      <afulllower>-0.27</afulllower>
      <aSFlower>-2.294</aSFlower>
      <bSFlower>1.522</bSFlower>
      <shapelower>1.708</shapelower>
    </shade>
  </shades>
  <windows>
    <enabled>1</enabled>
    <window>
      <id>1</id>
      <aop>2.151</aop>
      <bopout>0.172</bopout>
      <shapeop>0.418</shapeop>
      <a0larr>13.70</a0larr>
      <b0linarr>0.308</b0linarr>
      <b0loutarr>0.0395</b0loutarr>
      <b0labsprevarr>1.862</b0labsprevarr>
      <b0lrnarr>-0.43</b0lrnarr>
      <a0lint>-11.78</a0lint>
      <b0linint>0.263</b0linint>
      <b0loutint>0.0394</b0loutint>
      <b0lpresint>-0.0009</b0lpresint>
      <b0lrnint>-0.336</b0rnint>
      <a0ldep>-8.72</a0ldep>
      <b0loutdep>0.1352</b0loutdep>
      <b0labsdep>0.85</b0labsdep>
      <b0lgddep>0.82</b0lgddep>
      <a10dep>-8.68</a10dep>
      <b10indep>0.222</b10indep>
      <b10outdep>-0.0936</b10outdep>
      <b10absdep>1.534</b10absdep>
      <b10gddep>-0.845</b10gddep>
    </window>
  </windows>
  <lights>1</lights>
</models>
</simulation>

```

IDF File

```

Output:Variable,*,Schedule Value,Timestep;
Output:Variable,*,AFN Surface Venting Window or Door Opening Factor,timestep;
Output:Variable,*,Zone Exterior Windows Total Transmitted Beam Solar Radiation Rate,timestep;
Output:Variable,*,Window Shading Fraction,timestep;

Output:Variable,*,Zone People Radiant Heating Rate,timestep;
Output:Variable,*,Zone Air Relative Humidity,timestep;
Output:Variable,*,Zone Mean Radiant Temperature,timestep;
Output:Variable,*,Zone People Occupant Count,timestep;
Output:Variable,*,Daylighting Reference Point 1 Illuminance,timestep;
Output:Variable,*,Zone Mean Radiant Temperature,timestep;
Output:Variable,*,Site Exterior Horizontal Sky Illuminance,timestep;
Output:Variable,*,Site Rain Status,timestep;
Output:Variable,*,Site Outdoor Air Drybulb Temperature,timestep;
Output:Variable,*,Zone Lights Electric Power,Timestep;
Output:Variable,*,Zone Lights Electric Energy,Timestep;
Output:Variable,*,Zone Air System Sensible Heating Energy,Timestep;
Output:Variable,*,Zone Air System Sensible Heating Rate,Timestep;
Output:Variable,*,Zone Air System Sensible Cooling Energy,Timestep;
Output:Variable,*,Zone Air System Sensible Cooling Rate,Timestep;
Output:Variable,*,Zone Mean Air Temperature,Timestep;
Output:Variable, *, Zone Thermal Comfort Fanger Model PMV, Timestep;
Output:Variable, *, Zone Thermal Comfort Fanger Model PPD, Timestep;

Output:Variable,*,Zone People Radiant Heating Rate,monthly;
Output:Variable,*,Zone Air Relative Humidity,monthly;
Output:Variable,*,Zone Mean Radiant Temperature,monthly;
Output:Variable,*,Zone People Occupant Count,monthly;
Output:Variable,*,Daylighting Reference Point 1 Illuminance,monthly;
Output:Variable,*,Zone Mean Radiant Temperature,monthly;
Output:Variable,*,Site Exterior Horizontal Sky Illuminance,monthly;
Output:Variable,*,Site Rain Status,monthly;
Output:Variable,*,Site Outdoor Air Drybulb Temperature,monthly;
Output:Variable,*,Zone Lights Electric Power,monthly;
Output:Variable,*,Zone Lights Electric Energy,monthly;
Output:Variable,*,Zone Air System Sensible Heating Energy,monthly;
Output:Variable,*,Zone Air System Sensible Heating Rate,monthly;
Output:Variable,*,Zone Air System Sensible Cooling Energy,monthly;
Output:Variable,*,Zone Air System Sensible Cooling Rate,monthly;
Output:Variable,*,Zone Mean Air Temperature,monthly;

Output:Variable,*,Zone People Radiant Heating Rate,runperiod;
Output:Variable,*,Zone Air Relative Humidity,runperiod;
Output:Variable,*,Zone Mean Radiant Temperature,runperiod;
Output:Variable,*,Zone People Occupant Count,runperiod;
Output:Variable,*,Daylighting Reference Point 1 Illuminance,runperiod;
Output:Variable,*,Zone Mean Radiant Temperature,runperiod;
Output:Variable,*,Site Exterior Horizontal Sky Illuminance,runperiod;
Output:Variable,*,Site Rain Status,runperiod;
Output:Variable,*,Site Outdoor Air Drybulb Temperature,runperiod;
Output:Variable,*,Zone Lights Electric Power,runperiod;
Output:Variable,*,Zone Lights Electric Energy,runperiod;
Output:Variable,*,Zone Air System Sensible Heating Energy,runperiod;
Output:Variable,*,Zone Air System Sensible Heating Rate,runperiod;
Output:Variable,*,Zone Air System Sensible Cooling Energy,runperiod;
Output:Variable,*,Zone Air System Sensible Cooling Rate,runperiod;
Output:Variable,*,Zone Mean Air Temperature,runperiod;

ExternalInterface,
  FunctionalMockupUnitImport;           !- Name of External Interface

ExternalInterface:FunctionalMockupUnitImport,
  agentFMU.fmu,                         !- FMU Filename
  15,                                     !- FMU Timeout in milli-seconds
  0;                                      !- FMU LoggingOn Value

ExternalInterface:FunctionalMockupUnitImport:From:Variable,
  Block1:Zone1,                           !- EnergyPlus Key Value,
  Zone Air Relative Humidity,             !- EnergyPlus Variable Name,
  agentFMU.fmu,                          !- FMU Filename,
  FMI,                                    !- FMU Instance Name,
  Block1:Zone1ZoneAirRelativeHumidity;   !- FMU Variable Name

ExternalInterface:FunctionalMockupUnitImport:From:Variable,
  Block1:Zone1,                           !- EnergyPlus Key Value,
  Zone Mean Radiant Temperature,          !- EnergyPlus Variable Name,
  agentFMU.fmu,                          !- FMU Filename,
  FMI,                                    !- FMU Instance Name,
  Block1:Zone1ZoneMeanRadiantTemperature; !- FMU Variable Name

```

```

ExternalInterface:FunctionalMockupUnitImport:From:Variable,
Block1:Zone1,
Zone Mean Air Temperature, !- EnergyPlus Key Value,
agentFMU.fmu, !- EnergyPlus Variable Name,
FMI, !- FMU Filename,
Block1:Zone1ZoneMeanAirTemperature; !- FMU Instance Name,
Block1:Zone1ZoneMeanAirTemperature; !- FMU Variable Name

ExternalInterface:FunctionalMockupUnitImport:From:Variable,
Block1:Zone1,
Daylighting Reference Point 1 Illuminance,!- EnergyPlus Variable Name,
agentFMU.fmu, !- FMU Filename,
FMI, !- FMU Instance Name,
Block1:Zone1DaylightingReferencePoint1Illuminance;!- FMU Variable Name

ExternalInterface:FunctionalMockupUnitImport:To:Schedule,
Block1:Zone1NumberOfOccupants, !- EnergyPlus Key Value,
Any Number, !- Schedule Type Limits Names,
agentFMU.fmu, !- FMU Filename,
FMI, !- FMU Model Name,
Block1:Zone1NumberOfOccupants, !- FMU Model Variable Name,
0.0; !- Initial Value

ExternalInterface:FunctionalMockupUnitImport:To:Schedule,
Block1:Zone1Heating, !- EnergyPlus Key Value,
Any Number, !- Schedule Type Limits Names,
agentFMU.fmu, !- FMU Filename,
FMI, !- FMU Model Name,
Block1:Zone1Heating, !- FMU Model Variable Name,
0.0; !- Initial Value

ExternalInterface:FunctionalMockupUnitImport:To:Actuator,
Block1:Zone1_Wall_5_0_0_0_0_Win_Shading_Fraction, !- EnergyPlus Variable Name
Block1:Zone1_Wall_5_0_0_0_0_Win, !- Actuated Component Unique Name
Window Shading Fraction, !- Actuated Component Type
Control Fraction, !- Actuated Component Control Type
agentFMU.fmu, !- FMU Filename
FMI, !- FMU Model Name
Block1:Zone1BlindFraction, !- FMU Model Variable Name
1.0; !- Initial Value!

ExternalInterface:FunctionalMockupUnitImport:To:Schedule,
Block1:Zone1LightState, !- EnergyPlus Key Value,
Any Number, !- Schedule Type Limits Names,
agentFMU.fmu, !- FMU Filename,
FMI, !- FMU Model Name,
Block1:Zone1LightState, !- FMU Model Variable Name,
0.0; !- Initial Value

ExternalInterface:FunctionalMockupUnitImport:To:Schedule,
Block1:Zone1WindowState0, !- EnergyPlus Key Value,
Any Number, !- Schedule Type Limits Names,
agentFMU.fmu, !- FMU Filename,
FMI, !- FMU Model Name,
Block1:Zone1WindowState0, !- FMU Model Variable Name,
0.0; !- Initial Value

ExternalInterface:FunctionalMockupUnitImport:To:Schedule,
Block1:Zone1AverageGains, !- EnergyPlus Key Value,
Any Number, !- Schedule Type Limits Names,
agentFMU.fmu, !- FMU Filename,
FMI, !- FMU Model Name,
Block1:Zone1AverageGains, !- FMU Model Variable Name,
0.0; !- Initial Value

ExternalInterface:FunctionalMockupUnitImport:From:Variable,
Environment, !- EnergyPlus Key Value,
Site Exterior Horizontal Sky Illuminance,!- EnergyPlus Variable Name,
agentFMU.fmu, !- FMU Filename,
FMI, !- FMU Instance Name,
EnvironmentSiteExteriorHorizontalSkyIlluminance;!- FMU Variable Name

ExternalInterface:FunctionalMockupUnitImport:From:Variable,
Environment, !- EnergyPlus Key Value,
Site Rain Status, !- EnergyPlus Variable Name,
agentFMU.fmu, !- FMU Filename,
FMI, !- FMU Instance Name,
EnvironmentSiteRainStatus; !- FMU Variable Name

ExternalInterface:FunctionalMockupUnitImport:From:Variable,
Environment, !- EnergyPlus Key Value,
Site Outdoor Air Drybulb Temperature, !- EnergyPlus Variable Name,
agentFMU.fmu, !- FMU Filename,
FMI, !- FMU Instance Name,
EnvironmentSiteOutdoorAirDrybulbTemperature;!- FMU Variable Name

```

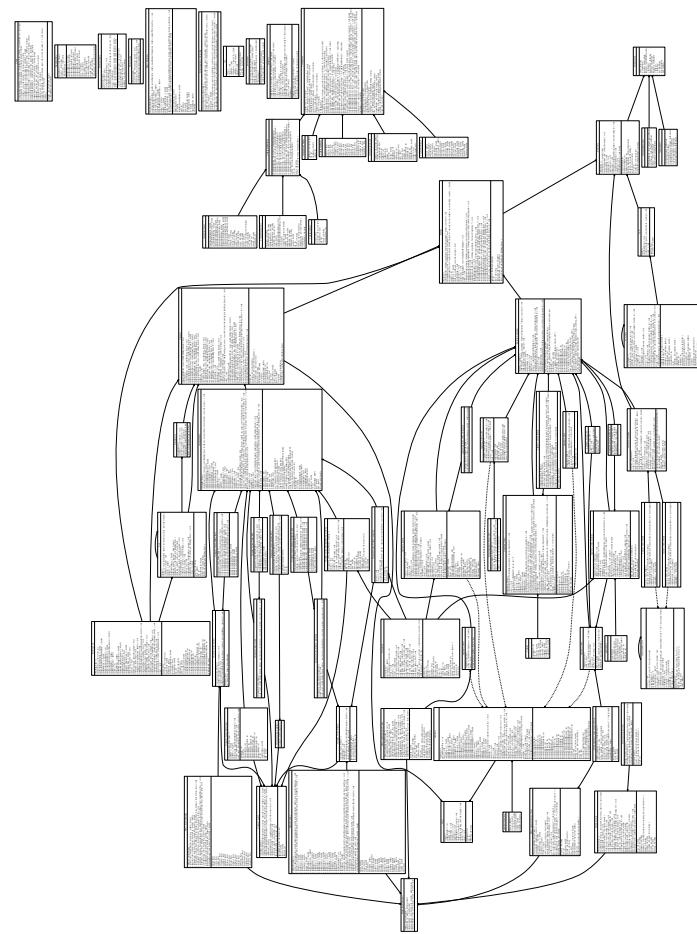
Model Description File

```

<fmiModelDescription description="Model with interfaces for media with moist air that will be linked to the
    BCVTB which models the response of the room" fmiVersion="1.0" generationDateAndTime="2012-04-17T19:12:58Z"
    generationTool="Dymola Version 2012 FD01 (32-bit), 2011-11-22" guid="{fd719ef5-c46e-48c7-ae95-96089a69ee64}"
    modelIdentifier="FMI" modelName="FMI" numberOfContinuousStates="0" numberOfEventIndicators="0"
    variableNamingConvention="structured" version="1.2">
<TypeDefinitions>
    <Type name="Modelica.Blocks.Interfaces.RealInput">
        <RealType />
    </Type>
    <Type name="Modelica.Blocks.Interfaces.RealOutput">
        <RealType />
    </Type>
</TypeDefinitions>
<DefaultExperiment startTime="0.0" stopTime="1.0" tolerance="1E-005" />
<ModelVariables>
    <ScalarVariable causality="input" name="Block1:Zone1ZoneAirRelativeHumidity" valueReference="1">
        <Real declaredType="Modelica.Blocks.Interfaces.RealInput" start="0.0" />
    </ScalarVariable>
    <ScalarVariable causality="input" name="Block1:Zone1ZoneMeanRadiantTemperature" valueReference="2">
        <Real declaredType="Modelica.Blocks.Interfaces.RealInput" start="0.0" />
    </ScalarVariable>
    <ScalarVariable causality="input" name="Block1:Zone1ZoneMeanAirTemperature" valueReference="3">
        <Real declaredType="Modelica.Blocks.Interfaces.RealInput" start="0.0" />
    </ScalarVariable>
    <ScalarVariable causality="input" name="Block1:Zone1DaylightingReferencePoint1Illuminance"
        valueReference="4">
        <Real declaredType="Modelica.Blocks.Interfaces.RealInput" start="0.0" />
    </ScalarVariable>
    <ScalarVariable causality="output" name="Block1:Zone1NumberOfOccupants" valueReference="5">
        <Real declaredType="Modelica.Blocks.Interfaces.RealOutput" start="0.0" />
    </ScalarVariable>
    <ScalarVariable causality="output" name="Block1:Zone1AverageGains" valueReference="6">
        <Real declaredType="Modelica.Blocks.Interfaces.RealOutput" start="0.0" />
    </ScalarVariable>
    <ScalarVariable causality="input" name="EnvironmentSiteExteriorHorizontalSkyIlluminance" valueReference
        ="7">
        <Real declaredType="Modelica.Blocks.Interfaces.RealInput" start="0.0" />
    </ScalarVariable>
    <ScalarVariable causality="input" name="EnvironmentSiteRainStatus" valueReference="8">
        <Real declaredType="Modelica.Blocks.Interfaces.RealInput" start="0.0" />
    </ScalarVariable>
    <ScalarVariable causality="input" name="EnvironmentSiteOutdoorAirDrybulbTemperature" valueReference="9"
        >
        <Real declaredType="Modelica.Blocks.Interfaces.RealInput" start="0.0" />
    </ScalarVariable>
    <ScalarVariable causality="input" name="EMSwarmUpComplete" valueReference="10">
        <Real declaredType="Modelica.Blocks.Interfaces.RealInput" start="0.0" />
    </ScalarVariable>
    <ScalarVariable causality="input" name="EMSepTimeStep" valueReference="11">
        <Real declaredType="Modelica.Blocks.Interfaces.RealInput" start="0.0" />
    </ScalarVariable>
    <ScalarVariable causality="output" name="Block1:Zone1BlindFraction" valueReference="12">
        <Real declaredType="Modelica.Blocks.Interfaces.RealInput" start="1.0" />
    </ScalarVariable>
    <ScalarVariable causality="output" name="Block1:Zone1WindowState0" valueReference="13">
        <Real declaredType="Modelica.Blocks.Interfaces.RealInput" start="0.0" />
    </ScalarVariable>
    <ScalarVariable causality="output" name="Block1:Zone1LightState" valueReference="14">
        <Real declaredType="Modelica.Blocks.Interfaces.RealInput" start="0.0" />
    </ScalarVariable>
</ModelVariables>
</fmiModelDescription>
```


Chapter 11

Class Diagram



Chapter 12

Compiling No-MASS

Build Status

Checkout

Clone the repository using git, ensure you do a recursive clone to receive all dependencies.

- Rapidxml for xml parsing
- googletest for testing

Use the following commands or alternatively use a GUI such as GitHub Desktop or SourceTree.

```
git clone https://github.com/jacoblchapman/No-MASS.git
cd No-MASS
git submodule update --init --recursive
```

Guide to compiling on Linux

Use a compiler with at least the c++11 standard. Most modern linux systems have this by default, if not it can be installed easily. CMake is used to build the make file.

Compiling

Build the No-MASS using cmake and make

```
# Move it the FMU directory
cd FMU

# Create a directory to create the make files in and change to that directory
mkdir build && cd build

# Use cmake to create the make files for your system
cmake ../

# Compile the share library using make
make
```

Testing

To run the tests enable testing in CMake, build, and run the test program

```
# Move into the build directory
cd FMU/build

# Enable debugging and testing in cake
cmake -DCMAKE_BUILD_TYPE=Debug -Dtest=on ../

# Compile using make
make

# Run the tests
./tests/runUnitTests
```

Guide to compiling on Windows

No-MASS can be built on windows using mingw-w64 or alternatively visual studio. This differs for each windows version, but the general process is:

- Ensure you know which EnergyPlus architecture you are targeting 32bit(i686) or 64bit(x86_64)
- Install [mingw-w64](#) or visual studio depending on preferred method of compiling. With mingw-w64 choose the correct architecture.
- Install [CMake](#)
- Use the CMake GUI to create the visual studio project or mingw-w64 make files from the FMU directory in the checked out source folder.
- Open in visual studio to compile or use make command as in linux.

Running

Take the compiled library (FMI.so or FML.dll) and place into the FMU zip file in the correct binary folder, depending on your architecture.

- win64
- win32
- linux64
- linux32

Create model description file and the SimulationConfig file for your simulation. Place the FMU file in the EnergyPlus folder, define the variables in the IDF and run EnergyPlus.

Chapter 13

Hierarchical Index

13.1 Class Hierarchy

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

Agent	45
Appliance	51
Appliance_Battery	77
Appliance_Battery_GridCost_Reward	94
Appliance_FMI	98
Appliance_Generic_CSV	102
Appliance_Large	128
Appliance_Large_CSV	135
Appliance_Large_Learning	140
Appliance_Large_Learning_CSV	148
Appliance_Small	151
Occupant	362
Appliance_Group< T >	107
Appliance_Group_Battery< T >	123
Appliance_Group< Appliance_Battery >	107
Appliance_Group_Battery< Appliance_Battery >	123
Appliance_Group< Appliance_Battery_GridCost_Reward >	107
Appliance_Group_Battery< Appliance_Battery_GridCost_Reward >	123
Appliance_Group< Appliance_FMI >	107
Appliance_Group< Appliance_Generic_CSV >	107
Appliance_Group< Appliance_Large >	107
Appliance_Group< Appliance_Large_CSV >	107
Appliance_Group< Appliance_Large_Learning >	107
Appliance_Group< Appliance_Large_Learning_CSV >	107
Appliance_Group< Appliance_Small >	107
ApplianceParameters	156
Building	158
Building_Appliances	167
Building_Zone	176
ConfigStructAgent	189
ConfigStructAppliance	195
ConfigStructBuilding	200
ConfigStructLVNNNode	203
ConfigStructShade	205

ConfigStructSimulation	209
ConfigStructWindow	214
ConfigStructZone	219
Configuration	221
Contract	228
Contract_Negotiation	231
Contract_Node_Tree< T >	244
Contract_Node_Tree< ContractPtr >	244
Contract_Node_Priority	236
Contract_Node_Supply	240
DataStore	251
Environment	258
fmiCallbackFunctions	261
fmiEventInfo	263
Log	265
LVN	269
LVN_Node	272
Model_Appliance_Ownership	314
Model_Appliance_Power_CSV	317
Model_HeatGains	333
Model_Lights	337
Model_Presence	340
Model_RandomWeibull	343
Model_Activity	281
Model_Activity_Survival	294
Model_Appliance_Large_Usage	298
Model_Appliance_Large_Usage_Survival	308
Model_Appliance_Small_Usage	319
Model_ExternalShading	326
Model_Windows	347
ModellInstance	358
Occupant_Action	375
Occupant_Action_Appliance	382
Occupant_Action_Appliance_BDI	384
Occupant_Action_Heat_Gains	387
Occupant_Action_HeatingSetPoints_Learning	393
Occupant_Action_Lights	399
Occupant_Action_Lights_BDI	403
Occupant_Action_Shades	407
Occupant_Action_Shades_BDI	411
Occupant_Action_Window	417
Occupant_Action_Window_Learning	422
Occupant_Action_Window_Stochastic	427
Occupant_Action_Window_Stochastic_BDI	431
Occupant_Zone	437
profileStruct	447
QLearning	450
Simulation	463
SimulationTime	468
State	474
StateMachine	481
Utility	484

Chapter 14

Class Index

14.1 Class List

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

Agent	
The Agent	45
Appliance	
Super class of each appliance type	51
Appliance_Battery	
Battery Appliance class	77
Appliance_Battery_GridCost_Reward	
Battery Appliance class with reward calculated form the grid	94
Appliance_FMI	
FMI appliances class	98
Appliance_Generic_CSV	
Appliance read in from csv class	102
Appliance_Group< T >	
Groups the types appliances for easy handling	107
Appliance_Group_Battery< T >	
Super class of each appliance type	123
Appliance_Large	
Large appliances class	128
Appliance_Large_CSV	
Large appliances class which uses CSV profiles	135
Appliance_Large_Learning	
Large appliances learning class	140
Appliance_Large_Learning_CSV	
Large appliances learning class with profile taken from CSV	148
Appliance_Small	
Small appliances class	151
ApplianceParameters	
Building	
A Building	158
Building_Appliances	
Manages the different building appliances agents	167
Building_Zone	
A zone within a building	176
ConfigStructAgent	
ConfigStructAppliance	

ConfigStructBuilding	200
ConfigStructLVNNode	203
ConfigStructShade	205
ConfigStructSimulation	209
ConfigStructWindow	214
ConfigStructZone	219
Configuration	
The simulation configuration reader	221
Contract	
A contract submitted from an appliance	228
Contract_Negotiation	
Manages the negotiation between the appliance contracts	231
Contract_Node_Priority	
Tree of contracts sorted by the priority	236
Contract_Node_Supply	
Tree of contracts sorted by the supplied energy left	240
Contract_Node_Tree< T >	
Template class of the contract tree	244
DataStore	
Stores all the data at each timestep	251
Environment	
Stores environmental parameters	258
fmiCallbackFunctions	261
fmiEventInfo	263
Log	
Logs the error messages	265
LVN	
Models the survival function for the activity model	269
LVN_Node	
Low voltage network for power flow analysis	272
Model_Activity	
Models the large appliance with survival	281
Model_Activity_Survival	
Models the survival function for the activity model	294
Model_Appliance_Large_Usage	
Models the large appliance	298
Model_Appliance_Large_Usage_Survival	
Models the large appliance with survival	308
Model_Appliance_Ownership	
Models appliance ownership	314
Model_Appliance_Power_CSV	
Appliance modeled from a CSV file	317
Model_Appliance_Small_Usage	
Models small appliance	319
Model_ExternalShading	
Models the prediction of external shades	326
Model_HeatGains	
Models the occupant heat gains and PMV	333
Model_Lights	
Models the lighting interactions	337
Model_Presence	
Models the Presence of an occupant	340
Model_RandomWeibull	
Models a weibull function	343
Model_Windows	
Models occupant window openings	347
ModelInstance	358

Occupant	
The occupant agent	362
Occupant_Action	
Occupant action super class	375
Occupant_Action_Appliance	
Occupant action on appliances	382
Occupant_Action_Appliance_BDI	
Occupant action on appliances using BDI	384
Occupant_Action_Heat_Gains	
Occupant action of heat gains	387
Occupant_Action_HeatingSetPoints_Learning	
Occupant Action HeatingSetPoints Learning	393
Occupant_Action_Lights	
Occupant action on lights	399
Occupant_Action_Lights_BDI	
Occupant action on lights using BDI	403
Occupant_Action_Shades	
Occupant action on shades	407
Occupant_Action_Shades_BDI	
Occupant action on shades using BDI	411
Occupant_Action_Window	
Occupant action on windows	417
Occupant_Action_Window_Learning	
Occupant action on windows using Q-Learning model	422
Occupant_Action_Window_Stochastic	
Occupant action on windows using stochastic model	427
Occupant_Action_Window_Stochastic_BDI	
Occupant action on windows using BDI	431
Occupant_Zone	
The occupants understanding of a zone	437
profileStruct	
Profile struct	447
QLearning	
The QLearning algorithm	450
Simulation	
Main NoMASS simulation manager	463
SimulationTime	
Keeps track of time	468
State	
The state of an occupant	474
StateMachine	
Moves an agent into a different state	481
Utility	
Utility functions	484

Chapter 15

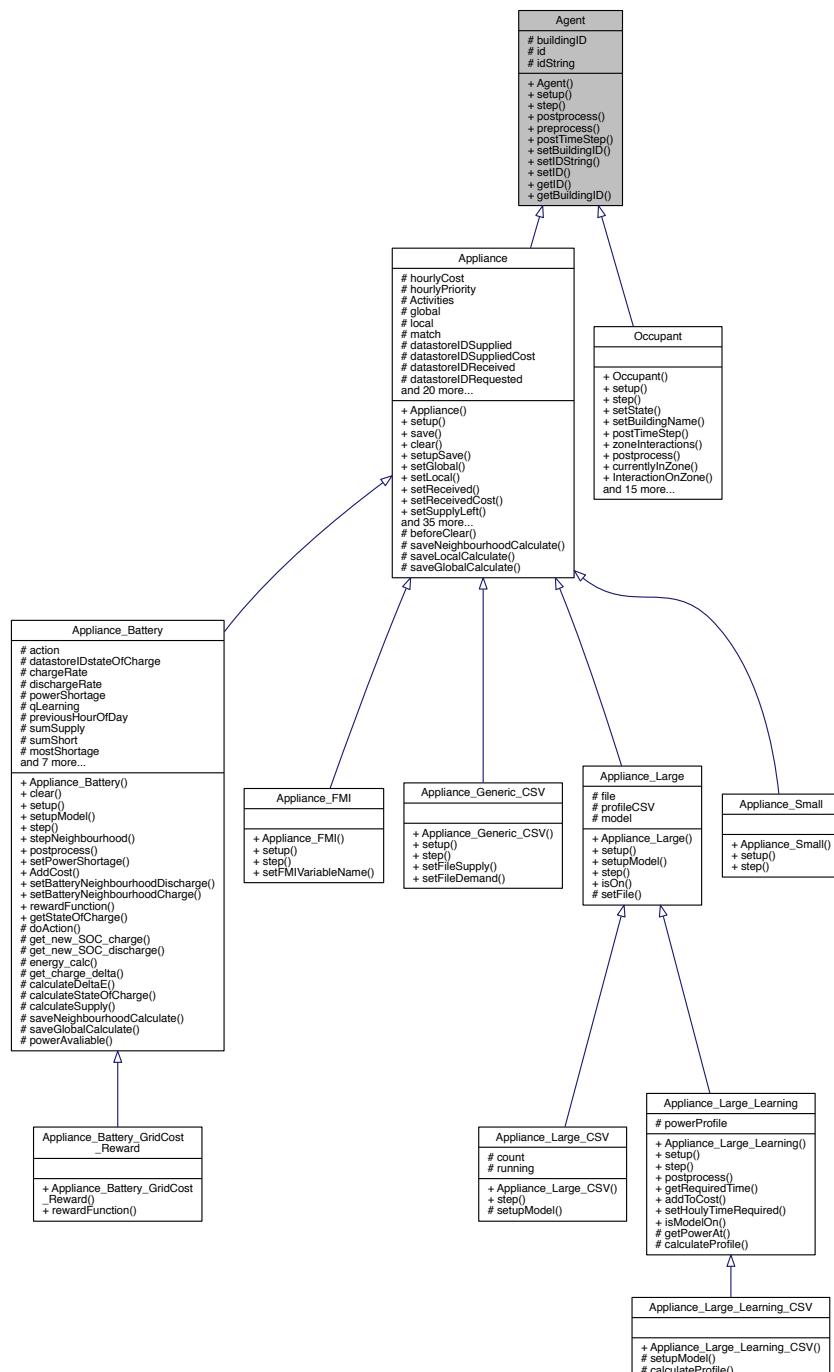
Class Documentation

15.1 Agent Class Reference

The Agent.

```
#include <Agent.hpp>
```

Inheritance diagram for Agent:



Collaboration diagram for Agent:

Agent
buildingID # id # idString
+ Agent() + setup() + step() + postprocess() + preprocess() + postTimeStep() + setBuildingID() + setIdString() + setId() + getID() + getBuildingID()

Public Member Functions

- `Agent ()`
- `void setup ()`
- `void step ()`
- `void postprocess ()`
- `void preprocess ()`
- `void postTimeStep ()`
- `void setBuildingID (const int id)`
- `void setIdString (const std::string &idString)`
- `void setId (const int id)`
- `int getID () const`
- `int getBuildingID () const`

Protected Attributes

- `int buildingID`
- `int id`
- `std::string idString`

15.1.1 Detailed Description

The Agent.

Contains all information about the Agents

Definition at line 13 of file Agent.hpp.

15.1.2 Constructor & Destructor Documentation

15.1.2.1 Agent()

```
Agent::Agent ( )
```

Definition at line 5 of file Agent.cpp.

15.1.3 Member Function Documentation

15.1.3.1 getBuildingID()

```
int Agent::getBuildingID ( ) const
```

Definition at line 29 of file Agent.cpp.

15.1.3.2 getID()

```
int Agent::getID ( ) const
```

Definition at line 25 of file Agent.cpp.

15.1.3.3 postprocess()

```
void Agent::postprocess ( )
```

Definition at line 13 of file Agent.cpp.

15.1.3.4 postTimeStep()

```
void Agent::postTimeStep ( )
```

Definition at line 15 of file Agent.cpp.

15.1.3.5 preprocess()

```
void Agent::preprocess ( )
```

Definition at line 9 of file Agent.cpp.

15.1.3.6 setBuildingID()

```
void Agent::setBuildingID ( const int id )
```

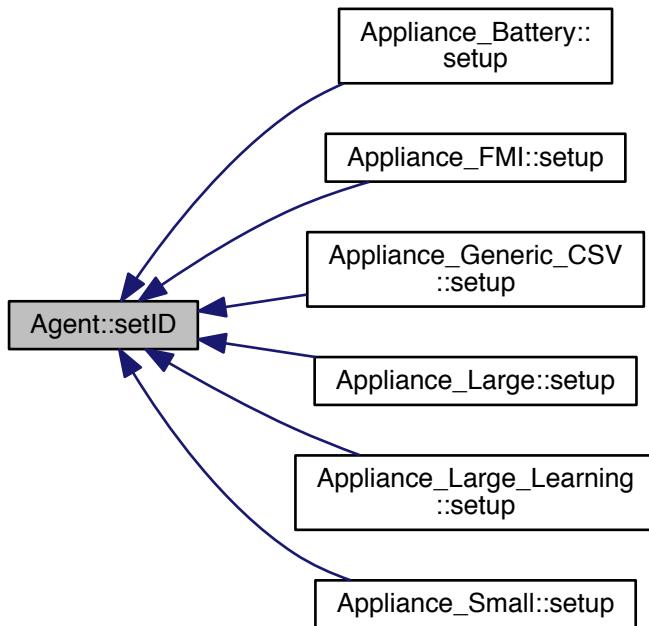
Definition at line 17 of file Agent.cpp.

15.1.3.7 setID()

```
void Agent::setID ( const int id )
```

Definition at line 21 of file Agent.cpp.

Here is the caller graph for this function:



15.1.3.8 setIDString()

```
void Agent::setIDString (  
    const std::string & idString )
```

Definition at line 33 of file Agent.cpp.

15.1.3.9 setup()

```
void Agent::setup ( )
```

Definition at line 7 of file Agent.cpp.

15.1.3.10 step()

```
void Agent::step ( )
```

Definition at line 11 of file Agent.cpp.

15.1.4 Member Data Documentation

15.1.4.1 buildingID

```
int Agent::buildingID [protected]
```

Definition at line 28 of file Agent.hpp.

15.1.4.2 id

```
int Agent::id [protected]
```

Definition at line 29 of file Agent.hpp.

15.1.4.3 idString

```
std::string Agent::idString [protected]
```

Definition at line 30 of file Agent.hpp.

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

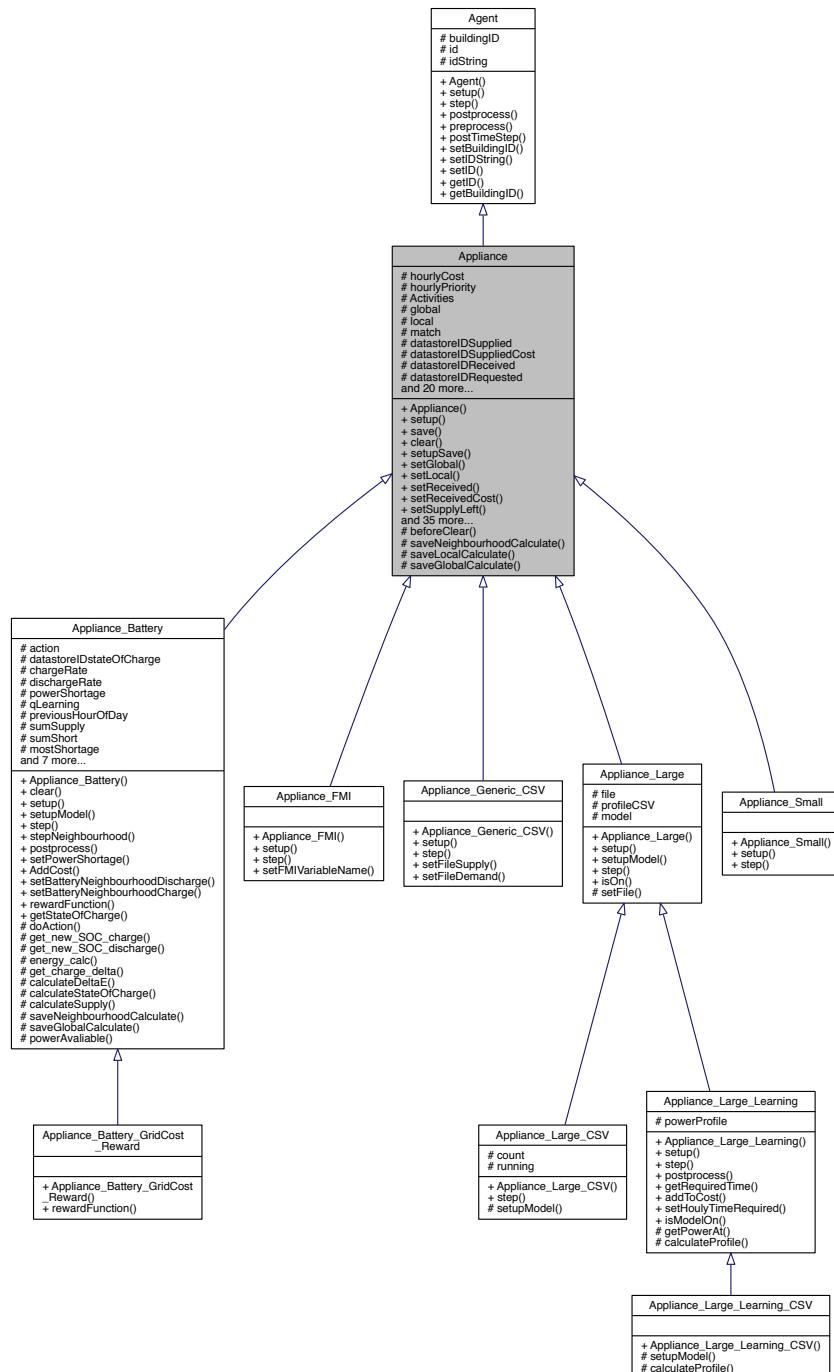
- Agent.hpp
- Agent.cpp

15.2 Appliance Class Reference

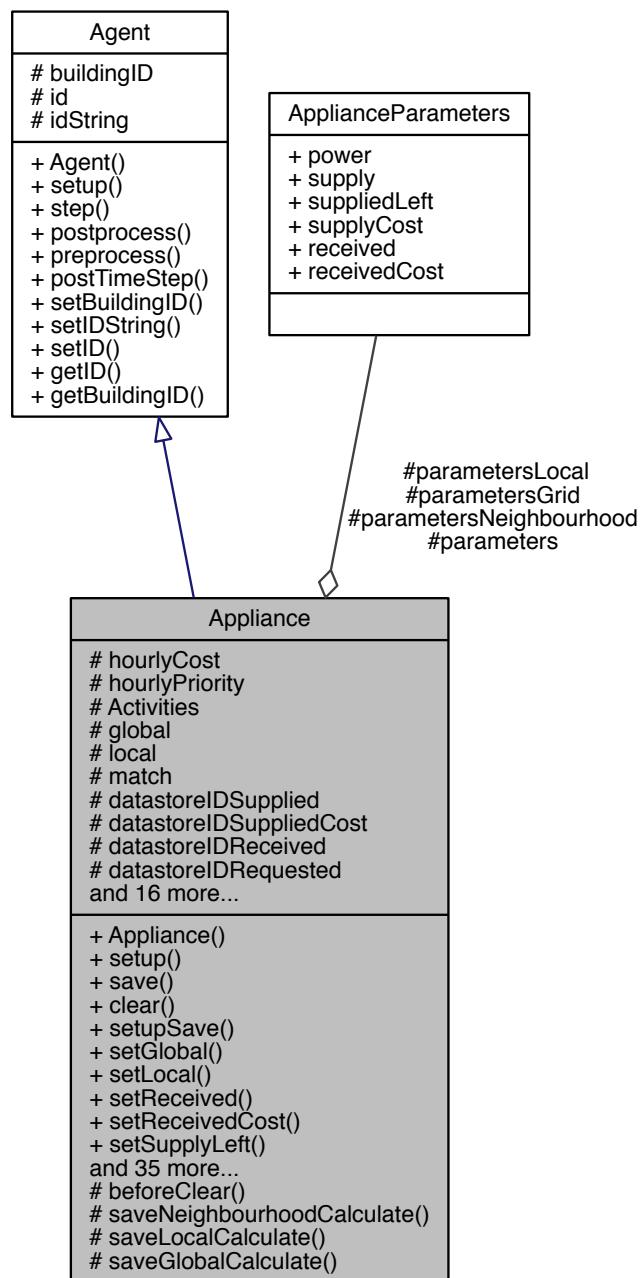
Super class of each appliance type.

```
#include <Appliance.hpp>
```

Inheritance diagram for Appliance:



Collaboration diagram for Appliance:



Public Member Functions

- **Appliance ()**
 - virtual void **setup (ConfigStructAppliance a)=0**
 - void **save ()**
 - void **clear ()**
 - void **setupSave ()**

- void `setGlobal` (bool `global`)
- void `setLocal` (bool `local`)
- void `setReceived` (const double `r`)
- void `setReceivedCost` (const double `c`)
- void `setSupplyLeft` (const double `supplyLeft`)
- void `setPower` (const double `power`)
- void `setSupply` (const double `supply`)
- void `setSupplyCost` (const double `supplyCost`)
- void `setHourlyCost` (const std::vector< double > &`cost`)
- void `setHourlyPriority` (const std::vector< double > &`priority`)
- bool `isGlobal` () const
- bool `isLocal` () const
- double `getSupply` () const
- double `getSupplyLeft` () const
- double `getSupplyCost` () const
- double `getReceived` () const
- double `getReceivedCost` () const
- double `getPriority` () const
- double `getPower` () const
- int `calculateHourOfDay` () const
- void `saveLocal` ()
- void `saveNeighbourhood` ()
- void `saveGlobal` ()
- double `getLocalSupply` () const
- double `getLocalSupplyLeft` () const
- double `getLocalReceived` () const
- double `getLocalReceivedCost` () const
- double `getLocalPower` () const
- double `getNeighbourhoodSupply` () const
- double `getNeighbourhoodSupplyLeft` () const
- double `getNeighbourhoodReceived` () const
- double `getNeighbourhoodReceivedCost` () const
- double `getNeighbourhoodPower` () const
- double `getGridSupply` () const
- double `getGridSupplyLeft` () const
- double `getGridReceived` () const
- double `getGridReceivedCost` () const
- double `getGridPower` () const
- void `setActivities` (const std::vector< int > &`Activities`)
Set the activity for which the appliance can be turned on during.
- bool `hasActivities` (const std::vector< int > &`Activities`)
Check the activities the appliance can be turned on during.

Protected Member Functions

- virtual void `beforeClear` ()
- virtual void `saveNeighbourhoodCalculate` ()
- virtual void `saveLocalCalculate` ()
- virtual void `saveGlobalCalculate` ()

Protected Attributes

- std::vector< double > `hourlyCost`
- std::vector< double > `hourlyPriority`
- std::vector< int > `Activities`
- bool `global`
- bool `local`
- bool `match` = true
- int `datastoreIDSupplied`
- int `datastoreIDSuppliedCost`
- int `datastoreIDReceived`
- int `datastoreIDRequested`
- int `datastoreIDCost`
- int `datastoreLocalIDSupplied`
- int `datastoreLocalIDSuppliedCost`
- int `datastoreLocalIDReceived`
- int `datastoreLocalIDRequested`
- int `datastoreLocalIDCost`
- int `datastoreNeighbourhoodIDSupplied`
- int `datastoreNeighbourhoodIDSuppliedCost`
- int `datastoreNeighbourhoodIDReceived`
- int `datastoreNeighbourhoodIDRequested`
- int `datastoreNeighbourhoodIDCost`
- int `datastoreGridIDSupplied`
- int `datastoreGridIDSuppliedCost`
- int `datastoreGridIDReceived`
- int `datastoreGridIDRequested`
- int `datastoreGridIDCost`
- `ApplianceParameters` `parameters`
- `ApplianceParameters` `parametersLocal`
- `ApplianceParameters` `parametersNeighbourhood`
- `ApplianceParameters` `parametersGrid`

15.2.1 Detailed Description

Super class of each appliance type.

An agent appliance that is used to house helper methods used in the sub classes

Definition at line 28 of file Appliance.hpp.

15.2.2 Constructor & Destructor Documentation

15.2.2.1 Appliance()

```
Appliance::Appliance ( )
```

Definition at line 10 of file Appliance.cpp.

15.2.3 Member Function Documentation

15.2.3.1 beforeClear()

```
void Appliance::beforeClear ( ) [protected], [virtual]
```

Definition at line 107 of file Appliance.cpp.

Here is the caller graph for this function:

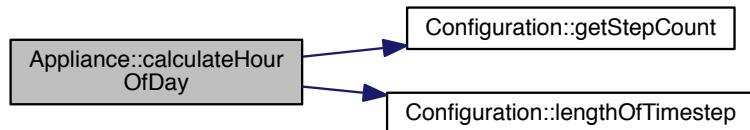


15.2.3.2 calculateHourOfDay()

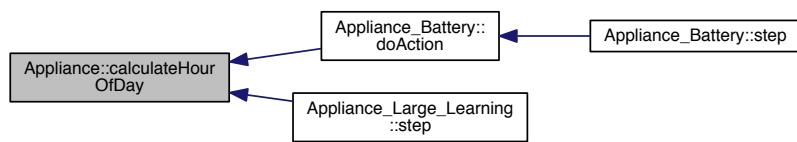
```
int Appliance::calculateHourOfDay ( ) const
```

Definition at line 161 of file Appliance.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.2.3.3 clear()

```
void Appliance::clear ( )
```

Definition at line 110 of file Appliance.cpp.

Here is the call graph for this function:

**15.2.3.4 getGridPower()**

```
double Appliance::getGridPower ( ) const
```

Definition at line 262 of file Appliance.cpp.

15.2.3.5 getGridReceived()

```
double Appliance::getGridReceived ( ) const
```

Definition at line 256 of file Appliance.cpp.

15.2.3.6 getGridReceivedCost()

```
double Appliance::getGridReceivedCost ( ) const
```

Definition at line 259 of file Appliance.cpp.

15.2.3.7 getGridSupply()

```
double Appliance::getGridSupply ( ) const
```

Definition at line 250 of file Appliance.cpp.

15.2.3.8 getGridSupplyLeft()

```
double Appliance::getGridSupplyLeft ( ) const
```

Definition at line 253 of file Appliance.cpp.

15.2.3.9 getLocalPower()

```
double Appliance::getLocalPower ( ) const
```

Definition at line 230 of file Appliance.cpp.

15.2.3.10 getLocalReceived()

```
double Appliance::getLocalReceived ( ) const
```

Definition at line 224 of file Appliance.cpp.

15.2.3.11 getLocalReceivedCost()

```
double Appliance::getLocalReceivedCost ( ) const
```

Definition at line 227 of file Appliance.cpp.

15.2.3.12 getLocalSupply()

```
double Appliance::getLocalSupply ( ) const
```

Definition at line 218 of file Appliance.cpp.

15.2.3.13 getLocalSupplyLeft()

```
double Appliance::getLocalSupplyLeft ( ) const
```

Definition at line 221 of file Appliance.cpp.

15.2.3.14 getNeighbourhoodPower()

```
double Appliance::getNeighbourhoodPower ( ) const
```

Definition at line 246 of file Appliance.cpp.

15.2.3.15 getNeighbourhoodReceived()

```
double Appliance::getNeighbourhoodReceived ( ) const
```

Definition at line 240 of file Appliance.cpp.

15.2.3.16 getNeighbourhoodReceivedCost()

```
double Appliance::getNeighbourhoodReceivedCost ( ) const
```

Definition at line 243 of file Appliance.cpp.

15.2.3.17 getNeighbourhoodSupply()

```
double Appliance::getNeighbourhoodSupply ( ) const
```

Definition at line 234 of file Appliance.cpp.

15.2.3.18 getNeighbourhoodSupplyLeft()

```
double Appliance::getNeighbourhoodSupplyLeft ( ) const
```

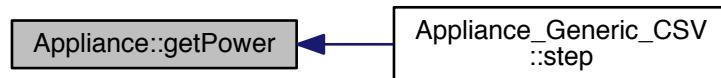
Definition at line 237 of file Appliance.cpp.

15.2.3.19 getPower()

```
double Appliance::getPower ( ) const
```

Definition at line 206 of file Appliance.cpp.

Here is the caller graph for this function:

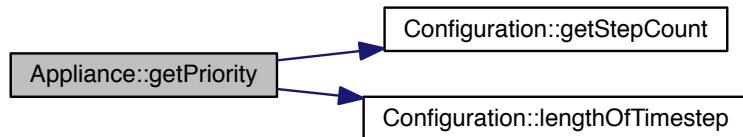


15.2.3.20 getPriority()

```
double Appliance::getPriority ( ) const
```

Definition at line 12 of file Appliance.cpp.

Here is the call graph for this function:

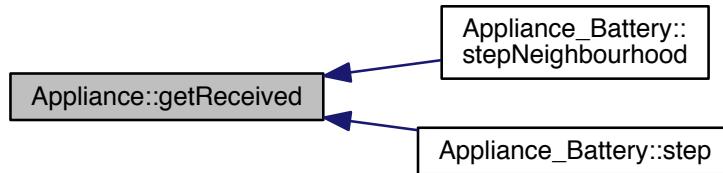


15.2.3.21 getReceived()

```
double Appliance::getReceived ( ) const
```

Definition at line 210 of file Appliance.cpp.

Here is the caller graph for this function:

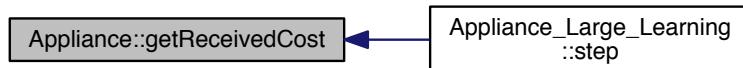


15.2.3.22 getReceivedCost()

```
double Appliance::getReceivedCost ( ) const
```

Definition at line 214 of file Appliance.cpp.

Here is the caller graph for this function:

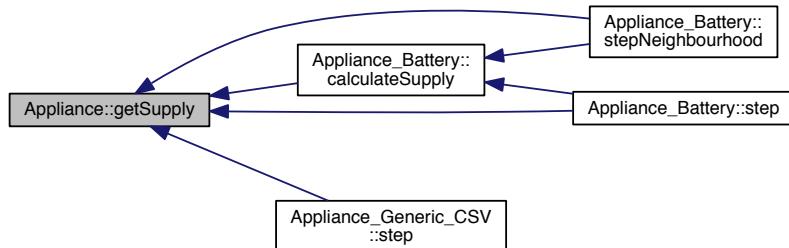


15.2.3.23 getSupply()

```
double Appliance::getSupply ( ) const
```

Definition at line 194 of file Appliance.cpp.

Here is the caller graph for this function:



15.2.3.24 getSupplyCost()

```
double Appliance::getSupplyCost ( ) const
```

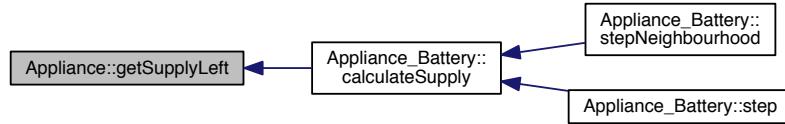
Definition at line 202 of file Appliance.cpp.

15.2.3.25 getSupplyLeft()

```
double Appliance::getSupplyLeft ( ) const
```

Definition at line 198 of file Appliance.cpp.

Here is the caller graph for this function:



15.2.3.26 hasActivities()

```
bool Appliance::hasActivities (
    const std::vector< int > & Activities )
```

Check the activities the appliance can be turned on during.

Returns

If there is a corresponding activity to turn the appliance on in.

For some appliances we may wish that they only turn on for some appliances, here we check if the activities match.

Definition at line 173 of file Appliance.cpp.

15.2.3.27 isGlobal()

```
bool Appliance::isGlobal ( ) const
```

Definition at line 130 of file Appliance.cpp.

15.2.3.28 isLocal()

```
bool Appliance::isLocal ( ) const
```

Definition at line 122 of file Appliance.cpp.

15.2.3.29 save()

```
void Appliance::save ( )
```

Definition at line 52 of file Appliance.cpp.

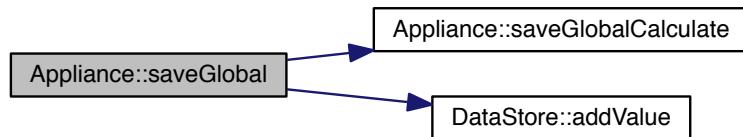
Here is the call graph for this function:

**15.2.3.30 saveGlobal()**

```
void Appliance::saveGlobal ( )
```

Definition at line 99 of file Appliance.cpp.

Here is the call graph for this function:



15.2.3.31 saveGlobalCalculate()

```
void Appliance::saveGlobalCalculate ( ) [protected], [virtual]
```

Reimplemented in [Appliance_Battery](#).

Definition at line 91 of file Appliance.cpp.

Here is the caller graph for this function:

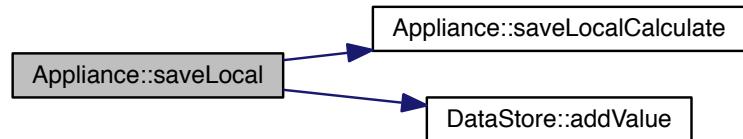


15.2.3.32 saveLocal()

```
void Appliance::saveLocal ( )
```

Definition at line 67 of file Appliance.cpp.

Here is the call graph for this function:



15.2.3.33 saveLocalCalculate()

```
void Appliance::saveLocalCalculate ( ) [protected], [virtual]
```

Definition at line 59 of file Appliance.cpp.

Here is the caller graph for this function:

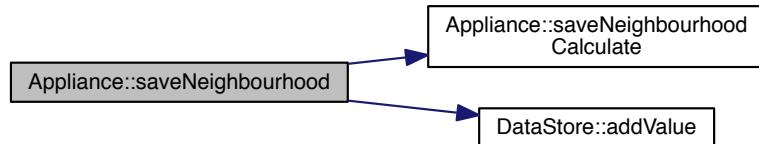


15.2.3.34 saveNeighbourhood()

```
void Appliance::saveNeighbourhood ( )
```

Definition at line 83 of file Appliance.cpp.

Here is the call graph for this function:



15.2.3.35 saveNeighbourhoodCalculate()

```
void Appliance::saveNeighbourhoodCalculate ( ) [protected], [virtual]
```

Reimplemented in [Appliance_Battery](#).

Definition at line 75 of file Appliance.cpp.

Here is the caller graph for this function:



15.2.3.36 setActivities()

```
void Appliance::setActivities (
    const std::vector< int > & Activities )
```

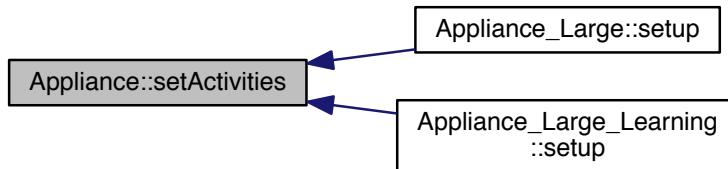
Set the activity for which the appliance can be turned on during.

Parameters

Activities	A set of activities
------------	---------------------

Definition at line 190 of file Appliance.cpp.

Here is the caller graph for this function:



15.2.3.37 setGlobal()

```
void Appliance::setGlobal (
    bool global )
```

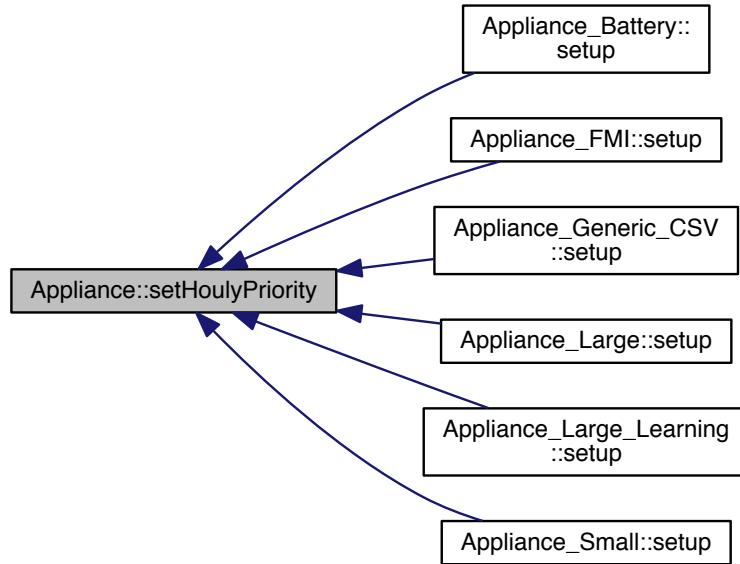
Definition at line 134 of file Appliance.cpp.

15.2.3.38 setHourlyPriority()

```
void Appliance::setHourlyPriority (
    const std::vector< double > & priority )
```

Definition at line 153 of file Appliance.cpp.

Here is the caller graph for this function:

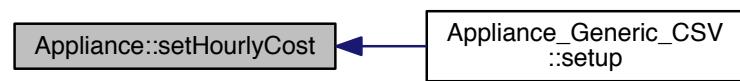


15.2.3.39 setHourlyCost()

```
void Appliance::setHourlyCost ( const std::vector< double > & cost )
```

Definition at line 149 of file Appliance.cpp.

Here is the caller graph for this function:



15.2.3.40 setLocal()

```
void Appliance::setLocal (  
    bool local )
```

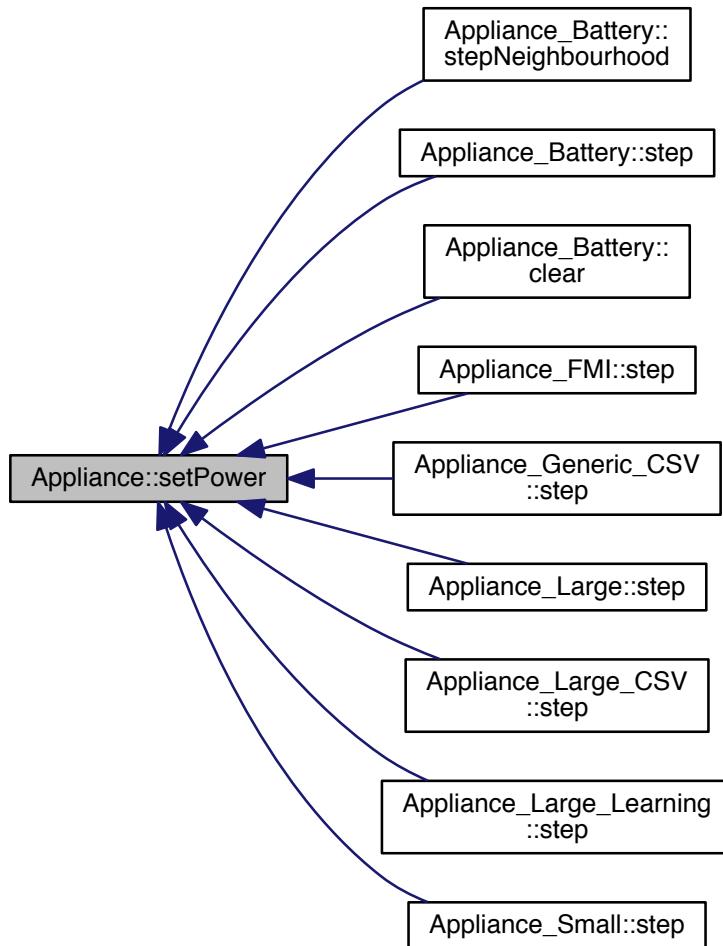
Definition at line 126 of file Appliance.cpp.

15.2.3.41 setPower()

```
void Appliance::setPower (   
    const double power )
```

Definition at line 137 of file Appliance.cpp.

Here is the caller graph for this function:



15.2.3.42 setReceived()

```
void Appliance::setReceived (const double r)
```

Definition at line 21 of file Appliance.cpp.

Here is the caller graph for this function:

**15.2.3.43 setReceivedCost()**

```
void Appliance::setReceivedCost (const double c)
```

Definition at line 25 of file Appliance.cpp.

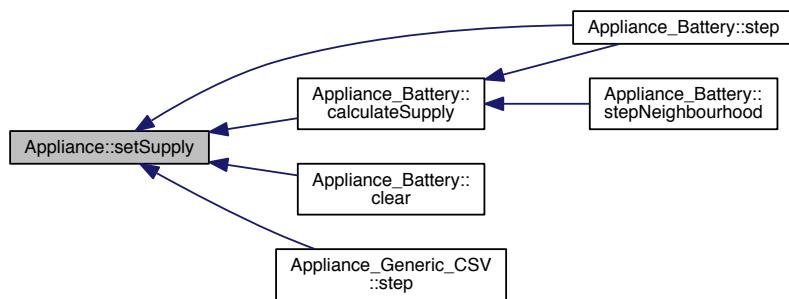
Here is the caller graph for this function:

**15.2.3.44 setSupply()**

```
void Appliance::setSupply (const double supply)
```

Definition at line 141 of file Appliance.cpp.

Here is the caller graph for this function:

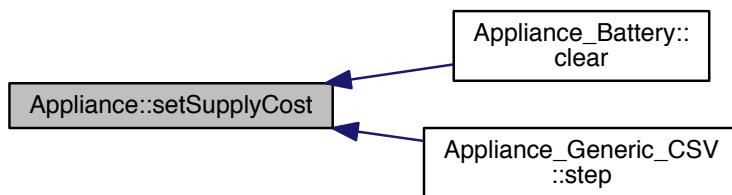


15.2.3.45 setSupplyCost()

```
void Appliance::setSupplyCost (
    const double supplyCost )
```

Definition at line 145 of file Appliance.cpp.

Here is the caller graph for this function:

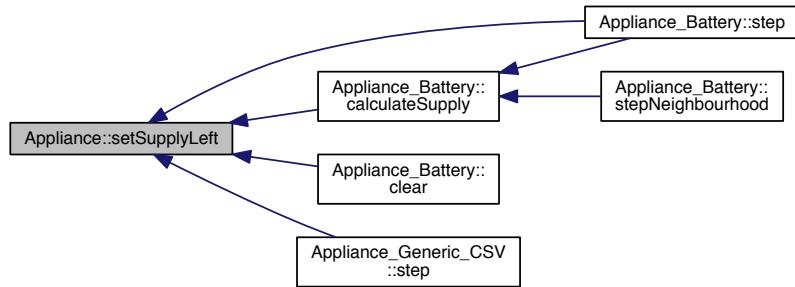


15.2.3.46 setSupplyLeft()

```
void Appliance::setSupplyLeft (
    const double supplyLeft )
```

Definition at line 157 of file Appliance.cpp.

Here is the caller graph for this function:



15.2.3.47 setup()

```
virtual void Appliance::setup (
    ConfigStructAppliance a ) [pure virtual]
```

Implemented in [Appliance_Large_Learning](#), [Appliance_Battery](#), [Appliance_Small](#), [Appliance_Generic_CSV](#), [Appliance_Large](#), and [Appliance_FMI](#).

15.2.3.48 setupSave()

```
void Appliance::setupSave ( )
```

Definition at line 29 of file `Appliance.cpp`.

Here is the call graph for this function:



15.2.4 Member Data Documentation

15.2.4.1 Activities

```
std::vector<int> Appliance::Activities [protected]
```

Definition at line 90 of file Appliance.hpp.

15.2.4.2 datastoreGridIDCost

```
int Appliance::datastoreGridIDCost [protected]
```

Definition at line 116 of file Appliance.hpp.

15.2.4.3 datastoreGridIDReceived

```
int Appliance::datastoreGridIDReceived [protected]
```

Definition at line 114 of file Appliance.hpp.

15.2.4.4 datastoreGridIDRequested

```
int Appliance::datastoreGridIDRequested [protected]
```

Definition at line 115 of file Appliance.hpp.

15.2.4.5 datastoreGridIDSupplied

```
int Appliance::datastoreGridIDSupplied [protected]
```

Definition at line 112 of file Appliance.hpp.

15.2.4.6 datastoreGridIDSuppliedCost

```
int Appliance::datastoreGridIDSuppliedCost [protected]
```

Definition at line 113 of file Appliance.hpp.

15.2.4.7 `datastoreIDCost`

```
int Appliance::datastoreIDCost [protected]
```

Definition at line 100 of file Appliance.hpp.

15.2.4.8 `datastoreIDReceived`

```
int Appliance::datastoreIDReceived [protected]
```

Definition at line 98 of file Appliance.hpp.

15.2.4.9 `datastoreIDRequested`

```
int Appliance::datastoreIDRequested [protected]
```

Definition at line 99 of file Appliance.hpp.

15.2.4.10 `datastoreIDSupplied`

```
int Appliance::datastoreIDSupplied [protected]
```

Definition at line 96 of file Appliance.hpp.

15.2.4.11 `datastoreIDSuppliedCost`

```
int Appliance::datastoreIDSuppliedCost [protected]
```

Definition at line 97 of file Appliance.hpp.

15.2.4.12 `datastoreLocalIDCost`

```
int Appliance::datastoreLocalIDCost [protected]
```

Definition at line 106 of file Appliance.hpp.

15.2.4.13 `datastoreLocalIDReceived`

```
int Appliance::datastoreLocalIDReceived [protected]
```

Definition at line 104 of file `Appliance.hpp`.

15.2.4.14 `datastoreLocalIDRequested`

```
int Appliance::datastoreLocalIDRequested [protected]
```

Definition at line 105 of file `Appliance.hpp`.

15.2.4.15 `datastoreLocalIDSupplied`

```
int Appliance::datastoreLocalIDSupplied [protected]
```

Definition at line 102 of file `Appliance.hpp`.

15.2.4.16 `datastoreLocalIDSuppliedCost`

```
int Appliance::datastoreLocalIDSuppliedCost [protected]
```

Definition at line 103 of file `Appliance.hpp`.

15.2.4.17 `datastoreNeighbourhoodIDCost`

```
int Appliance::datastoreNeighbourhoodIDCost [protected]
```

Definition at line 111 of file `Appliance.hpp`.

15.2.4.18 `datastoreNeighbourhoodIDReceived`

```
int Appliance::datastoreNeighbourhoodIDReceived [protected]
```

Definition at line 109 of file `Appliance.hpp`.

15.2.4.19 `datastoreNeighbourhoodIDRequested`

```
int Appliance::datastoreNeighbourhoodIDRequested [protected]
```

Definition at line 110 of file `Appliance.hpp`.

15.2.4.20 `datastoreNeighbourhoodIDSupplied`

```
int Appliance::datastoreNeighbourhoodIDSupplied [protected]
```

Definition at line 107 of file `Appliance.hpp`.

15.2.4.21 `datastoreNeighbourhoodIDSuppliedCost`

```
int Appliance::datastoreNeighbourhoodIDSuppliedCost [protected]
```

Definition at line 108 of file `Appliance.hpp`.

15.2.4.22 `global`

```
bool Appliance::global [protected]
```

Definition at line 92 of file `Appliance.hpp`.

15.2.4.23 `hourlyCost`

```
std::vector<double> Appliance::hourlyCost [protected]
```

Definition at line 88 of file `Appliance.hpp`.

15.2.4.24 `hourlyPriority`

```
std::vector<double> Appliance::hourlyPriority [protected]
```

Definition at line 89 of file `Appliance.hpp`.

15.2.4.25 local

```
bool Appliance::local [protected]
```

Definition at line 93 of file Appliance.hpp.

15.2.4.26 match

```
bool Appliance::match = true [protected]
```

Definition at line 94 of file Appliance.hpp.

15.2.4.27 parameters

```
ApplianceParameters Appliance::parameters [protected]
```

Definition at line 118 of file Appliance.hpp.

15.2.4.28 parametersGrid

```
ApplianceParameters Appliance::parametersGrid [protected]
```

Definition at line 121 of file Appliance.hpp.

15.2.4.29 parametersLocal

```
ApplianceParameters Appliance::parametersLocal [protected]
```

Definition at line 119 of file Appliance.hpp.

15.2.4.30 parametersNeighbourhood

```
ApplianceParameters Appliance::parametersNeighbourhood [protected]
```

Definition at line 120 of file Appliance.hpp.

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

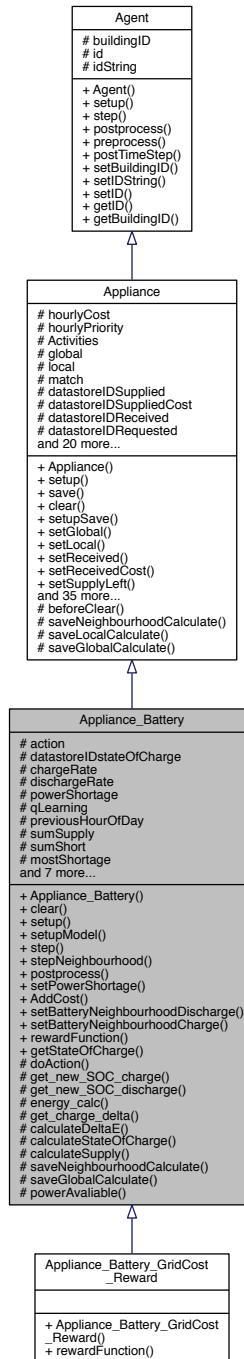
- Appliance.hpp
- Appliance.cpp

15.3 Appliance_Battery Class Reference

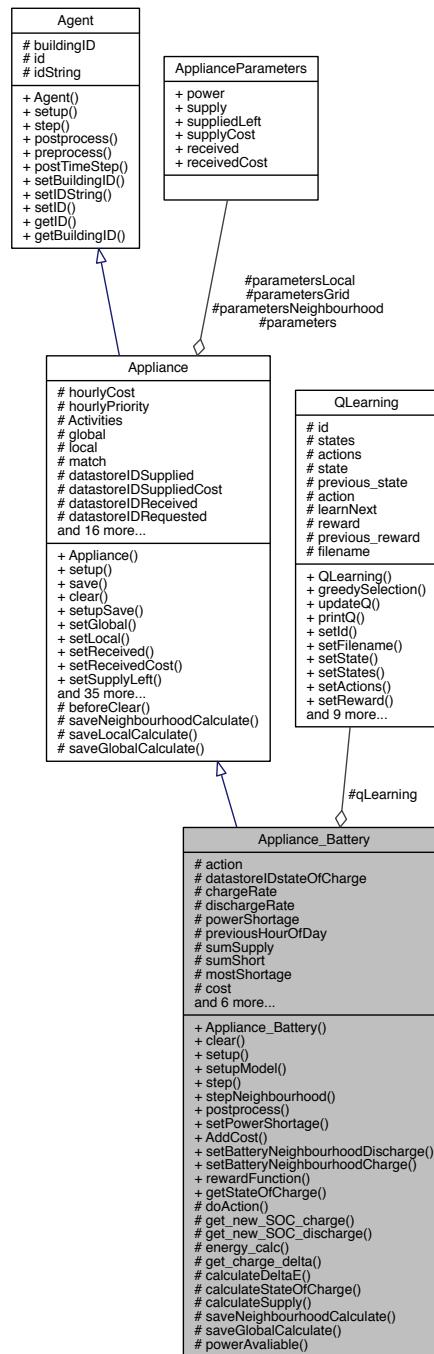
Battery Appliance class.

```
#include <Appliance_Battery.hpp>
```

Inheritance diagram for Appliance_Battery:



Collaboration diagram for Appliance_Battery:



Public Member Functions

- [Appliance_Battery \(\)](#)
 - void [clear \(\)](#)
 - void [setup \(ConfigStructAppliance a\)](#)
- Set up the large appliance model, reading in the large appliance configuration file.*
- void [setupModel \(\)](#)

Set up the large appliance model, reading in the large appliance configuration file.

- void **step** ()
The timestep call of the battery appliance.
- void **stepNeighbourhood** ()
The timestep call of the battery appliance.
- void **postprocess** ()
Appliance postprocess, calls Q-Learning save data.
- void **setPowerShortage** (double power)
- void **AddCost** (double cost)
- void **setBatteryNeighbourhoodDischarge** (bool batteryNeighbourhoodDischarge)
- void **setBatteryNeighbourhoodCharge** (bool batteryNeighbourhoodCharge)
- virtual double **rewardFunction** (double mostShortage, double binShortage) const
- double **getStateOfCharge** () const

Protected Member Functions

- void **doAction** ()
- void **get_new_SOC_charge** (double P_request)
- double **get_new_SOC_discharge** (double P_request)
- double **energy_calc** () const
- double **get_charge_delta** () const
- double **calculateDeltaE** (double P_request) const
- void **calculateStateOfCharge** (double energy)
- void **calculateSupply** ()
- void **saveNeighbourhoodCalculate** ()
- void **saveGlobalCalculate** ()
- double **powerAvailable** (const double energy) const

Protected Attributes

- bool **action**
- int **datastoreIDstateOfCharge**
- double **chargeRate** = 1000
- double **dischargeRate** = 1000
- double **powerShortage**
- **QLearning** qLearning
- int **previousHourOfDay**
- double **sumSupply**
- double **sumShort**
- double **mostShortage**
- double **cost**
- double **capacity** = 2000
- double **efficiency** = 0.98
- double **stateOfCharge**
- double **BatteryDeltaT**
- bool **batteryNeighbourhoodDischarge**
- bool **batteryNeighbourhoodCharge**

15.3.1 Detailed Description

Battery Appliance class.

The Battery appliance agent

Definition at line 15 of file Appliance_Battery.hpp.

15.3.2 Constructor & Destructor Documentation

15.3.2.1 Appliance_Battery()

```
Appliance_Battery::Appliance_Battery ( )
```

Definition at line 10 of file Appliance_Battery.cpp.

15.3.3 Member Function Documentation

15.3.3.1 AddCost()

```
void Appliance_Battery::AddCost (
    double cost )
```

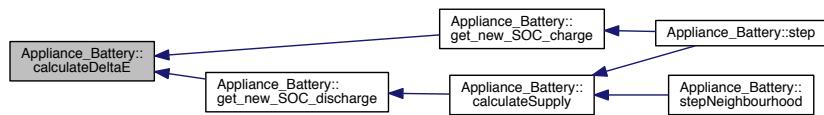
Definition at line 196 of file Appliance_Battery.cpp.

15.3.3.2 calculateDeltaE()

```
double Appliance_Battery::calculateDeltaE (
    double P_request ) const [protected]
```

Definition at line 209 of file Appliance_Battery.cpp.

Here is the caller graph for this function:

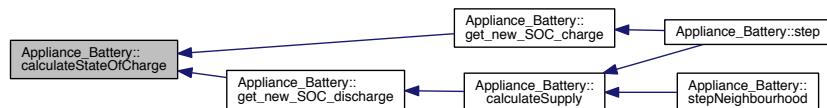


15.3.3.3 calculateStateOfCharge()

```
void Appliance_Battery::calculateStateOfCharge (
    double energy ) [protected]
```

Definition at line 214 of file Appliance_Battery.cpp.

Here is the caller graph for this function:

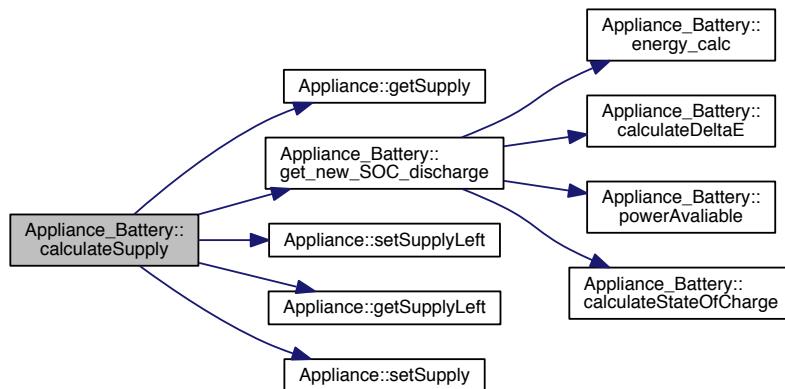


15.3.3.4 calculateSupply()

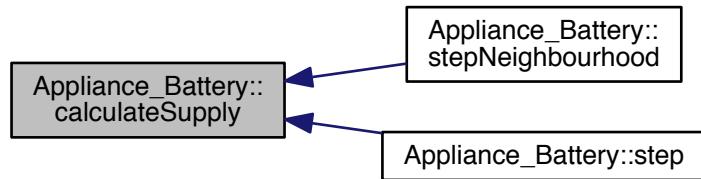
```
void Appliance_Battery::calculateSupply ( ) [protected]
```

Definition at line 152 of file Appliance_Battery.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

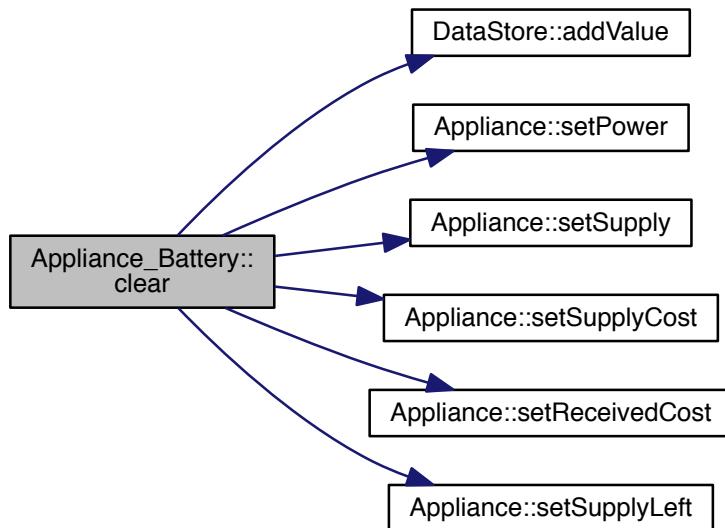


15.3.3.5 clear()

```
void Appliance_Battery::clear ( )
```

Definition at line 173 of file `Appliance_Battery.cpp`.

Here is the call graph for this function:

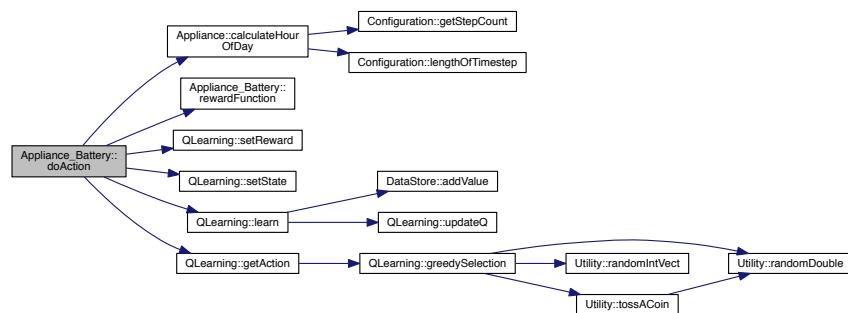


15.3.3.6 doAction()

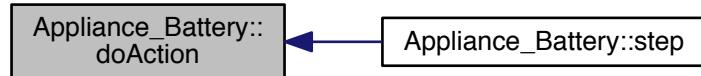
```
void Appliance_Battery::doAction ( ) [protected]
```

Definition at line 68 of file Appliance_Battery.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.3.3.7 energy_calc()

```
double Appliance_Battery::energy_calc ( ) const [protected]
```

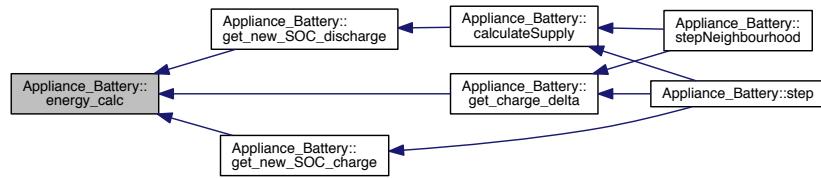
calculates the current energy available

Returns

current energy

Definition at line 253 of file Appliance_Battery.cpp.

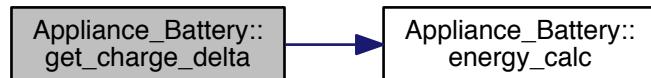
Here is the caller graph for this function:

**15.3.3.8 `get_charge_delta()`**

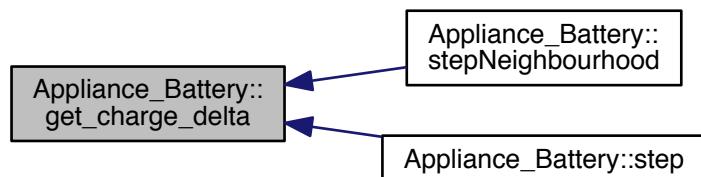
```
double Appliance_Battery::get_charge_delta () const [protected]
```

Definition at line 200 of file Appliance_Battery.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

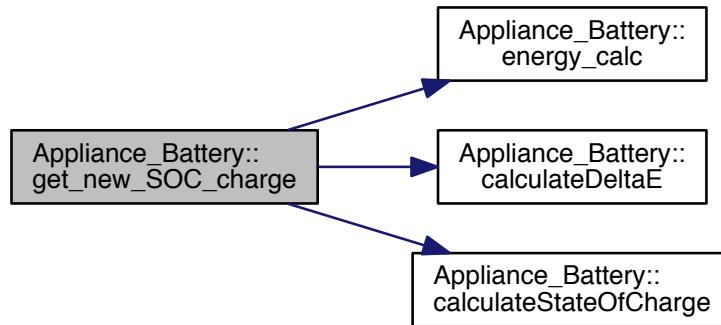


15.3.3.9 get_new_SOC_charge()

```
void Appliance_Battery::get_new_SOC_charge (
    double P_request ) [protected]
```

Definition at line 223 of file Appliance_Battery.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

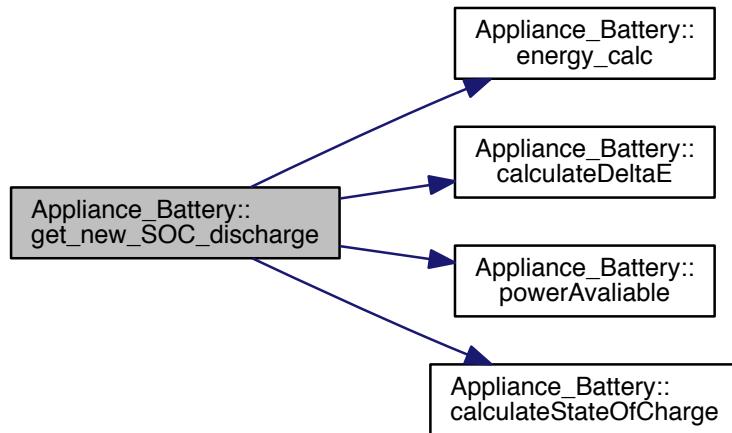


15.3.3.10 get_new_SOC_discharge()

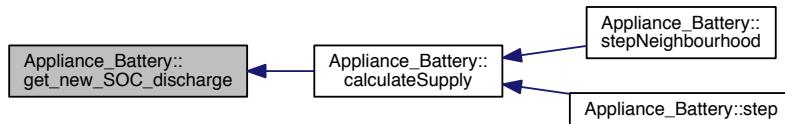
```
double Appliance_Battery::get_new_SOC_discharge (
    double P_request ) [protected]
```

Definition at line 233 of file Appliance_Battery.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.3.3.11 getStateOfCharge()

```
double Appliance_Battery::getStateOfCharge ( ) const
```

Definition at line 287 of file `Appliance_Battery.cpp`.

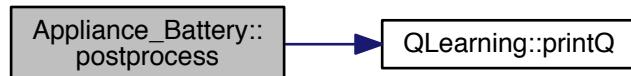
15.3.3.12 postprocess()

```
void Appliance_Battery::postprocess ( )
```

`Appliance` postprocess, calls Q-Learning save data.

Definition at line 188 of file `Appliance_Battery.cpp`.

Here is the call graph for this function:

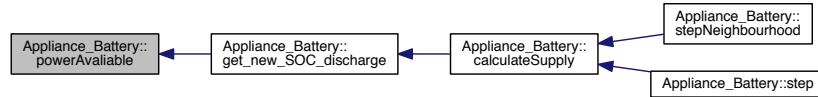


15.3.3.13 powerAvailable()

```
double Appliance_Battery::powerAvailable (
    const double energy ) const [protected]
```

Definition at line 228 of file `Appliance_Battery.cpp`.

Here is the caller graph for this function:



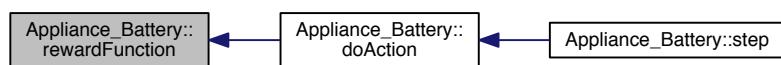
15.3.3.14 rewardFunction()

```
double Appliance_Battery::rewardFunction (
    double mostShortage,
    double binShortage ) const [virtual]
```

Reimplemented in [Appliance_Battery_GridCost_Reward](#).

Definition at line 13 of file `Appliance_Battery.cpp`.

Here is the caller graph for this function:



15.3.3.15 saveGlobalCalculate()

```
void Appliance_Battery::saveGlobalCalculate ( ) [protected], [virtual]
```

Reimplemented from [Appliance](#).

Definition at line 277 of file [Appliance_Battery.cpp](#).

15.3.3.16 saveNeighbourhoodCalculate()

```
void Appliance_Battery::saveNeighbourhoodCalculate ( ) [protected], [virtual]
```

Reimplemented from [Appliance](#).

Definition at line 257 of file [Appliance_Battery.cpp](#).

15.3.3.17 setBatteryNeighbourhoodCharge()

```
void Appliance_Battery::setBatteryNeighbourhoodCharge ( bool batteryNeighbourhoodCharge )
```

Definition at line 283 of file [Appliance_Battery.cpp](#).

Here is the caller graph for this function:

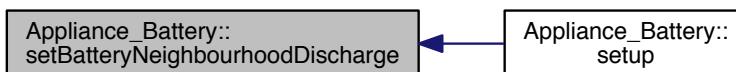


15.3.3.18 setBatteryNeighbourhoodDischarge()

```
void Appliance_Battery::setBatteryNeighbourhoodDischarge ( bool batteryNeighbourhoodDischarge )
```

Definition at line 279 of file [Appliance_Battery.cpp](#).

Here is the caller graph for this function:



15.3.3.19 setPowerShortage()

```
void Appliance_Battery::setPowerShortage (
    double power )
```

Definition at line 192 of file Appliance_Battery.cpp.

15.3.3.20 setup()

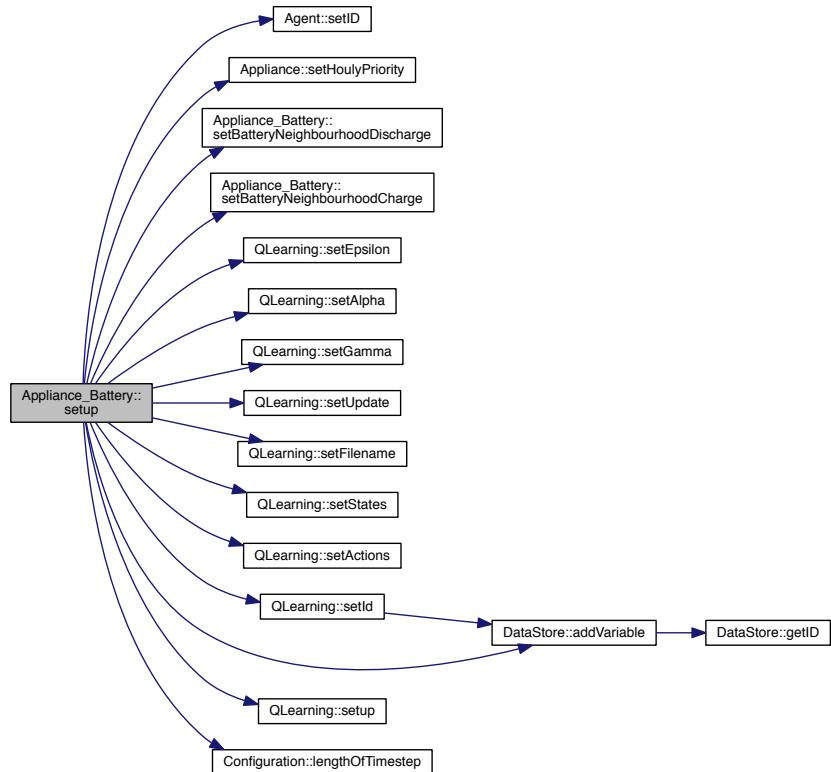
```
void Appliance_Battery::setup (
    ConfigStructAppliance a ) [virtual]
```

Set up the large appliance model, reading in the large appliance configuration file.

Implements [Appliance](#).

Definition at line 32 of file Appliance_Battery.cpp.

Here is the call graph for this function:



15.3.3.21 setupModel()

```
void Appliance_Battery::setupModel ( )
```

Set up the large appliance model, reading in the large appliance configuration file.

Sets the large appliance configuration file and gives the model the id of the appliance in the file

Definition at line 66 of file Appliance_Battery.cpp.

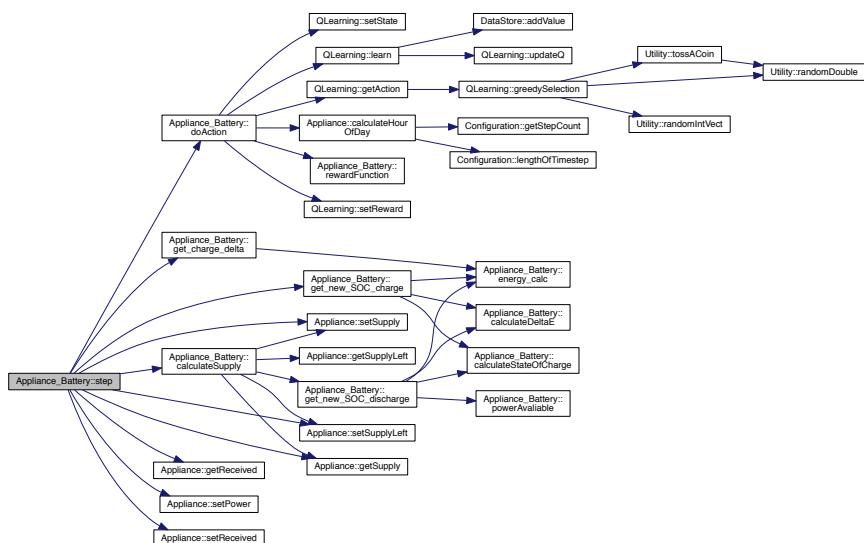
15.3.3.22 step()

```
void Appliance_Battery::step ( )
```

The timestep call of the battery appliance.

Definition at line 130 of file Appliance_Battery.cpp.

Here is the call graph for this function:



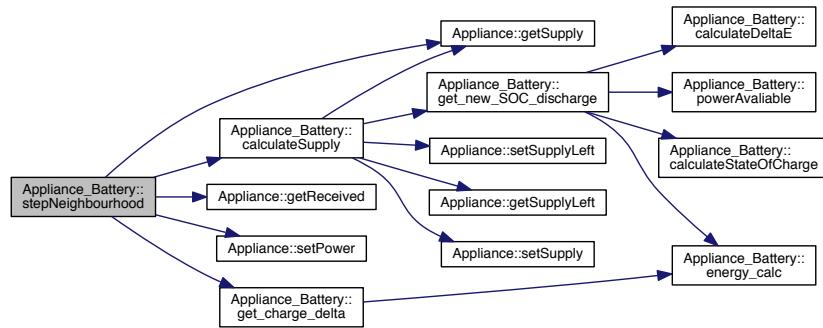
15.3.3.23 stepNeighbourhood()

```
void Appliance_Battery::stepNeighbourhood ( )
```

The timestep call of the battery appliance.

Definition at line 105 of file Appliance_Battery.cpp.

Here is the call graph for this function:



15.3.4 Member Data Documentation

15.3.4.1 action

```
bool Appliance_Battery::action [protected]
```

Definition at line 33 of file Appliance_Battery.hpp.

15.3.4.2 BatteryDeltaT

```
double Appliance_Battery::BatteryDeltaT [protected]
```

Definition at line 60 of file Appliance_Battery.hpp.

15.3.4.3 batteryNeighbourhoodCharge

```
bool Appliance_Battery::batteryNeighbourhoodCharge [protected]
```

Definition at line 63 of file Appliance_Battery.hpp.

15.3.4.4 **batteryNeighbourhoodDischarge**

```
bool Appliance_Battery::batteryNeighbourhoodDischarge [protected]
```

Definition at line 62 of file Appliance_Battery.hpp.

15.3.4.5 **capacity**

```
double Appliance_Battery::capacity = 2000 [protected]
```

Definition at line 57 of file Appliance_Battery.hpp.

15.3.4.6 **chargeRate**

```
double Appliance_Battery::chargeRate = 1000 [protected]
```

Definition at line 46 of file Appliance_Battery.hpp.

15.3.4.7 **cost**

```
double Appliance_Battery::cost [protected]
```

Definition at line 56 of file Appliance_Battery.hpp.

15.3.4.8 **datastoreIDstateOfCharge**

```
int Appliance_Battery::datastoreIDstateOfCharge [protected]
```

Definition at line 45 of file Appliance_Battery.hpp.

15.3.4.9 **dischargeRate**

```
double Appliance_Battery::dischargeRate = 1000 [protected]
```

Definition at line 47 of file Appliance_Battery.hpp.

15.3.4.10 efficiency

```
double Appliance_Battery::efficiency = 0.98 [protected]
```

Definition at line 58 of file Appliance_Battery.hpp.

15.3.4.11 mostShortage

```
double Appliance_Battery::mostShortage [protected]
```

Definition at line 55 of file Appliance_Battery.hpp.

15.3.4.12 powerShortage

```
double Appliance_Battery::powerShortage [protected]
```

Definition at line 48 of file Appliance_Battery.hpp.

15.3.4.13 previousHourOfDay

```
int Appliance_Battery::previousHourOfDay [protected]
```

Definition at line 50 of file Appliance_Battery.hpp.

15.3.4.14 qLearning

```
QLearning Appliance_Battery::qLearning [protected]
```

Definition at line 49 of file Appliance_Battery.hpp.

15.3.4.15 stateOfCharge

```
double Appliance_Battery::stateOfCharge [protected]
```

Definition at line 59 of file Appliance_Battery.hpp.

15.3.4.16 sumShort

```
double Appliance_Battery::sumShort [protected]
```

Definition at line 52 of file Appliance_Battery.hpp.

15.3.4.17 sumSupply

```
double Appliance_Battery::sumSupply [protected]
```

Definition at line 51 of file Appliance_Battery.hpp.

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

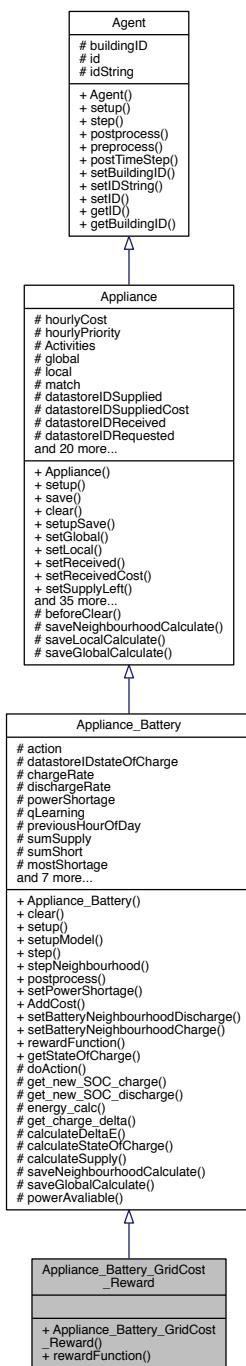
- Appliance_Battery.hpp
- Appliance_Battery.cpp

15.4 Appliance_Battery_GridCost_Reward Class Reference

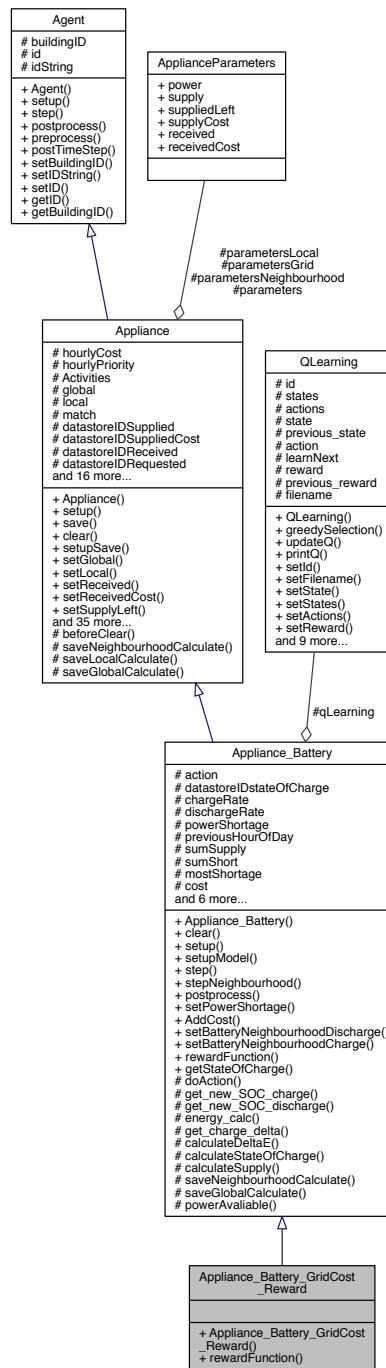
Battery Appliance class with reward calculated form the grid.

```
#include <Appliance_Battery_GridCost_Reward.hpp>
```

Inheritance diagram for Appliance_Battery_GridCost_Reward:



Collaboration diagram for Appliance_Battery_GridCost_Reward:



Public Member Functions

- **Appliance_Battery_GridCost_Reward ()**
- double **rewardFunction** (double **mostShortage**, double **binShortage**) const

Additional Inherited Members

15.4.1 Detailed Description

Battery Appliance class with reward calculated from the grid.

The Battery appliance agent

Definition at line 13 of file Appliance_Battery_GridCost_Reward.hpp.

15.4.2 Constructor & Destructor Documentation

15.4.2.1 Appliance_Battery_GridCost_Reward()

```
Appliance_Battery_GridCost_Reward::Appliance_Battery_GridCost_Reward ()
```

Definition at line 6 of file Appliance_Battery_GridCost_Reward.cpp.

15.4.3 Member Function Documentation

15.4.3.1 rewardFunction()

```
double Appliance_Battery_GridCost_Reward::rewardFunction (
    double mostShortage,
    double binShortage ) const [virtual]
```

Reimplemented from [Appliance_Battery](#).

Definition at line 8 of file Appliance_Battery_GridCost_Reward.cpp.

Here is the call graph for this function:



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

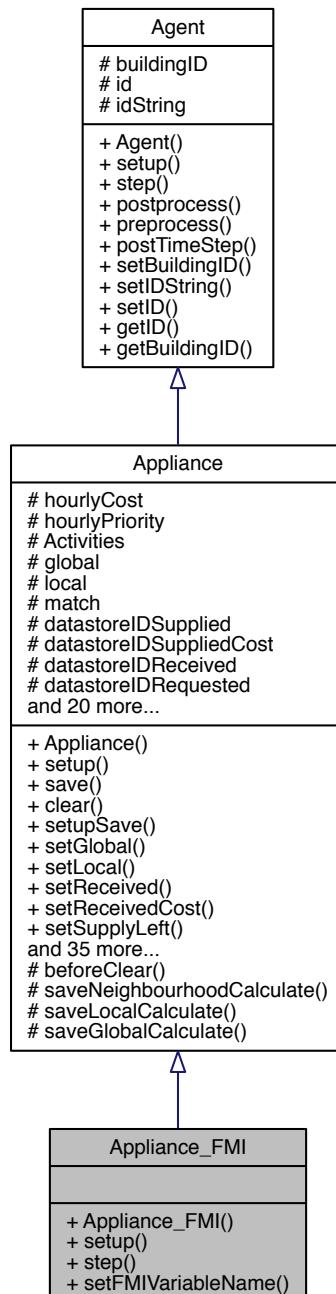
- Appliance_Battery_GridCost_Reward.hpp
- Appliance_Battery_GridCost_Reward.cpp

15.5 Appliance_FMI Class Reference

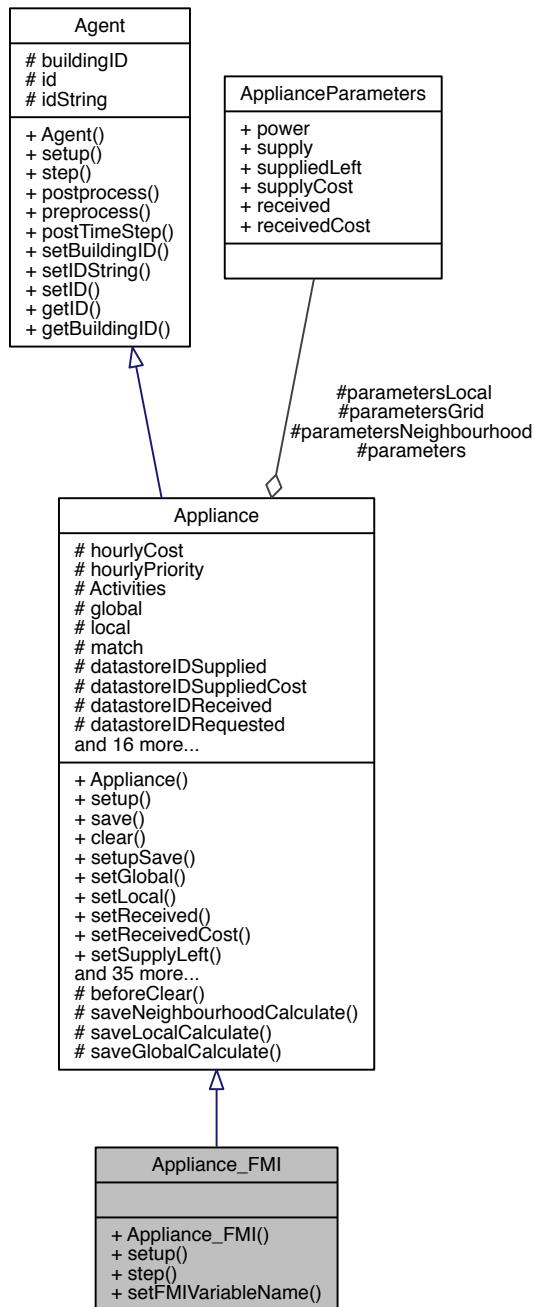
FMI appliances class.

```
#include <Appliance_FMI.hpp>
```

Inheritance diagram for Appliance_FMI:



Collaboration diagram for Appliance_FMI:



Public Member Functions

- [Appliance_FMI \(\)](#)
- void [setup \(ConfigStructAppliance a\)](#)
- void [step \(\)](#)
 - retrieves the value from the given variable name in the DataStore*
- void [setFМИVariableName \(const std::string &FМИVariableName\)](#)
 - Set the variable name of the variable we want to retrieve at run time.*

Additional Inherited Members

15.5.1 Detailed Description

FMI appliances class.

This will handles the power coming from the FMI interface for example electric heaters

Definition at line 15 of file Appliance_FMI.hpp.

15.5.2 Constructor & Destructor Documentation

15.5.2.1 Appliance_FMI()

```
Appliance_FMI::Appliance_FMI ( )
```

Definition at line 7 of file Appliance_FMI.cpp.

15.5.3 Member Function Documentation

15.5.3.1 setFMIVariableName()

```
void Appliance_FMI::setFMIVariableName (
    const std::string & FMIVariableName )
```

Set the variable name of the variable we want to retrieve at run time.

Definition at line 19 of file Appliance_FMI.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



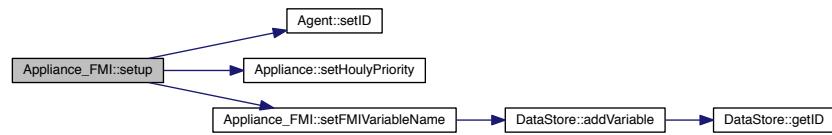
15.5.3.2 setup()

```
void Appliance_FMI::setup (
    ConfigStructAppliance a ) [virtual]
```

Implements [Appliance](#).

Definition at line 10 of file [Appliance_FMI.cpp](#).

Here is the call graph for this function:



15.5.3.3 step()

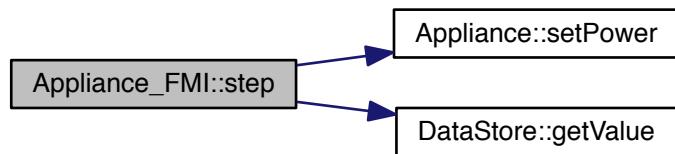
```
void Appliance_FMI::step ( )
```

retrieves the value from the given variable name in the DataStore

At each simulation timestep with FMI all input variables are written to the data store, this simply returns the variable

Definition at line 28 of file [Appliance_FMI.cpp](#).

Here is the call graph for this function:



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

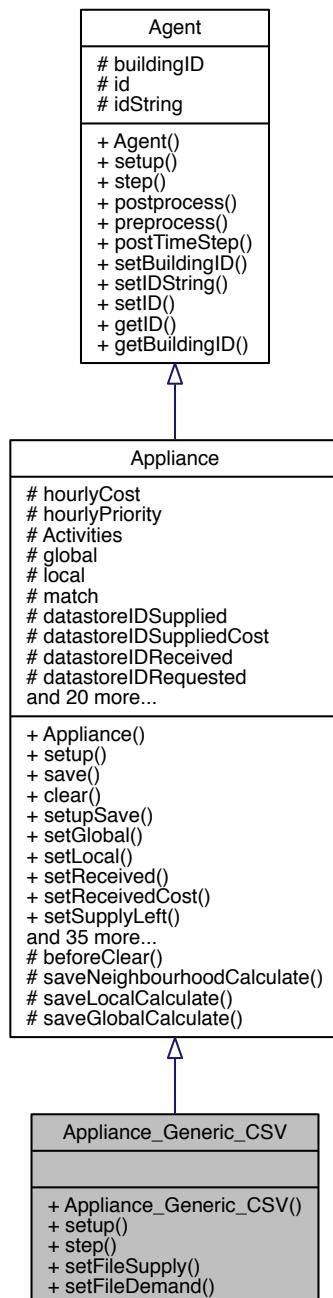
- [Appliance_FMI.hpp](#)
- [Appliance_FMI.cpp](#)

15.6 Appliance_Generic_CSV Class Reference

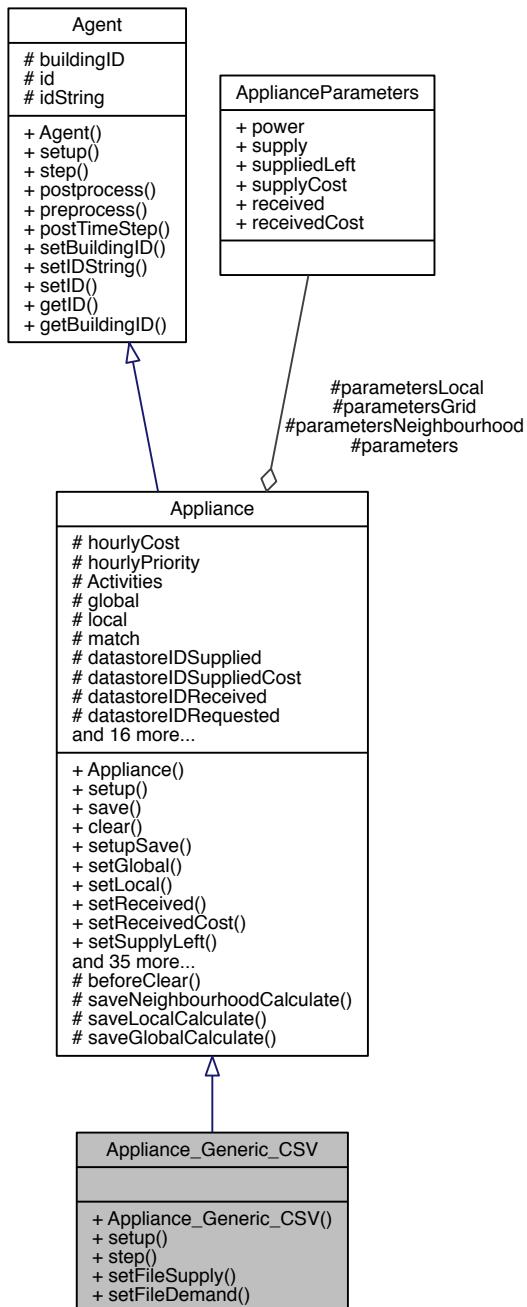
appliance read in from csv class

```
#include <Appliance_Generic_CSV.hpp>
```

Inheritance diagram for Appliance_Generic_CSV:



Collaboration diagram for Appliance_Generic_CSV:



Public Member Functions

- **Appliance_Generic_CSV ()**
- void **setup (ConfigStructAppliance a)**
read in the csv file containing the power values
- void **step ()**
- void **setFileSupply (const std::string &filename)**

- Sets the csv file to be read in.*
- void [setFileDemand](#) (const std::string &filename)
Sets the csv file to be read in.

Additional Inherited Members

15.6.1 Detailed Description

appliance read in from csv class

The CSV agent, handles the csv model

Definition at line 16 of file Appliance_Generic_CSV.hpp.

15.6.2 Constructor & Destructor Documentation

15.6.2.1 Appliance_Generic_CSV()

Appliance_Generic_CSV::Appliance_Generic_CSV ()

Definition at line 9 of file Appliance_Generic_CSV.cpp.

15.6.3 Member Function Documentation

15.6.3.1 setFileDemand()

```
void Appliance_Generic_CSV::setFileDemand ( const std::string & filename )
```

Sets the csv file to be read in.

Parameters

<i>filename</i>	the file name
-----------------	---------------

Definition at line 85 of file Appliance_Generic_CSV.cpp.

Here is the caller graph for this function:



15.6.3.2 setFileSupply()

```
void Appliance_Generic_CSV::setFileSupply ( const std::string & filename )
```

Sets the csv file to be read in.

Parameters

<i>filename</i>	the file name
-----------------	---------------

Definition at line 77 of file Appliance_Generic_CSV.cpp.

Here is the caller graph for this function:



15.6.3.3 setup()

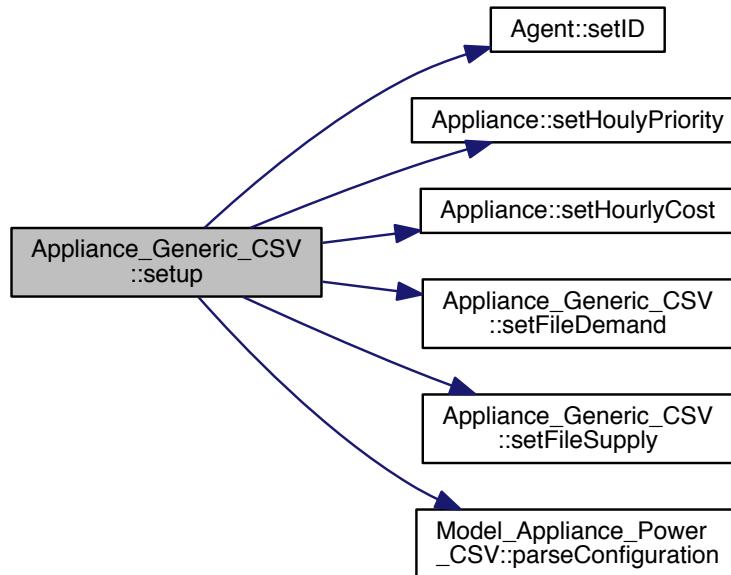
```
void Appliance_Generic_CSV::setup ( ConfigStructAppliance a ) [virtual]
```

read in the csv file containing the power values

Implements [Appliance](#).

Definition at line 14 of file Appliance_Generic_CSV.cpp.

Here is the call graph for this function:

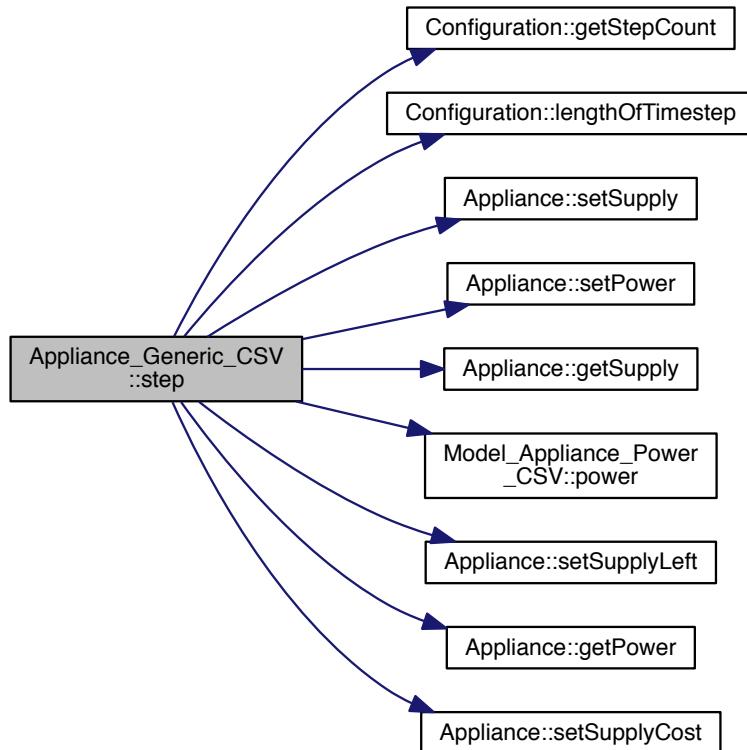


15.6.3.4 step()

```
void Appliance_Generic_CSV::step ( )
```

Definition at line 36 of file Appliance_Generic_CSV.cpp.

Here is the call graph for this function:



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

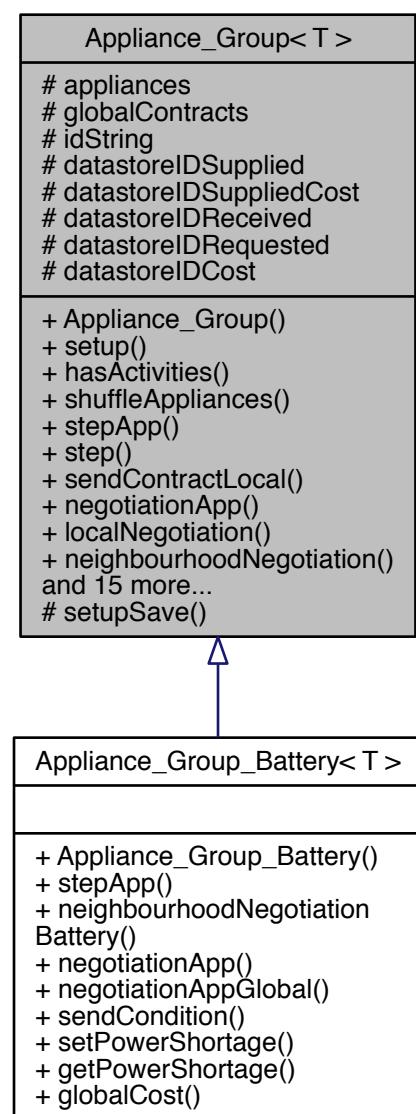
- `Appliance_Generic_CSV.hpp`
- `Appliance_Generic_CSV.cpp`

15.7 Appliance_Group< T > Class Template Reference

Groups the types appliances for easy handling.

```
#include <Appliance_Group.hpp>
```

Inheritance diagram for Appliance_Group< T >:



Collaboration diagram for Appliance_Group< T >:

Appliance_Group< T >
<pre># appliances # globalContracts # idString # datastoreIDSupplied # datastoreIDSuppliedCost # datastoreIDReceived # datastoreIDRequested # datastoreIDCost + Appliance_Group() + setup() + hasActivities() + shuffleAppliances() + stepApp() + step() + sendContractLocal() + negotiationApp() + localNegotiation() + neighbourhoodNegotiation() and 15 more... # setupSave()</pre>

Public Member Functions

- [Appliance_Group \(\)](#)
- void [setup](#) (const std::vector< [ConfigStructAppliance](#) > &app, const int &buildingID, const std::string &buildingString)
- void [hasActivities](#) (const std::vector< int > &Activities)
- void [shuffleAppliances](#) ()
- virtual void [stepApp](#) (T &a, [Contract_Negotiation](#) *app_negotiation)
- void [step](#) ([Contract_Negotiation](#) *app_negotiation)
- bool [sendContractLocal](#) (const T &a, [Contract_Negotiation](#) *app_negotiation)
- virtual void [negotiationApp](#) (const [Contract_Negotiation](#) &app_negotiation, T &app, const bool negotiate)
- void [localNegotiation](#) (const [Contract_Negotiation](#) &app_negotiation)
- void [neighbourhoodNegotiation](#) (const [Contract_Negotiation](#) &building_negotiation)
- virtual void [negotiationAppGlobal](#) (const [Contract_Negotiation](#) &app_negotiation, T &app, const bool negotiate)
- void [globalNegotiation](#) (const [Contract_Negotiation](#) &building_negotiation)
- void [clear](#) ()
- virtual bool [sendCondition](#) (const [Contract](#) &c)
- bool [sendContractGlobal](#) (const [Contract](#) &c)
- void [postprocess](#) ()
- void [addGlobalContactsTo](#) ([Contract_Negotiation](#) *building_negotiation)
- double [getSupply](#) () const
- double [getSupplyLeft](#) () const
- double [getSupplyCost](#) () const
- double [getPower](#) () const

- double `getReceived () const`
- double `getReceivedCost () const`
- T `getApplianceAt (int BuildingID, int id)`
- void `setIDString (const std::string id)`

Protected Member Functions

- void `setupSave ()`

Protected Attributes

- std::vector< T > `appliances`
- std::vector< `Contract` > `globalContracts`
- std::string `idString`
- int `datastoreIDSupplied`
- int `datastoreIDSuppliedCost`
- int `datastoreIDReceived`
- int `datastoreIDRequested`
- int `datastoreIDCost`

15.7.1 Detailed Description

```
template<class T>
class Appliance_Group< T >
```

Groups the types appliances for easy handling.

Definition at line 19 of file Appliance_Group.hpp.

15.7.2 Constructor & Destructor Documentation

15.7.2.1 Appliance_Group()

```
template<class T>
Appliance_Group< T >::Appliance_Group ( ) [inline]
```

Definition at line 22 of file Appliance_Group.hpp.

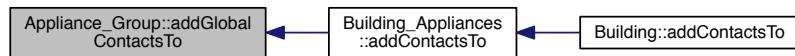
15.7.3 Member Function Documentation

15.7.3.1 addGlobalContactsTo()

```
template<class T>
void Appliance_Group< T >::addGlobalContactsTo (
    Contract_Negotiation * building_negotiation ) [inline]
```

Definition at line 156 of file Appliance_Group.hpp.

Here is the caller graph for this function:



15.7.3.2 clear()

```
template<class T>
void Appliance_Group< T >::clear () [inline]
```

Definition at line 132 of file Appliance_Group.hpp.

Here is the caller graph for this function:



15.7.3.3 getApplianceAt()

```
template<class T>
T Appliance_Group< T >::getApplianceAt (
    int BuildingID,
    int id ) [inline]
```

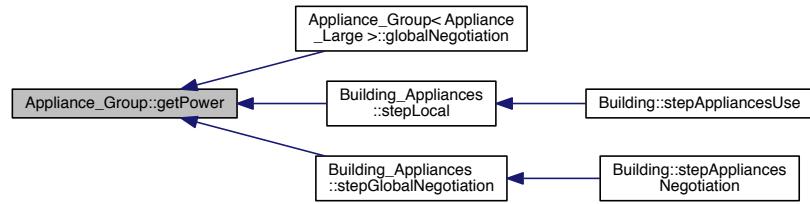
Definition at line 212 of file Appliance_Group.hpp.

15.7.3.4 `getPower()`

```
template<class T>
double Appliance_Group< T >::getPower ( ) const [inline]
```

Definition at line 188 of file Appliance_Group.hpp.

Here is the caller graph for this function:



15.7.3.5 `getReceived()`

```
template<class T>
double Appliance_Group< T >::getReceived ( ) const [inline]
```

Definition at line 196 of file Appliance_Group.hpp.

Here is the caller graph for this function:

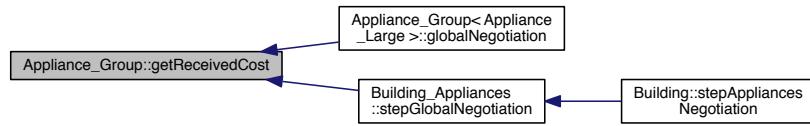


15.7.3.6 getReceivedCost()

```
template<class T>
double Appliance_Group< T >::getReceivedCost ( ) const [inline]
```

Definition at line 204 of file Appliance_Group.hpp.

Here is the caller graph for this function:

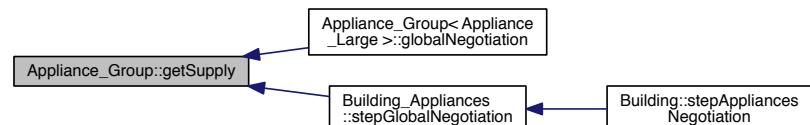


15.7.3.7 getSupply()

```
template<class T>
double Appliance_Group< T >::getSupply ( ) const [inline]
```

Definition at line 164 of file Appliance_Group.hpp.

Here is the caller graph for this function:

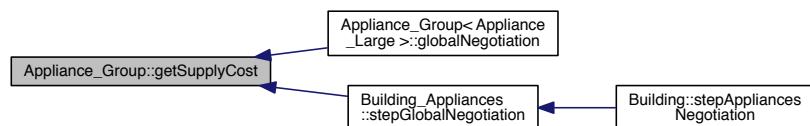


15.7.3.8 getSupplyCost()

```
template<class T>
double Appliance_Group< T >::getSupplyCost ( ) const [inline]
```

Definition at line 180 of file Appliance_Group.hpp.

Here is the caller graph for this function:



15.7.3.9 getSupplyLeft()

```
template<class T>
double Appliance_Group< T >::getSupplyLeft ( ) const [inline]
```

Definition at line 172 of file Appliance_Group.hpp.

Here is the caller graph for this function:



15.7.3.10 globalNegotiation()

```
template<class T>
void Appliance_Group< T >::globalNegotiation (
    const Contract_Negotiation & building_negotiation ) [inline]
```

Definition at line 117 of file Appliance_Group.hpp.

Here is the caller graph for this function:



15.7.3.11 hasActivities()

```
template<class T>
void Appliance_Group< T >::hasActivities (
    const std::vector< int > & Activities ) [inline]
```

Definition at line 37 of file Appliance_Group.hpp.

Here is the caller graph for this function:



15.7.3.12 localNegotiation()

```
template<class T>
void Appliance_Group< T >::localNegotiation (
    const Contract_Negotiation & app_negotiation ) [inline]
```

Definition at line 98 of file Appliance_Group.hpp.

Here is the caller graph for this function:



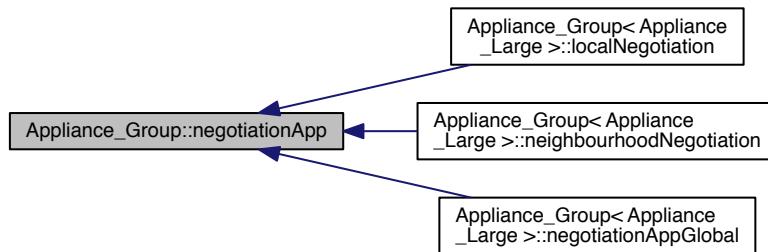
15.7.3.13 negotiationApp()

```
template<class T>
virtual void Appliance_Group< T >::negotiationApp (
    const Contract_Negotiation & app_negotiation,
    T & app,
    const bool negotiate ) [inline], [virtual]
```

Reimplemented in [Appliance_Group_Battery< T >](#), [Appliance_Group_Battery< Appliance_Battery_GridCost_<- Reward >](#), and [Appliance_Group_Battery< Appliance_Battery >](#).

Definition at line 79 of file Appliance_Group.hpp.

Here is the caller graph for this function:



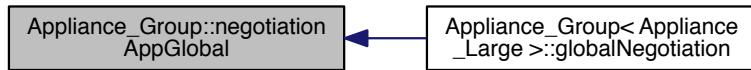
15.7.3.14 negotiationAppGlobal()

```
template<class T>
virtual void Appliance_Group< T >::negotiationAppGlobal (
    const Contract_Negotiation & app_negotiation,
    T & app,
    const bool negotiate ) [inline], [virtual]
```

Reimplemented in [Appliance_Group_Battery< T >](#), [Appliance_Group_Battery< Appliance_Battery_GridCost_↪ Reward >](#), and [Appliance_Group_Battery< Appliance_Battery >](#).

Definition at line 112 of file [Appliance_Group.hpp](#).

Here is the caller graph for this function:



15.7.3.15 neighbourhoodNegotiation()

```
template<class T>
void Appliance_Group< T >::neighbourhoodNegotiation (
    const Contract_Negotiation & building_negotiation ) [inline]
```

Definition at line 105 of file [Appliance_Group.hpp](#).

Here is the caller graph for this function:

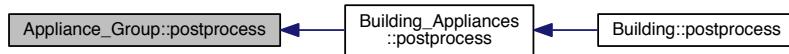


15.7.3.16 postprocess()

```
template<class T>
void Appliance_Group< T >::postprocess ( ) [inline]
```

Definition at line 150 of file Appliance_Group.hpp.

Here is the caller graph for this function:



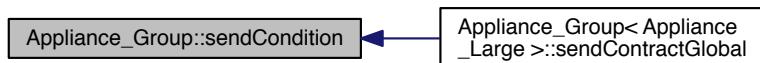
15.7.3.17 sendCondition()

```
template<class T>
virtual bool Appliance_Group< T >::sendCondition (
    const Contract & c ) [inline], [virtual]
```

Reimplemented in [Appliance_Group_Battery< T >](#), [Appliance_Group_Battery< Appliance_Battery_GridCost_<- Reward >](#), and [Appliance_Group_Battery< Appliance_Battery >](#).

Definition at line 138 of file Appliance_Group.hpp.

Here is the caller graph for this function:

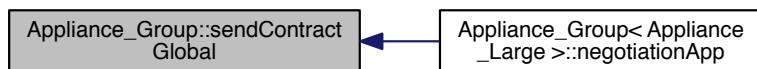


15.7.3.18 sendContractGlobal()

```
template<class T>
bool Appliance_Group< T >::sendContractGlobal (
    const Contract & c ) [inline]
```

Definition at line 142 of file Appliance_Group.hpp.

Here is the caller graph for this function:

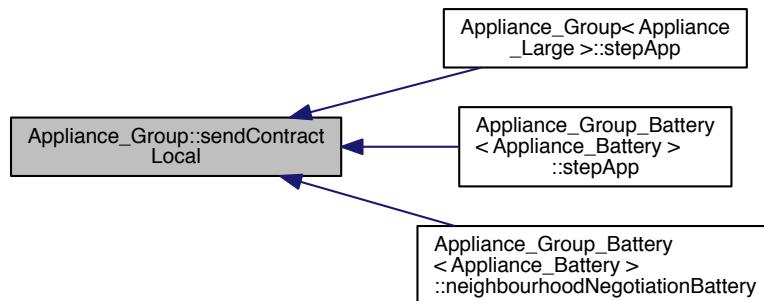


15.7.3.19 sendContractLocal()

```
template<class T>
bool Appliance_Group< T >::sendContractLocal (
    const T & a,
    Contract_Negotiation * app_negotiation ) [inline]
```

Definition at line 61 of file Appliance_Group.hpp.

Here is the caller graph for this function:



15.7.3.20 setIdString()

```
template<class T>
void Appliance_Group< T >::setIdString (
    const std::string id) [inline]
```

Definition at line 222 of file Appliance_Group.hpp.

Here is the caller graph for this function:



15.7.3.21 setup()

```
template<class T>
void Appliance_Group< T >::setup (
    const std::vector< ConfigStructAppliance > & app,
    const int & buildingID,
    const std::string & buildingString) [inline]
```

Definition at line 24 of file Appliance_Group.hpp.

Here is the caller graph for this function:

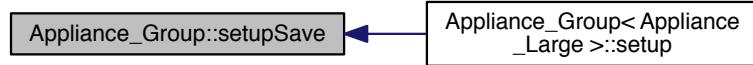


15.7.3.22 setupSave()

```
template<class T>
void Appliance_Group< T >::setupSave( ) [inline], [protected]
```

Definition at line 231 of file Appliance_Group.hpp.

Here is the caller graph for this function:

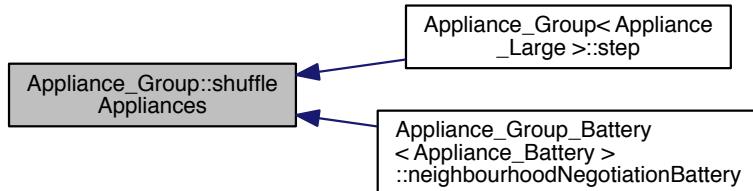


15.7.3.23 shuffleAppliances()

```
template<class T>
void Appliance_Group< T >::shuffleAppliances ( ) [inline]
```

Definition at line 43 of file Appliance_Group.hpp.

Here is the caller graph for this function:



15.7.3.24 step()

```
template<class T>
void Appliance_Group< T >::step (
    Contract_Negotiation * app_negotiation ) [inline]
```

Definition at line 54 of file Appliance_Group.hpp.

Here is the caller graph for this function:



15.7.3.25 stepApp()

```
template<class T>
virtual void Appliance_Group< T >::stepApp (
    T & a,
    Contract_Negotiation * app_negotiation ) [inline], [virtual]
```

Reimplemented in [Appliance_Group_Battery< T >](#), [Appliance_Group_Battery< Appliance_Battery_GridCost_↪Reward >](#), and [Appliance_Group_Battery< Appliance_Battery >](#).

Definition at line 47 of file Appliance_Group.hpp.

Here is the caller graph for this function:



15.7.4 Member Data Documentation

15.7.4.1 appliances

```
template<class T>
std::vector<T> Appliance_Group< T >::appliances [protected]
```

Definition at line 227 of file Appliance_Group.hpp.

15.7.4.2 datastoreIDCost

```
template<class T>
int Appliance_Group< T >::datastoreIDCost [protected]
```

Definition at line 245 of file Appliance_Group.hpp.

15.7.4.3 datastoreIDReceived

```
template<class T>
int Appliance_Group< T >::datastoreIDReceived [protected]
```

Definition at line 243 of file Appliance_Group.hpp.

15.7.4.4 `datastoreIDRequested`

```
template<class T>
int Appliance\_Group< T >::datastoreIDRequested [protected]
```

Definition at line 244 of file `Appliance_Group.hpp`.

15.7.4.5 `datastoreIDSupplied`

```
template<class T>
int Appliance\_Group< T >::datastoreIDSupplied [protected]
```

Definition at line 241 of file `Appliance_Group.hpp`.

15.7.4.6 `datastoreIDSuppliedCost`

```
template<class T>
int Appliance\_Group< T >::datastoreIDSuppliedCost [protected]
```

Definition at line 242 of file `Appliance_Group.hpp`.

15.7.4.7 `globalContracts`

```
template<class T>
std::vector<Contract> Appliance\_Group< T >::globalContracts [protected]
```

Definition at line 228 of file `Appliance_Group.hpp`.

15.7.4.8 `idString`

```
template<class T>
std::string Appliance\_Group< T >::idString [protected]
```

Definition at line 229 of file `Appliance_Group.hpp`.

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

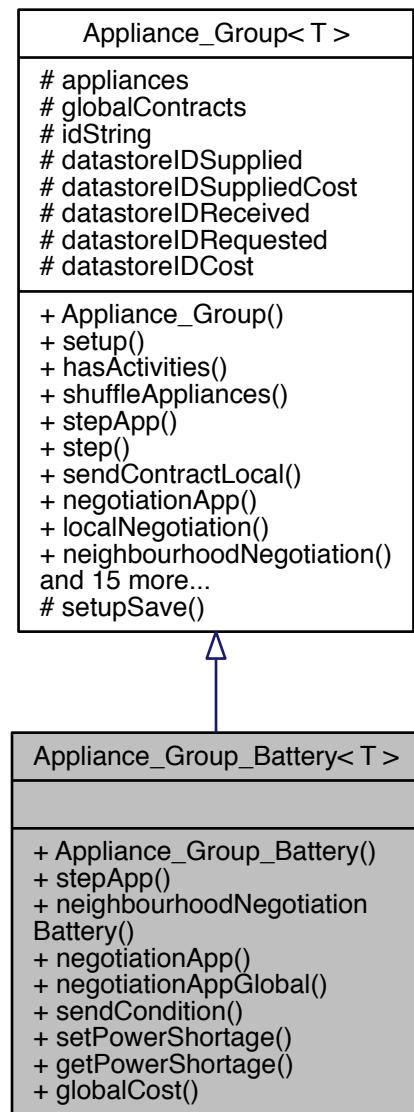
- `Appliance_Group.hpp`

15.8 Appliance_Group_Battery< T > Class Template Reference

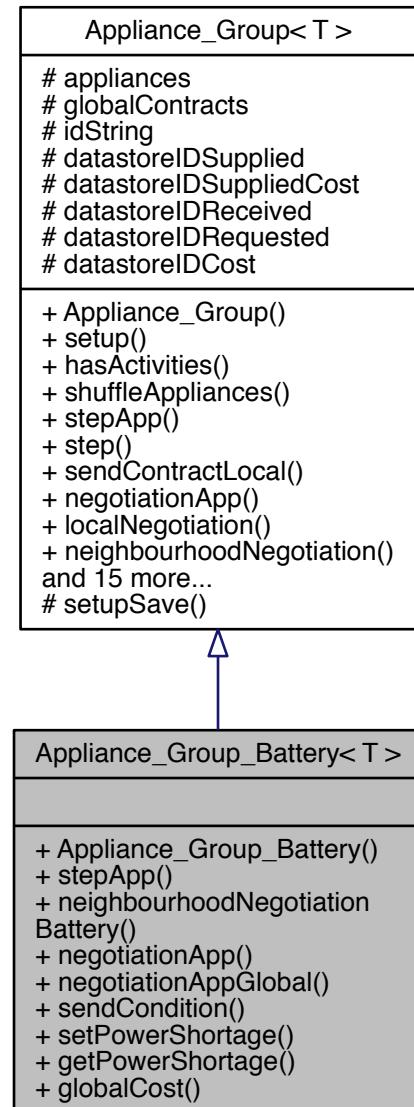
Super class of each appliance type.

```
#include <Appliance_Group_Battery.hpp>
```

Inheritance diagram for Appliance_Group_Battery< T >:



Collaboration diagram for Appliance_Group_Battery< T >:



Public Member Functions

- `Appliance_Group_Battery ()`
- `void stepApp (T &a, Contract_Negotiation *app_negotiation)`
- `void neighbourhoodNegotiationBattery (Contract_Negotiation *building_negotiation)`
- `void negotiationApp (const Contract_Negotiation &app_negotiation, T &app, const bool negotiate)`
- `void negotiationAppGlobal (const Contract_Negotiation &app_negotiation, T &app, const bool negotiate)`
- `virtual bool sendCondition (const Contract &c)`
- `void setPowerShortage (double powerShortage)`
- `double getPowerShortage () const`
- `void globalCost (double globalCost)`

Additional Inherited Members

15.8.1 Detailed Description

```
template<class T>
class Appliance_Group_Battery< T >
```

Super class of each appliance type.

An agent appliance that is used to house helper methods used in the sub classes

Definition at line 14 of file Appliance_Group_Battery.hpp.

15.8.2 Constructor & Destructor Documentation

15.8.2.1 Appliance_Group_Battery()

```
template<class T>
Appliance_Group_Battery< T >::Appliance_Group_Battery ( ) [inline]
```

Definition at line 16 of file Appliance_Group_Battery.hpp.

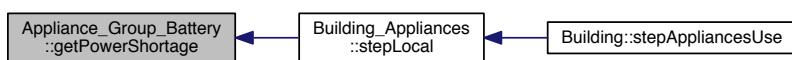
15.8.3 Member Function Documentation

15.8.3.1 getPowerShortage()

```
template<class T>
double Appliance_Group_Battery< T >::getPowerShortage ( ) const [inline]
```

Definition at line 71 of file Appliance_Group_Battery.hpp.

Here is the caller graph for this function:



15.8.3.2 globalCost()

```
template<class T>
void Appliance_Group_Battery< T >::globalCost (
    double globalCost ) [inline]
```

Definition at line 75 of file Appliance_Group_Battery.hpp.

Here is the caller graph for this function:



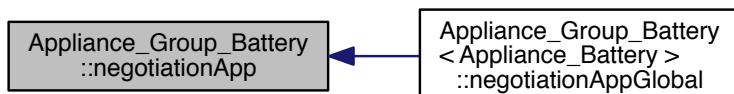
15.8.3.3 negotiationApp()

```
template<class T>
void Appliance_Group_Battery< T >::negotiationApp (
    const Contract_Negotiation & app_negotiation,
    T & app,
    const bool negotiate ) [inline], [virtual]
```

Reimplemented from [Appliance_Group< T >](#).

Definition at line 38 of file Appliance_Group_Battery.hpp.

Here is the caller graph for this function:



15.8.3.4 negotiationAppGlobal()

```
template<class T>
void Appliance_Group_Battery< T >::negotiationAppGlobal (
    const Contract_Negotiation & app_negotiation,
    T & app,
    const bool negotiate ) [inline], [virtual]
```

Reimplemented from [Appliance_Group< T >](#).

Definition at line 58 of file Appliance_Group_Battery.hpp.

15.8.3.5 neighbourhoodNegotiationBattery()

```
template<class T>
void Appliance_Group_Battery< T >::neighbourhoodNegotiationBattery (
    Contract_Negotiation * building_negotiation ) [inline]
```

Definition at line 27 of file Appliance_Group_Battery.hpp.

Here is the caller graph for this function:



15.8.3.6 sendCondition()

```
template<class T>
virtual bool Appliance_Group_Battery< T >::sendCondition (
    const Contract & c ) [inline], [virtual]
```

Reimplemented from [Appliance_Group< T >](#).

Definition at line 63 of file Appliance_Group_Battery.hpp.

15.8.3.7 setPowerShortage()

```
template<class T>
void Appliance_Group_Battery< T >::setPowerShortage (
    double powerShortage ) [inline]
```

Definition at line 67 of file Appliance_Group_Battery.hpp.

Here is the caller graph for this function:



15.8.3.8 stepApp()

```
template<class T>
void Appliance\_Group\_Battery< T >::stepApp (
    T & a,
    Contract\_Negotiation * app_negotiation ) [inline], [virtual]
```

Reimplemented from [Appliance_Group](#)< T >.

Definition at line 18 of file [Appliance_Group_Battery.hpp](#).

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

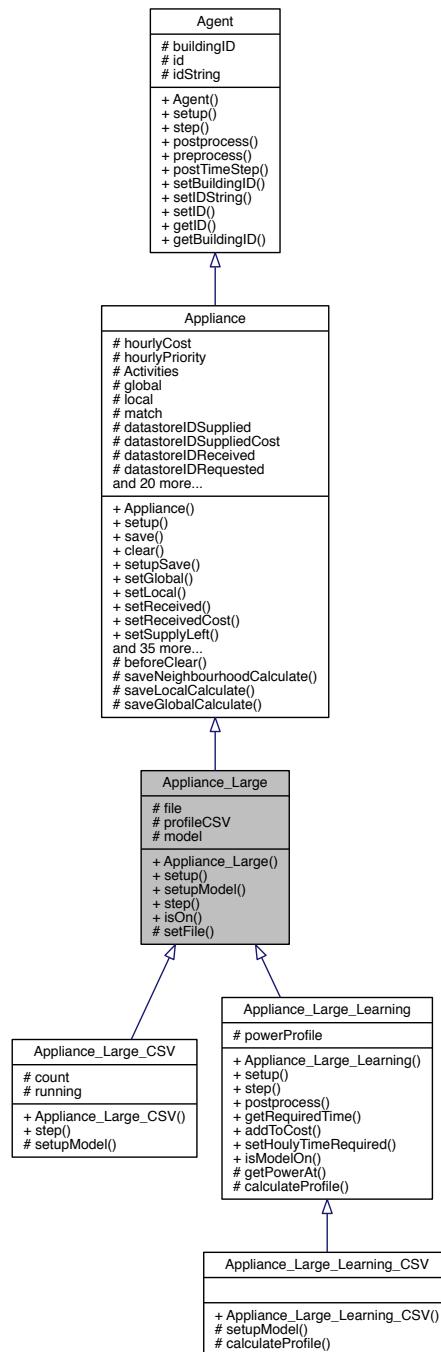
- [Appliance_Group_Battery.hpp](#)

15.9 Appliance_Large Class Reference

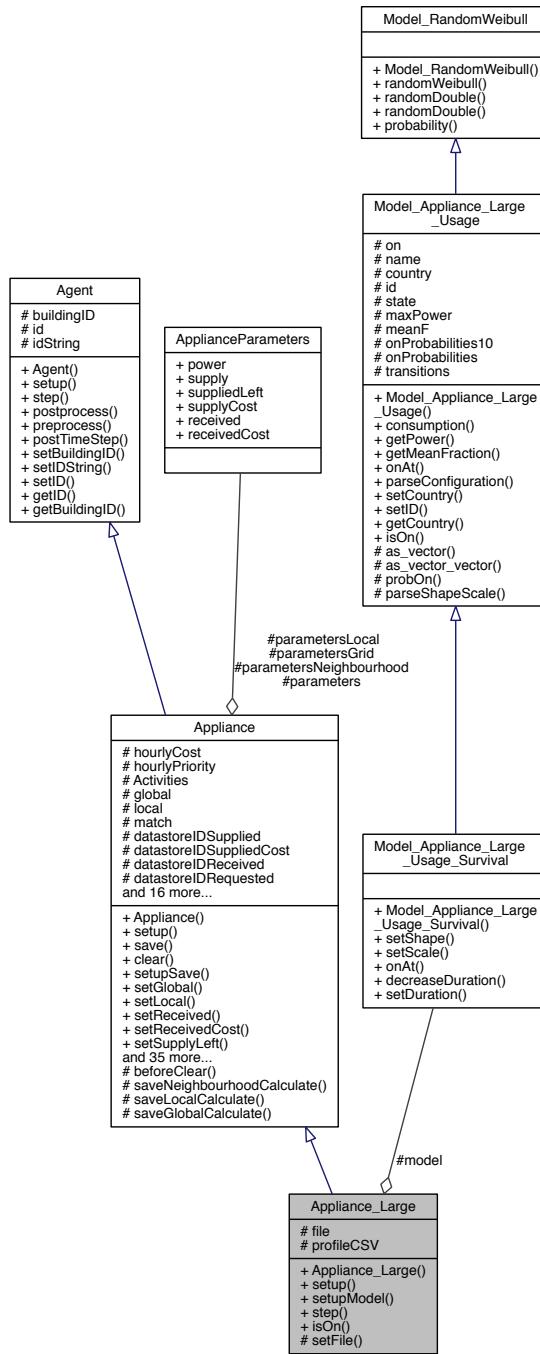
Large appliances class.

```
#include <Appliance_Large.hpp>
```

Inheritance diagram for Appliance_Large:



Collaboration diagram for Appliance_Large:



Public Member Functions

- `Appliance_Large ()`
- void `setup (ConfigStructAppliance a)`
Set up the large appliance model, reading in the large appliance configuration file.
- virtual void `setupModel ()`
Set up the large appliance model, reading in the large appliance configuration file.

- void [step \(\)](#)
The timestep call of the appliance.
- bool [isOn \(\) const](#)
Check the model for if the appliance is turned on.

Protected Member Functions

- void [setFile \(std::string file\)](#)

Protected Attributes

- std::string [file](#)
- std::vector< double > [profileCSV](#)
- [Model_Appliance_Large_Usage_Survival model](#)

15.9.1 Detailed Description

Large appliances class.

The large appliance agent, handles the model survival/ markov hybrid model

Definition at line 15 of file Appliance_Large.hpp.

15.9.2 Constructor & Destructor Documentation

15.9.2.1 Appliance_Large()

```
Appliance_Large::Appliance_Large ( )
```

Definition at line 11 of file Appliance_Large.cpp.

15.9.3 Member Function Documentation

15.9.3.1 isOn()

```
bool Appliance_Large::isOn ( ) const
```

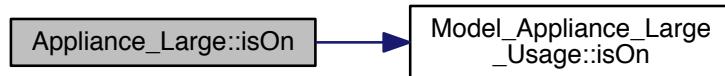
Check the model for if the appliance is turned on.

Returns

if the appliance is turned on or not.

Definition at line 58 of file Appliance_Large.cpp.

Here is the call graph for this function:

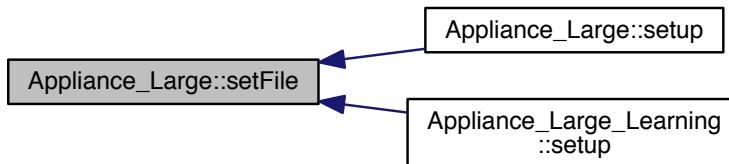


15.9.3.2 setFile()

```
void Appliance_Large::setFile (
    std::string file ) [protected]
```

Definition at line 63 of file Appliance_Large.cpp.

Here is the caller graph for this function:



15.9.3.3 setup()

```
void Appliance_Large::setup (
    ConfigStructAppliance a ) [virtual]
```

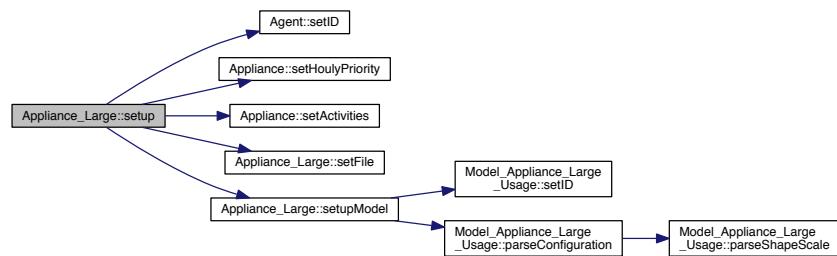
Set up the large appliance model, reading in the large appliance configuration file.

Implements [Appliance](#).

Reimplemented in [Appliance_Large_Learning](#).

Definition at line 17 of file [Appliance_Large.cpp](#).

Here is the call graph for this function:



15.9.3.4 setupModel()

```
void Appliance_Large::setupModel ( ) [virtual]
```

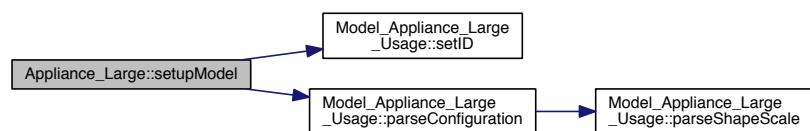
Set up the large appliance model, reading in the large appliance configuration file.

Sets the large appliance configuration file and gives the model the id of the appliance in the file

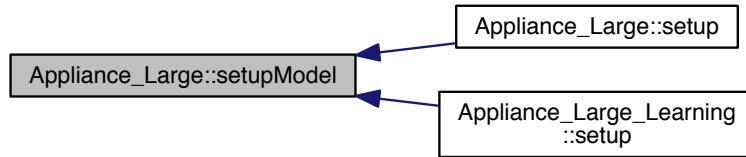
Reimplemented in [Appliance_Large_Learning_CSV](#), and [Appliance_Large_CSV](#).

Definition at line 31 of file [Appliance_Large.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:



15.9.3.5 step()

```
void Appliance_Large::step ( )
```

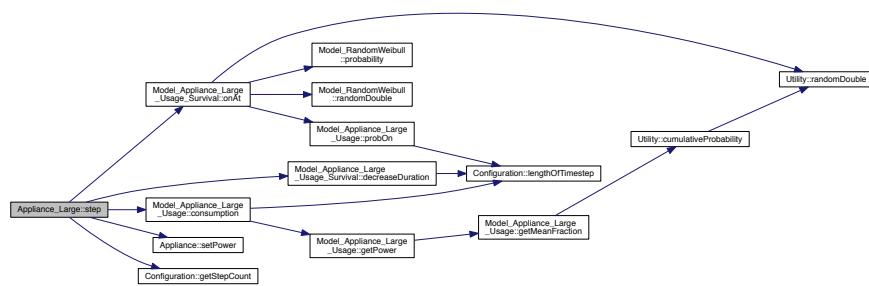
The timestep call of the appliance.

If the appliance is on increase the survival time and retrieve the power.

If the appliance is off calculate if the appliance turns on and how long for and retrieve power.
Otherwise set power to 0.

Definition at line 44 of file `Appliance_Large.cpp`.

Here is the call graph for this function:



15.9.4 Member Data Documentation

15.9.4.1 file

```
std::string Appliance_Large::file [protected]
```

Definition at line 26 of file `Appliance_Large.hpp`.

15.9.4.2 model

```
Model_Appliance_Large_Usage_Survival Appliance_Large::model [protected]
```

Definition at line 28 of file Appliance_Large.hpp.

15.9.4.3 profileCSV

```
std::vector<double> Appliance_Large::profileCSV [protected]
```

Definition at line 27 of file Appliance_Large.hpp.

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

- Appliance_Large.hpp
- Appliance_Large.cpp

15.10 Appliance_Large_CSV Class Reference

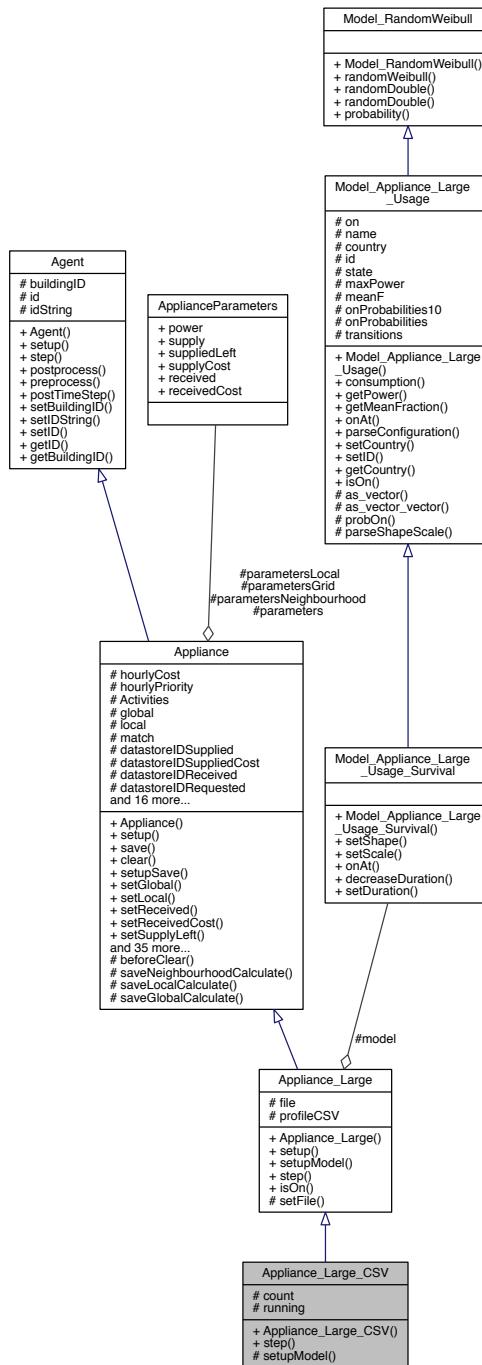
Large appliances class which uses CSV profiles.

```
#include <Appliance_Large_CSV.hpp>
```

Inheritance diagram for Appliance_Large_CSVD:



Collaboration diagram for Appliance_Large_CSV:



Public Member Functions

- [Appliance_Large_CSV \(\)](#)
- [void step \(\)](#)

Check large appliance model for a turn on, then generate the profile.

Protected Member Functions

- void [setupModel \(\)](#)
Set up the large appliance model, reading in the large appliance configuration file.

Protected Attributes

- unsigned int [count](#)
- bool [running](#)

15.10.1 Detailed Description

Large appliances class which uses CSV profiles.

The large appliance agent, handles the model survival/ markov hybrid model however the profile is now take from a csv file

Definition at line 16 of file Appliance_Large_CSV.hpp.

15.10.2 Constructor & Destructor Documentation

15.10.2.1 Appliance_Large_CSV()

`Appliance_Large_CSV::Appliance_Large_CSV ()`

Definition at line 11 of file Appliance_Large_CSV.cpp.

15.10.3 Member Function Documentation

15.10.3.1 setupModel()

`void Appliance_Large_CSV::setupModel () [protected], [virtual]`

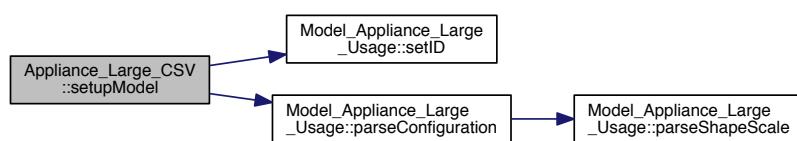
Set up the large appliance model, reading in the large appliance configuration file.

Sets the large appliance configuration file and gives the model the id of the appliance in the file

Reimplemented from [Appliance_Large](#).

Definition at line 43 of file Appliance_Large_CSV.cpp.

Here is the call graph for this function:



15.10.3.2 step()

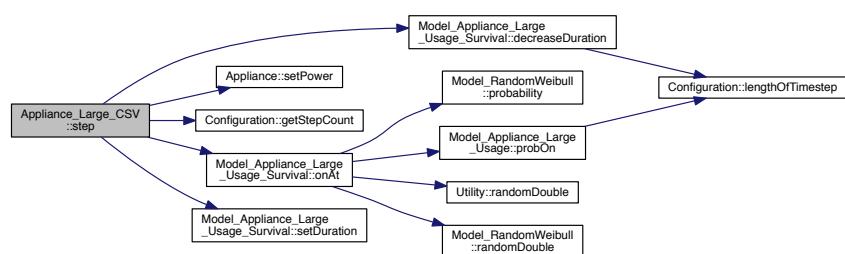
```
void Appliance_Large_CSV::step ( )
```

Check large appliance model for a turn on, then generate the profile.

Calculate if the appliance is predicted a turn on if so increment model and save the power demand until turn off

Definition at line 18 of file Appliance_Large_CSV.cpp.

Here is the call graph for this function:



15.10.4 Member Data Documentation

15.10.4.1 count

```
unsigned int Appliance_Large_CSV::count [protected]
```

Definition at line 24 of file Appliance_Large_CSV.hpp.

15.10.4.2 running

```
bool Appliance_Large_CSV::running [protected]
```

Definition at line 25 of file Appliance_Large_CSV.hpp.

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

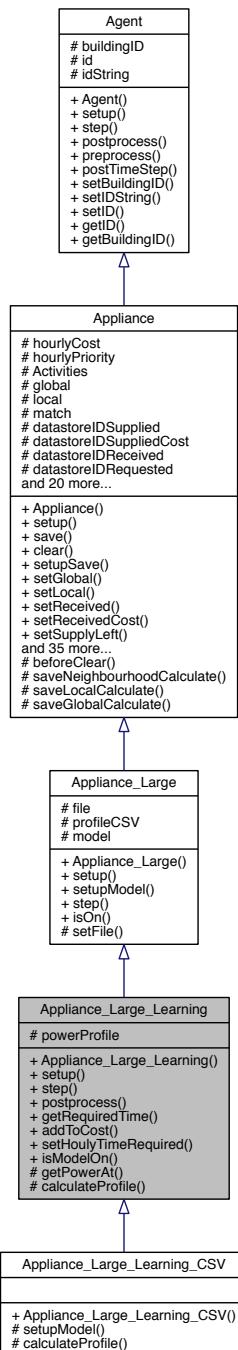
- `Appliance_Large_CSV.hpp`
- `Appliance_Large_CSV.cpp`

15.11 Appliance_Large_Learning Class Reference

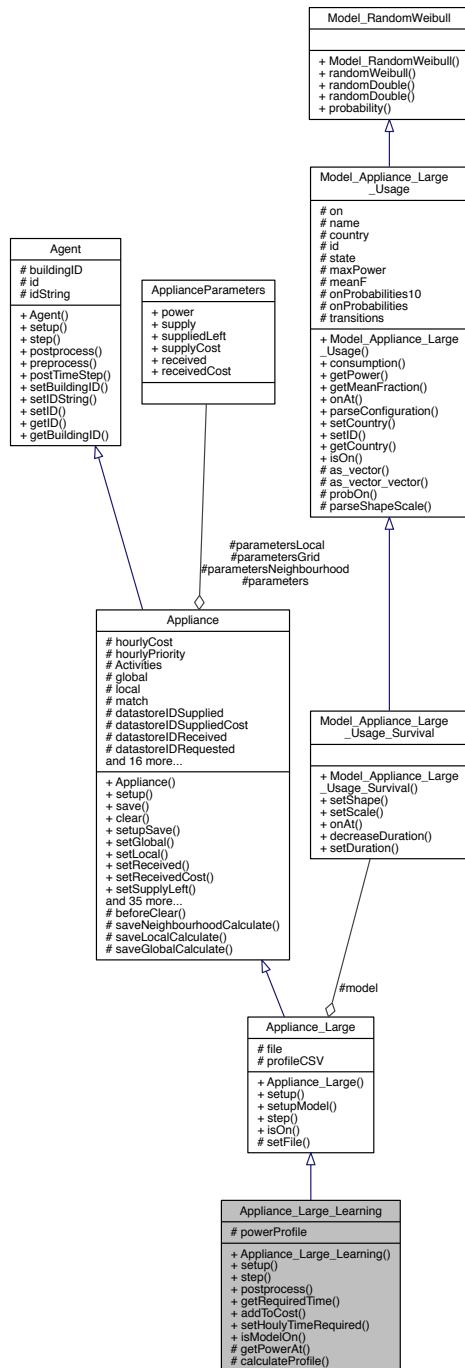
Large appliances learning class.

```
#include <Appliance_Large_Learning.hpp>
```

Inheritance diagram for Appliance_Large_Learning:



Collaboration diagram for Appliance_Large_Learning:



Public Member Functions

- `Appliance_Large_Learning ()`
- void `setup (ConfigStructAppliance a)`
Set up the large appliance model, reading in the large appliance configuration file.
- void `step ()`
- void `postprocess ()`

Appliance postprocess, calls Q-Learning save data.

- double `getRequiredTime` (int hourOfDay) const
- void `addToCost` (const double cost)

Appliance cost is added to for calculating the reward function.

- void `setHourlyTimeRequired` (const std::vector< double > &hourlyTimeRequired)
- bool `isModelOn` ()

Protected Member Functions

- virtual double `getPowerAt` (const int timestep)
- virtual void `calculateProfile` ()

Check large appliance model for a turn on, then generate the profile.

Protected Attributes

- std::queue< `profileStruct` > `powerProfile`

15.11.1 Detailed Description

Large appliances learning class.

This will handle the appliance learning model for profile shifting

Definition at line 29 of file Appliance_Large_Learning.hpp.

15.11.2 Constructor & Destructor Documentation

15.11.2.1 `Appliance_Large_Learning()`

```
Appliance_Large_Learning::Appliance_Large_Learning ( )
```

Definition at line 10 of file Appliance_Large_Learning.cpp.

15.11.3 Member Function Documentation

15.11.3.1 `addToCost()`

```
void Appliance_Large_Learning::addToCost (
    const double cost )
```

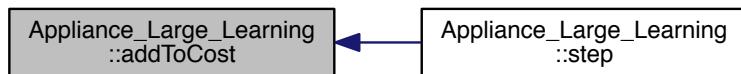
Appliance cost is added to for calculating the reward function.

Parameters

<code>cost</code>	cost of using the appliance
-------------------	-----------------------------

Definition at line 183 of file Appliance_Large_Learning.cpp.

Here is the caller graph for this function:



15.11.3.2 calculateProfile()

```
void Appliance_Large_Learning::calculateProfile( ) [protected], [virtual]
```

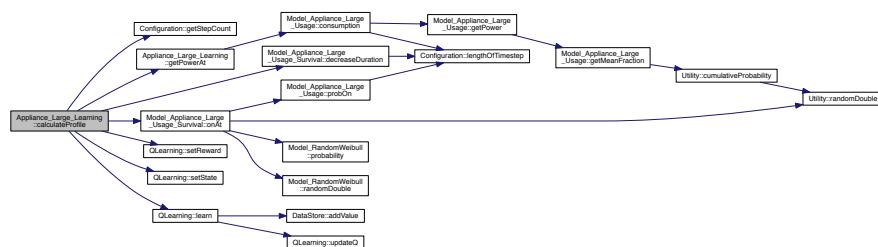
Check large appliance model for a turn on, then generate the profile.

Calculate if the appliance is predicted a turn on if so increment model and save the power demand until turn off

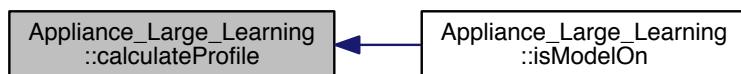
Reimplemented in [Appliance_Large_Learning_CSV](#).

Definition at line 55 of file Appliance_Large_Learning.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.11.3.3 getPowerAt()

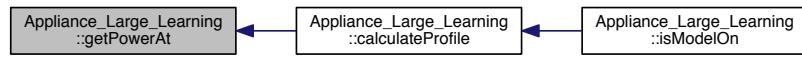
```
double Appliance_Large_Learning::getPowerAt (
    const int timestep ) [protected], [virtual]
```

Definition at line 46 of file Appliance_Large_Learning.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.11.3.4 getRequiredTime()

```
double Appliance_Large_Learning::getRequiredTime (
    int hourOfDay ) const
```

Definition at line 114 of file Appliance_Large_Learning.cpp.

Here is the caller graph for this function:

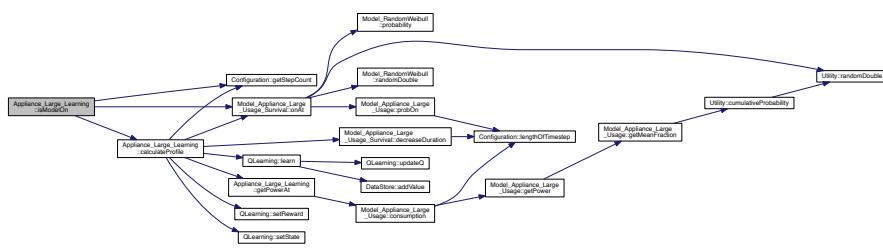


15.11.3.5 isModelOn()

```
bool Appliance_Large_Learning::isModelOn ( )
```

Definition at line 161 of file Appliance_Large_Learning.cpp.

Here is the call graph for this function:



15.11.3.6 postprocess()

```
void Appliance_Large_Learning::postprocess ( )
```

Appliance postprocess, calls Q-Learning save data.

Definition at line 192 of file Appliance_Large_Learning.cpp.

Here is the call graph for this function:



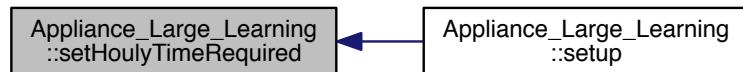
15.11.3.7 setHourlyTimeRequired()

```
void Appliance_Large_Learning::setHourlyTimeRequired (
```

const std::vector< double > & hourlyTimeRequired)

Definition at line 196 of file Appliance_Large_Learning.cpp.

Here is the caller graph for this function:



15.11.3.8 setup()

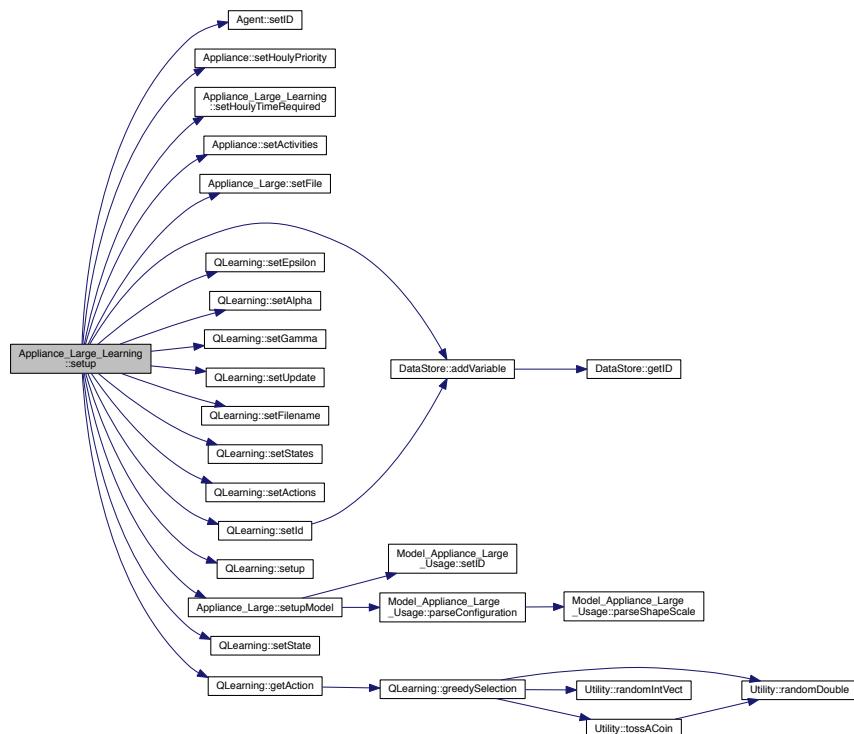
```
void Appliance_Large_Learning::setup (
```

Set up the large appliance model, reading in the large appliance configuration file.

Reimplemented from [Appliance_Large](#).

Definition at line 12 of file Appliance_Large_Learning.cpp.

Here is the call graph for this function:

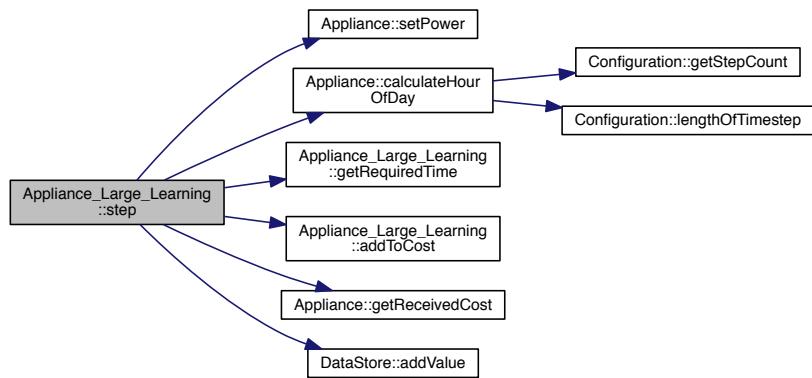


15.11.3.9 step()

```
void Appliance_Large_Learning::step ( )
```

Definition at line 118 of file Appliance_Large_Learning.cpp.

Here is the call graph for this function:



15.11.4 Member Data Documentation

15.11.4.1 powerProfile

```
std::queue<profileStruct> Appliance_Large_Learning::powerProfile [protected]
```

Definition at line 45 of file Appliance_Large_Learning.hpp.

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

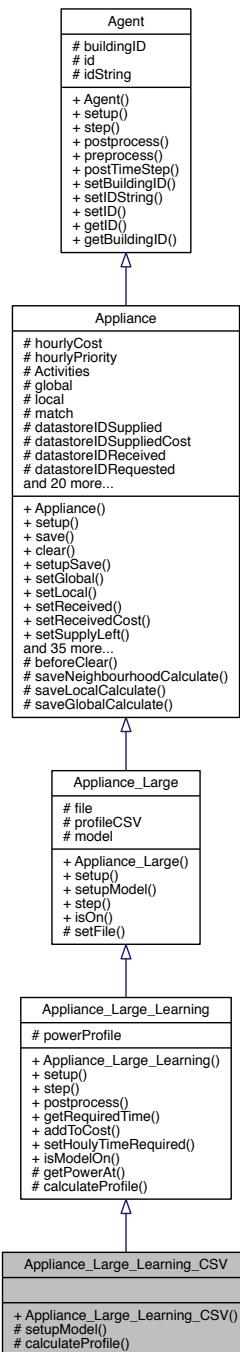
- Appliance_Large_Learning.hpp
- Appliance_Large_Learning.cpp

15.12 Appliance_Large_Learning_CSV Class Reference

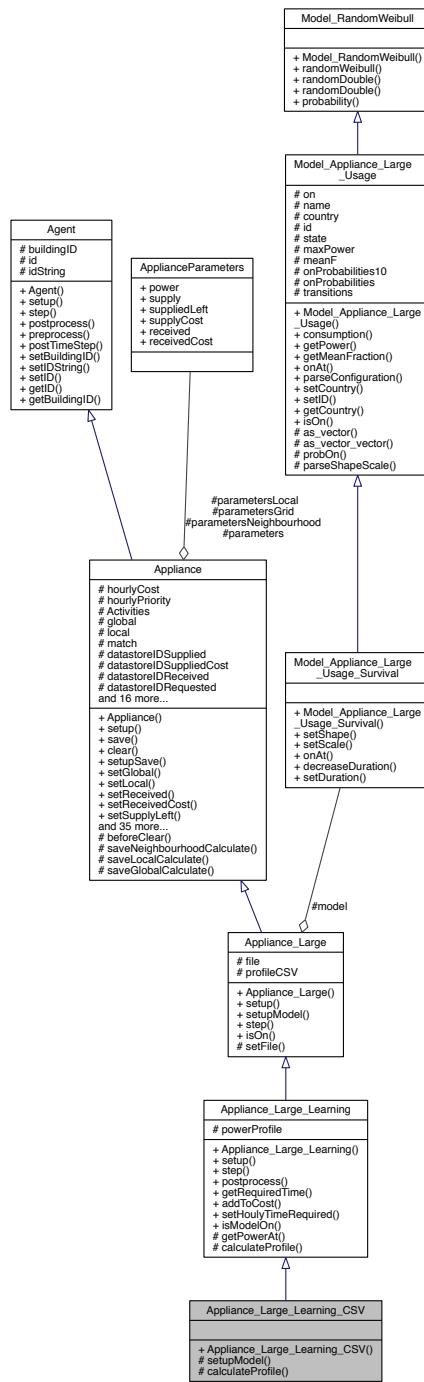
Large appliances learning class with profile taken from CSV.

```
#include <Appliance_Large_Learning_CSV.hpp>
```

Inheritance diagram for Appliance_Large_Learning_CSV:



Collaboration diagram for Appliance_Large_Learning_CS:



Public Member Functions

- Appliance_Large_Learning_CSV ()

Protected Member Functions

- void **setupModel** ()

Set up the large appliance model, reading in the large appliance configuration file.

- void [calculateProfile \(\)](#)

Check large appliance model for a turn on, then generate the profile.

Additional Inherited Members

15.12.1 Detailed Description

Large appliances learning class with profile taken from CSV.

This will handle the appliance learning model for profile shifting however the profile is now take from a csv file

Definition at line 17 of file Appliance_Large_Learning_CSV.hpp.

15.12.2 Constructor & Destructor Documentation

15.12.2.1 [Appliance_Large_Learning_CSV\(\)](#)

```
Appliance_Large_Learning_CSV::Appliance_Large_Learning_CSV ( )
```

Definition at line 9 of file Appliance_Large_Learning_CSV.cpp.

15.12.3 Member Function Documentation

15.12.3.1 [calculateProfile\(\)](#)

```
void Appliance_Large_Learning_CSV::calculateProfile ( ) [protected], [virtual]
```

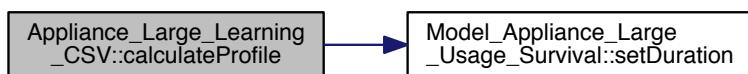
Check large appliance model for a turn on, then generate the profile.

Calculate if the appliance is predicted a turn on if so increment model and save the power demand until turn off

Reimplemented from [Appliance_Large_Learning](#).

Definition at line 16 of file Appliance_Large_Learning_CSV.cpp.

Here is the call graph for this function:



15.12.3.2 setupModel()

```
void Appliance_Large_Learning_CSV::setupModel ( ) [protected], [virtual]
```

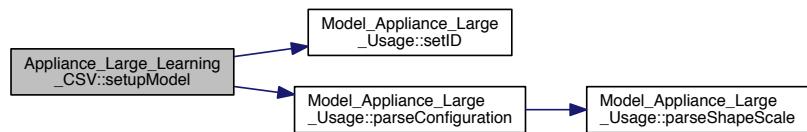
Set up the large appliance model, reading in the large appliance configuration file.

Sets the large appliance configuration file and gives the model the id of the appliance in the file

Reimplemented from [Appliance_Large](#).

Definition at line 30 of file `Appliance_Large_Learning_CSV.cpp`.

Here is the call graph for this function:



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

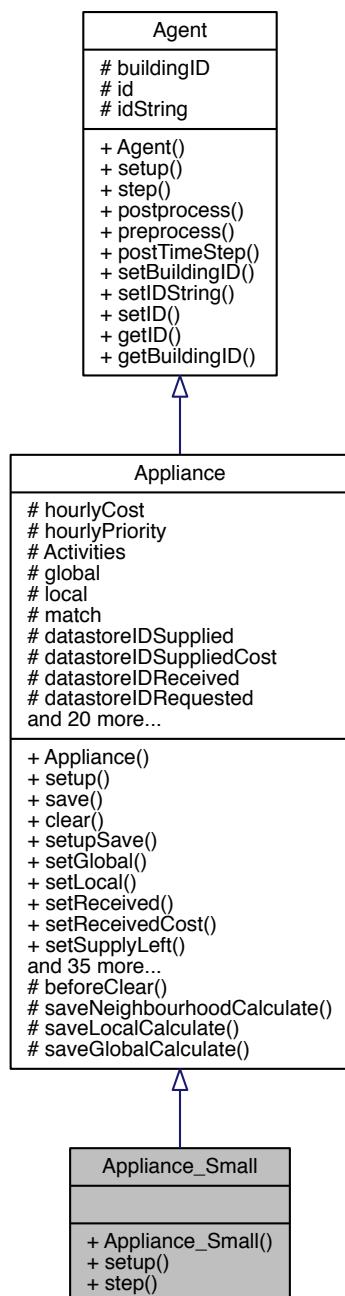
- `Appliance_Large_Learning_CSV.hpp`
- `Appliance_Large_Learning_CSV.cpp`

15.13 Appliance_Small Class Reference

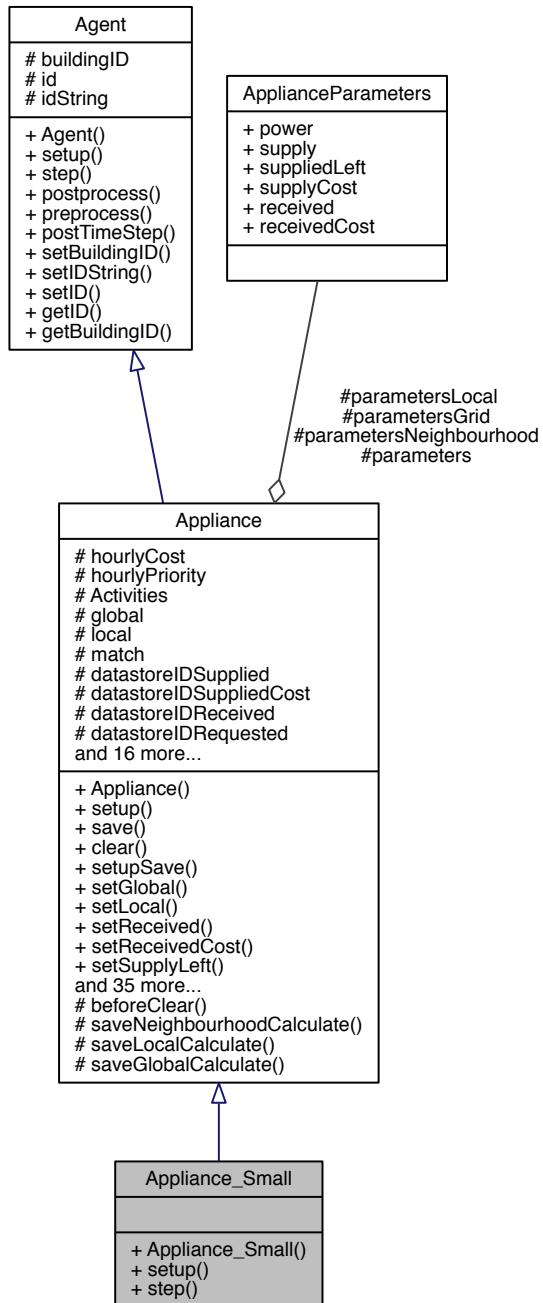
Small appliances class.

```
#include <Appliance_Small.hpp>
```

Inheritance diagram for Appliance_Small:



Collaboration diagram for Appliance_Small:



Public Member Functions

- [Appliance_Small \(\)](#)
- void [setup \(ConfigStructAppliance a\)](#)
- void [step \(\)](#)

Additional Inherited Members

15.13.1 Detailed Description

Small appliances class.

The small appliance agent, handles the model

Definition at line 16 of file Appliance_Small.hpp.

15.13.2 Constructor & Destructor Documentation

15.13.2.1 Appliance_Small()

```
Appliance_Small::Appliance_Small ( )
```

Definition at line 10 of file Appliance_Small.cpp.

15.13.3 Member Function Documentation

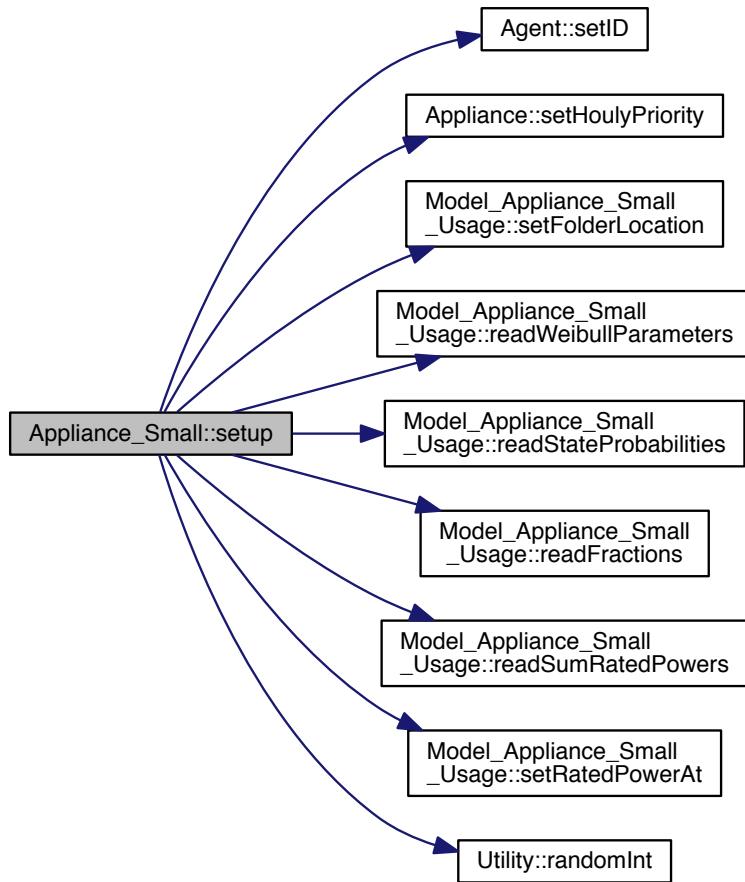
15.13.3.1 setup()

```
void Appliance_Small::setup (
    ConfigStructAppliance a ) [virtual]
```

Implements [Appliance](#).

Definition at line 12 of file Appliance_Small.cpp.

Here is the call graph for this function:

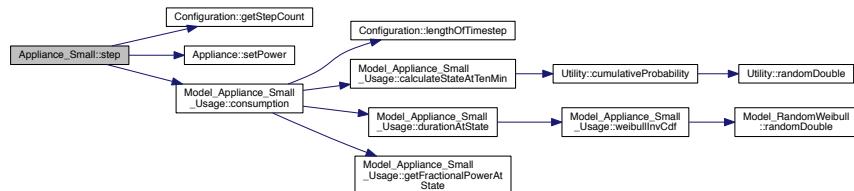


15.13.3.2 step()

```
void Appliance_Small::step( )
```

Definition at line 23 of file `Appliance_Small.cpp`.

Here is the call graph for this function:



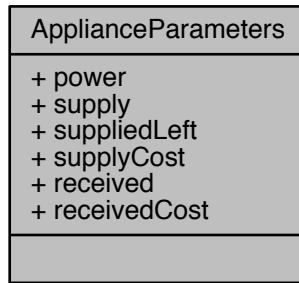
The documentation for this class was generated from the following files:

- Appliance_Small.hpp
- Appliance_Small.cpp

15.14 ApplianceParameters Struct Reference

```
#include <Appliance.hpp>
```

Collaboration diagram for ApplianceParameters:



Public Attributes

- double **power** = 0.0
- double **supply** = 0.0
- double **suppliedLeft** = 0.0
- double **supplyCost** = 0.0
- double **received** = 0.0
- double **receivedCost** = 0.0

15.14.1 Detailed Description

The basic parameters need by the appliances

Definition at line 14 of file Appliance.hpp.

15.14.2 Member Data Documentation

15.14.2.1 power

```
double ApplianceParameters::power = 0.0
```

The power requested

Definition at line 15 of file Appliance.hpp.

15.14.2.2 received

```
double ApplianceParameters::received = 0.0
```

The power received

Definition at line 19 of file Appliance.hpp.

15.14.2.3 receivedCost

```
double ApplianceParameters::receivedCost = 0.0
```

The cost of power received

Definition at line 20 of file Appliance.hpp.

15.14.2.4 suppliedLeft

```
double ApplianceParameters::suppliedLeft = 0.0
```

The power supplied after consumed by other appliances

Definition at line 17 of file Appliance.hpp.

15.14.2.5 supply

```
double ApplianceParameters::supply = 0.0
```

The power supplied

Definition at line 16 of file Appliance.hpp.

15.14.2.6 supplyCost

```
double ApplianceParameters::supplyCost = 0.0
```

The cost of the supplied power

Definition at line 18 of file Appliance.hpp.

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

- Appliance.hpp

15.15 Building Class Reference

A Building.

```
#include <Building.hpp>
```

Collaboration diagram for Building:

Building
+ Building() + setup() + step() + postprocess() + preprocess() + postTimeStep() + stepAppliancesUse() + stepAppliancesUseBatteries() + stepAppliancesNegotiation Neighbourhood() + stepAppliancesNegotiation() and 6 more...

Public Member Functions

- **Building ()**
- void **setup** (const ConfigStructBuilding &buildingInput)

Set up the building from the configuration struct.
- void **step** ()

The buildings timestep function.
- void **postprocess** ()

calls the postprocess functions on the occupants and appliances
- void **preprocess** ()

- building preprocess*
- void [postTimeStep \(\)](#)
calls the postime functions on the occupants and appliances
 - void [stepAppliancesUse \(\)](#)
Steps over the appliances locally, then calls the contract negotiate to resolve local demand and supply.
 - void [stepAppliancesUseBatteries \(Contract_Negotiation *building_negotiation\)](#)
 - void [stepAppliancesNegotiationNeighbourhood \(const Contract_Negotiation &building_negotiation\)](#)
Steps over the appliances on a Neighbourhood level.
 - void [stepAppliancesNegotiation \(const Contract_Negotiation &building_negotiation\)](#)
Steps over the appliances on a Grid level.
 - void [addContactsTo \(Contract_Negotiation *building_negotiation, const bool battery\)](#)
 - bool [decisionBoolean \(const double val1, const double val2\) const](#)
takes two values and return true if val1 is less than val2
 - bool [hasZone \(const std::string &zoneName\) const](#)
checks if a building has a given zone
 - int [getID \(\) const](#)
 - double [getPower \(\) const](#)
returns the buildings appliance use
 - double [decisionDoubleVec \(const std::vector< double > &val, const std::vector< double > &power, const double currentState\) const](#)
Loops over the values and chooses the value with the highest power.

15.15.1 Detailed Description

A Building.

A Building

Definition at line 17 of file Building.hpp.

15.15.2 Constructor & Destructor Documentation

15.15.2.1 Building()

Building::Building ()

Definition at line 16 of file Building.cpp.

15.15.3 Member Function Documentation

15.15.3.1 addContactsTo()

```
void Building::addContactsTo (
    Contract_Negotiation * building_negotiation,
    const bool battery )
```

Definition at line 456 of file Building.cpp.

Here is the call graph for this function:



15.15.3.2 decisionBoolean()

```
bool Building::decisionBoolean (
    const double val1,
    const double val2 ) const
```

takes two values and return true if val1 is less than val2

If they are equal toss a coin

Parameters

<i>val1</i>	The first value
<i>val2</i>	The second value

Returns

true if *val1* < *val2*, or random if equal

Definition at line 367 of file Building.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.15.3.3 decisionDoubleVec()

```

double Building::decisionDoubleVec (
    const std::vector< double > & val,
    const std::vector< double > & power,
    const double currentState ) const
  
```

Loops over the values and chooses the value with the highest power.

Loops over the values and calculates if the choice is an increase or decrease on the current state chooses whether to stay the same, increase, or decrease based on the highest value if the power is equal for two or more options then toss a coin.

Parameters

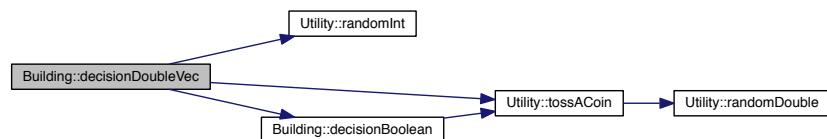
<i>val</i>	Occupant desires
<i>power</i>	Occupant power
<i>currentState</i>	current state of the device

Returns

The final negotiated choice for the device

Definition at line 172 of file Building.cpp.

Here is the call graph for this function:



15.15.3.4 getID()

```
int Building::getID ( ) const
```

Definition at line 452 of file Building.cpp.

15.15.3.5 getPower()

```
double Building::getPower ( ) const
```

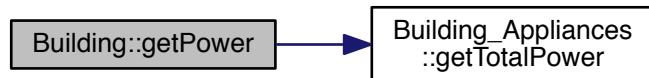
returns the buildings appliance use

Returns

building power useage

Definition at line 85 of file Building.cpp.

Here is the call graph for this function:



15.15.3.6 hasZone()

```
bool Building::hasZone (   
    const std::string & zoneName ) const
```

checks if a building has a given zone

Parameters

<code>zoneName</code>	name of the zone
-----------------------	------------------

Returns

true if the zone is in the building

Definition at line 442 of file Building.cpp.

15.15.3.7 postprocess()

```
void Building::postprocess ( )
```

calls the postprocess functions on the occupants and appliances

Definition at line 430 of file Building.cpp.

Here is the call graph for this function:



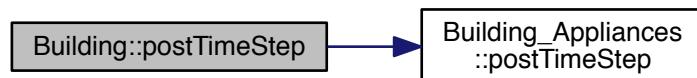
15.15.3.8 postTimeStep()

```
void Building::postTimeStep ( )
```

calls the posttime functions on the occupants and appliances

Definition at line 420 of file Building.cpp.

Here is the call graph for this function:



15.15.3.9 preprocess()

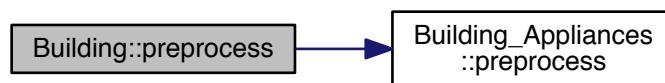
```
void Building::preprocess ( )
```

building preprocess

calls the building appliances preprocess

Definition at line 52 of file Building.cpp.

Here is the call graph for this function:



15.15.3.10 setup()

```
void Building::setup (
    const ConfigStructBuilding & buildingInput )
```

Set up the building from the configuration struct.

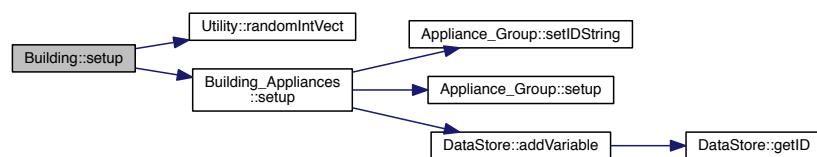
Setup the building, adds the zones, adds the occupants, adds the appliances

Parameters

<code>buildingInput</code>	The configuration struct which values have been populated form configuration file
----------------------------	---

Definition at line 23 of file Building.cpp.

Here is the call graph for this function:



15.15.3.11 step()

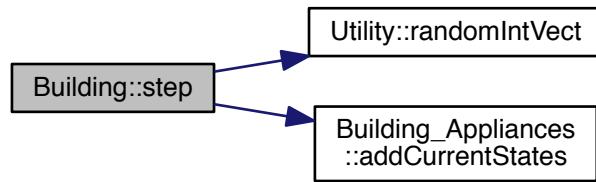
```
void Building::step ( )
```

The buildings timestep function.

calls the timestep function for the appliance, occupants and zones. Also calls the functions to run the social interactions between occupants, and between occupants and appliances.

Definition at line 94 of file Building.cpp.

Here is the call graph for this function:



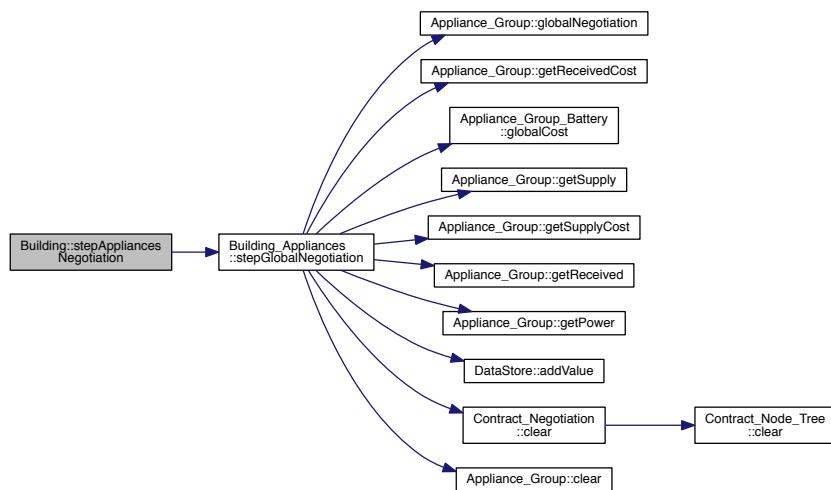
15.15.3.12 stepAppliancesNegotiation()

```
void Building::stepAppliancesNegotiation (
    const Contract_Negotiation & building_negotiation )
```

Steps over the appliances on a Grid level.

Definition at line 76 of file Building.cpp.

Here is the call graph for this function:



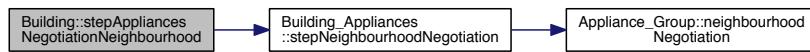
15.15.3.13 stepAppliancesNegotiationNeighbourhood()

```
void Building::stepAppliancesNegotiationNeighbourhood (
    const Contract_Negotiation & building_negotiation )
```

Steps over the appliances on a Neighbourhood level.

Definition at line 68 of file Building.cpp.

Here is the call graph for this function:



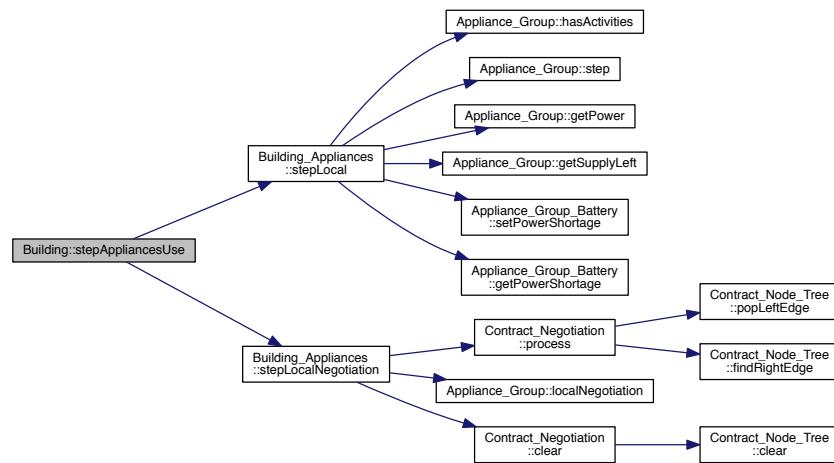
15.15.3.14 stepAppliancesUse()

```
void Building::stepAppliancesUse ( )
```

Steps over the appliances locally, then calls the contract negotiate to resolve local demand and supply.

Definition at line 60 of file Building.cpp.

Here is the call graph for this function:

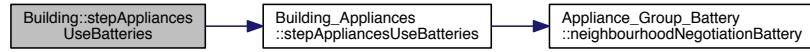


15.15.3.15 stepAppliancesUseBatteries()

```
void Building::stepAppliancesUseBatteries (
    Contract_Negotiation * building_negotiation )
```

Definition at line 460 of file Building.cpp.

Here is the call graph for this function:



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

- Building.hpp
- Building.cpp

15.16 Building_Appliances Class Reference

Manages the different building appliances agents.

```
#include <Building_Appliances.hpp>
```

Collaboration diagram for Building_Appliances:

Building_Appliances
+ Building_Appliances() + setup() + stepLocal() + stepAppliancesUseBatteries() + stepLocalNegotiation() + stepGlobalNegotiation() + stepNeighbourhoodNegotiation() + postprocess() + preprocess() + postTimeStep() + addCurrentStates() + addContactsTo() + getTotalPower()

Public Member Functions

- `Building_Appliances ()`
- `void setup (const ConfigStructBuilding &b)`
- `void stepLocal ()`
- `void stepAppliancesUseBatteries (Contract_Negotiation *building_negotiation)`
- `void stepLocalNegotiation ()`
- `void stepGlobalNegotiation (const Contract_Negotiation &building_negotiation)`
- `void stepNeighbourhoodNegotiation (const Contract_Negotiation &building_negotiation)`
- `void postprocess ()`
- `void preprocess ()`
- `void postTimeStep ()`
- `void addCurrentStates (const int stateid)`
- `void addContactsTo (Contract_Negotiation *building_negotiation, const bool battery)`
- `double getTotalPower () const`

15.16.1 Detailed Description

Manages the different building appliances agents.

Each building has a set of appliance this class manages them

Definition at line 27 of file Building_Appliances.hpp.

15.16.2 Constructor & Destructor Documentation

15.16.2.1 Building_Appliances()

```
Building_Appliances::Building_Appliances ( )
```

Definition at line 11 of file Building_Appliances.cpp.

15.16.3 Member Function Documentation

15.16.3.1 addContactsTo()

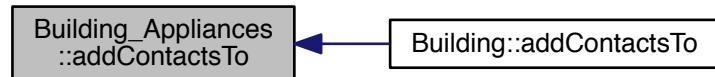
```
void Building_Appliances::addContactsTo (
    Contract_Negotiation * building_negotiation,
    const bool battery )
```

Definition at line 197 of file Building_Appliances.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.16.3.2 addCurrentStates()

```
void Building_Appliances::addCurrentStates (
    const int stateid )
```

Definition at line 193 of file Building_Appliances.cpp.

Here is the caller graph for this function:

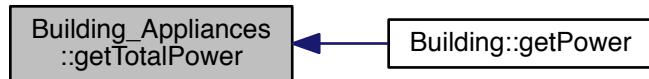


15.16.3.3 getTotalPower()

```
double Building_Appliances::getTotalPower ( ) const
```

Definition at line 189 of file Building_Appliances.cpp.

Here is the caller graph for this function:

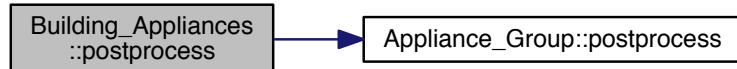


15.16.3.4 postprocess()

```
void Building_Appliances::postprocess ( )
```

Definition at line 179 of file Building_Appliances.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.16.3.5 postTimeStep()

```
void Building_Appliances::postTimeStep ( )
```

Definition at line 186 of file Building_Appliances.cpp.

Here is the caller graph for this function:



15.16.3.6 preprocess()

```
void Building_Appliances::preprocess ( )
```

Definition at line 44 of file Building_Appliances.cpp.

Here is the caller graph for this function:

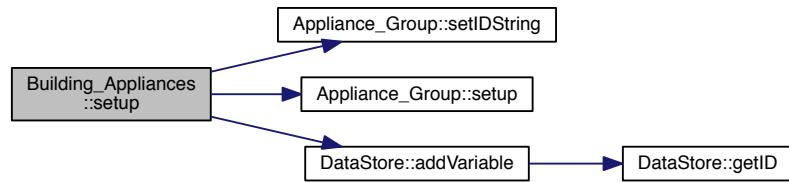


15.16.3.7 setup()

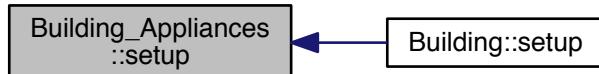
```
void Building_Appliances::setup ( const ConfigStructBuilding & b )
```

Definition at line 13 of file Building_Appliances.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

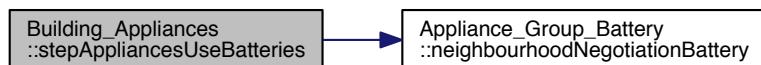


15.16.3.8 stepAppliancesUseBatteries()

```
void Building_Appliances::stepAppliancesUseBatteries (
    Contract_Negotiation * building_negotiation )
```

Definition at line 213 of file `Building_Appliances.cpp`.

Here is the call graph for this function:



Here is the caller graph for this function:

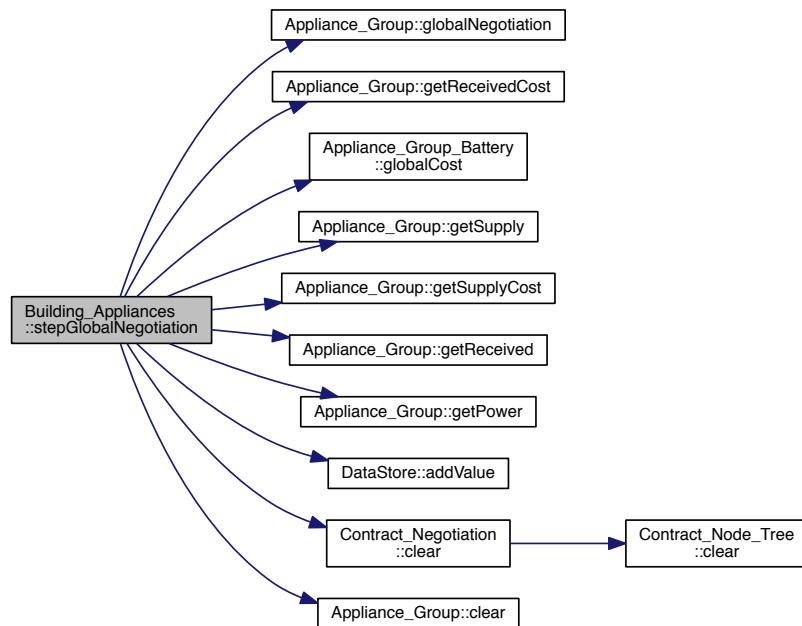


15.16.3.9 stepGlobalNegotiation()

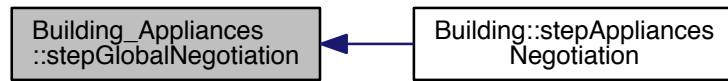
```
void Building_Appliances::stepGlobalNegotiation (
    const Contract_Negotiation & building_negotiation )
```

Definition at line 109 of file Building_Appliances.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

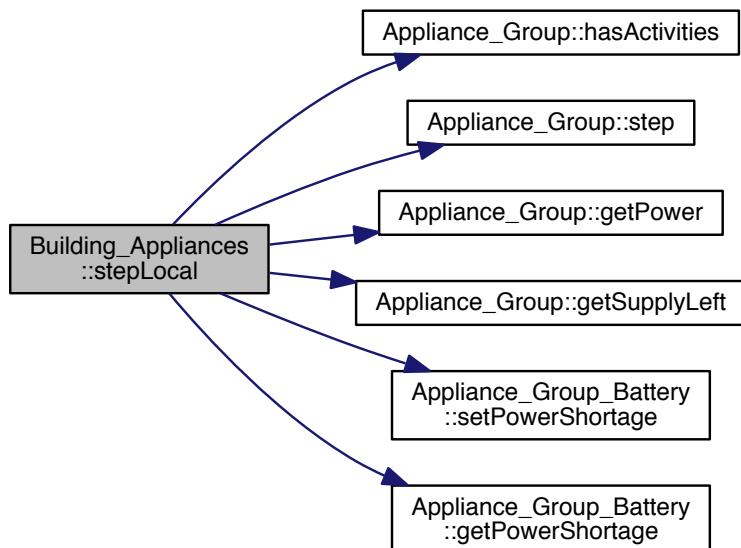


15.16.3.10 stepLocal()

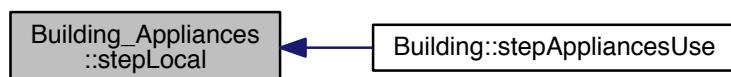
```
void Building_Appliances::stepLocal ( )
```

Definition at line 46 of file Building_Appliances.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

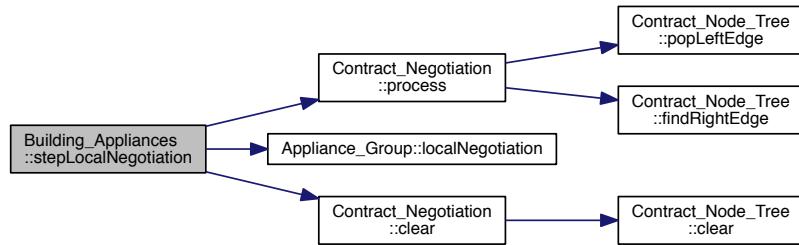


15.16.3.11 stepLocalNegotiation()

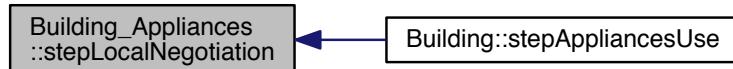
```
void Building_Appliances::stepLocalNegotiation ( )
```

Definition at line 81 of file Building_Appliances.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.16.3.12 stepNeighbourhoodNegotiation()

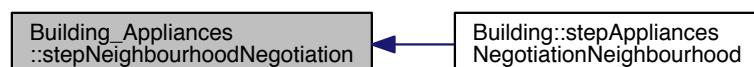
```
void Building_Appliances::stepNeighbourhoodNegotiation (
    const Contract_Negotiation & building_negotiation )
```

Definition at line 95 of file `Building_Appliances.cpp`.

Here is the call graph for this function:



Here is the caller graph for this function:



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

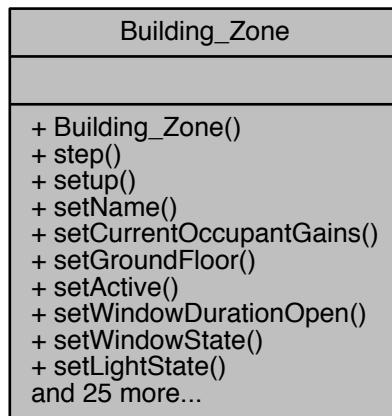
- Building_Appliances.hpp
- Building_Appliances.cpp

15.17 Building_Zone Class Reference

A zone within a building.

```
#include <Building_Zone.hpp>
```

Collaboration diagram for Building_Zone:



Public Member Functions

- [Building_Zone \(\)](#)
- [void step \(\)](#)
the zones step function
- [void setup \(const ConfigStructZone &zoneStruct\)](#)
Set up the zone from the configuration struct.
- [void setName \(const std::string &name\)](#)
- [void setCurrentOccupantGains \(double currentOccupantGains\)](#)
- [void setGroundFloor \(bool groundFloor\)](#)
- [void setActive \(bool active\)](#)
- [void setWindowDurationOpen \(double windowDurationOpen\)](#)
- [void setWindowState \(bool windowState\)](#)
- [void setLightState \(bool lightState\)](#)
- [void setBlindState \(double state\)](#)
- [void setHeatingState \(double state\)](#)
- [void setOccupantFraction \(double occupantFraction\)](#)
- [void setAppFraction \(double appFraction\)](#)

- void `setIdString` (const std::string &idString)
- bool `isActive` () const
- bool `hasActivity` (int activity) const
- bool `isNamed` (const std::string &name) const
- int `getCurrentOccupantCount` () const
- int `getId` () const
- int `getNumberOfActivities` () const
- float `getOccupantFraction` () const
- double `getCurrentOccupantGains` () const
- double `getWindowState` () const
- double `getBlindState` () const
- double `getHeatingState` () const
- double `getGroundFloor` () const
- double `getAirSystemSensibleHeatingRate` () const
- double `getLightState` () const
- double `getMeanAirTemperature` () const
- double `getAirRelativeHumidity` () const
- double `getMeanRadiantTemperature` () const
- double `getDaylightingReferencePoint1Illuminance` () const
- double `getWindowDurationOpen` () const
- std::vector< int > `getActivities` () const

15.17.1 Detailed Description

A zone within a building.

A zone within a building

Definition at line 16 of file Building_Zone.hpp.

15.17.2 Constructor & Destructor Documentation

15.17.2.1 Building_Zone()

`Building_Zone::Building_Zone ()`

Definition at line 12 of file Building_Zone.cpp.

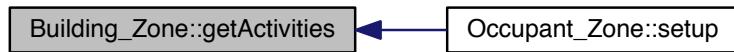
15.17.3 Member Function Documentation

15.17.3.1 getActivities()

```
std::vector< int > Building_Zone::getActivities ( ) const
```

Definition at line 118 of file Building_Zone.cpp.

Here is the caller graph for this function:



15.17.3.2 getAirRelativeHumidity()

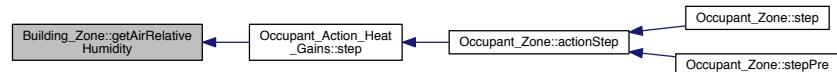
```
double Building_Zone::getAirRelativeHumidity ( ) const
```

Definition at line 130 of file Building_Zone.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.17.3.3 getAirSystemSensibleHeatingRate()

```
double Building_Zone::getAirSystemSensibleHeatingRate ( ) const
```

Definition at line 142 of file Building_Zone.cpp.

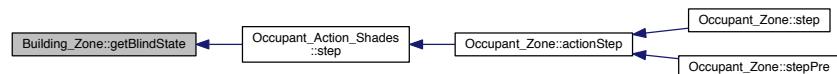
Here is the call graph for this function:

**15.17.3.4 getBlindState()**

```
double Building_Zone::getBlindState ( ) const
```

Definition at line 110 of file Building_Zone.cpp.

Here is the caller graph for this function:

**15.17.3.5 getCurrentOccupantCount()**

```
int Building_Zone::getCurrentOccupantCount ( ) const
```

15.17.3.6 getCurrentOccupantGains()

```
double Building_Zone::getCurrentOccupantGains ( ) const
```

Definition at line 102 of file Building_Zone.cpp.

15.17.3.7 `getDaylightingReferencePoint1Illuminance()`

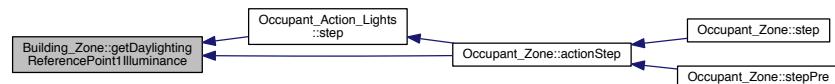
```
double Building_Zone::getDaylightingReferencePoint1Illuminance ( ) const
```

Definition at line 138 of file Building_Zone.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

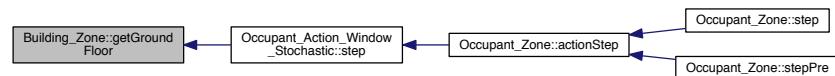


15.17.3.8 `getGroundFloor()`

```
double Building_Zone::getGroundFloor ( ) const
```

Definition at line 167 of file Building_Zone.cpp.

Here is the caller graph for this function:

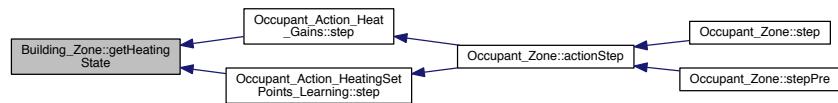


15.17.3.9 getHeatingState()

```
double Building_Zone::getHeatingState ( ) const
```

Definition at line 114 of file Building_Zone.cpp.

Here is the caller graph for this function:

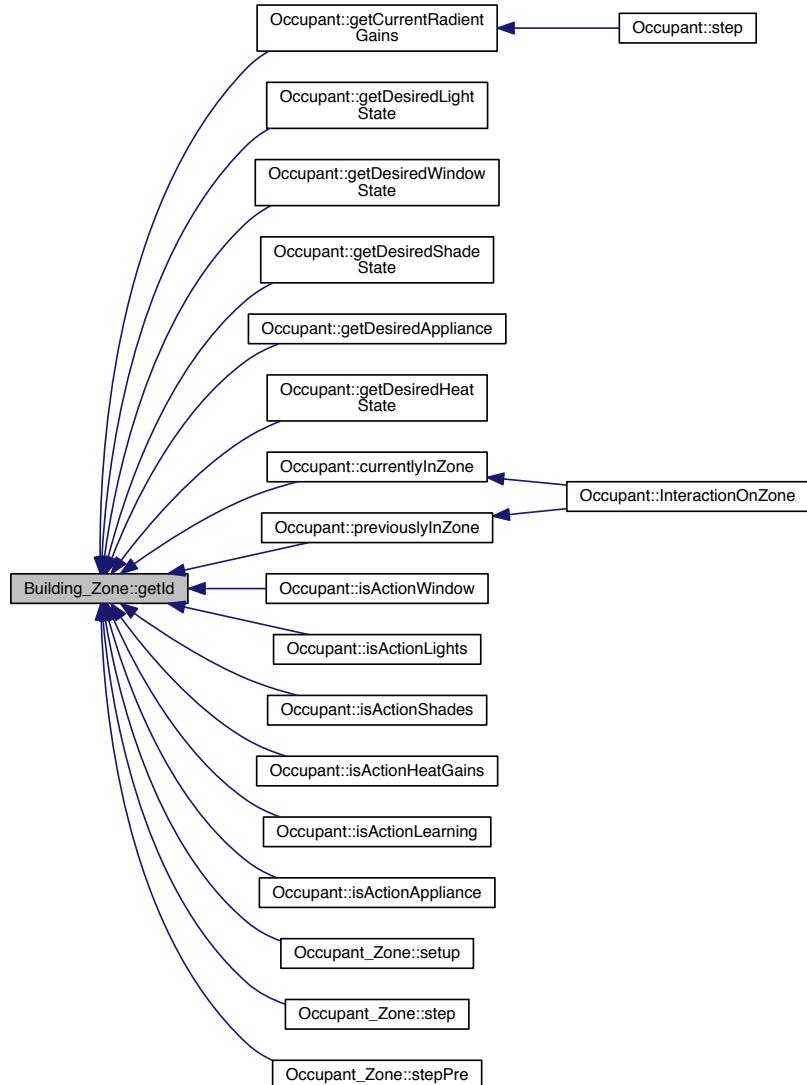


15.17.3.10 getId()

```
int Building_Zone::getId ( ) const
```

Definition at line 94 of file Building_Zone.cpp.

Here is the caller graph for this function:

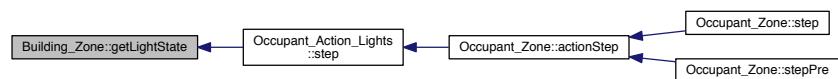


15.17.3.11 `getLightState()`

```
double Building_Zone::getLightState ( ) const
```

Definition at line 179 of file `Building_Zone.cpp`.

Here is the caller graph for this function:

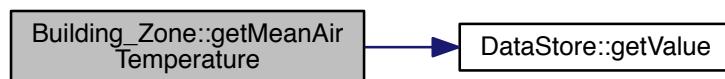


15.17.3.12 getMeanAirTemperature()

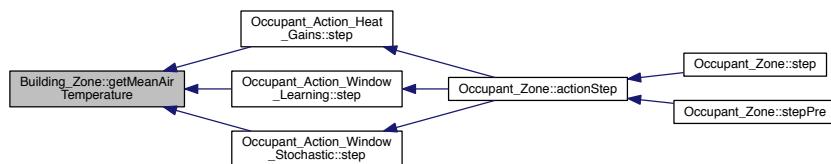
```
double Building_Zone::getMeanAirTemperature ( ) const
```

Definition at line 126 of file Building_Zone.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.17.3.13 getMeanRadiantTemperature()

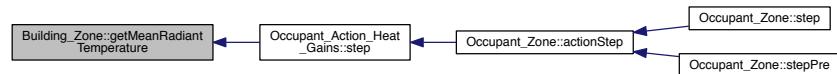
```
double Building_Zone::getMeanRadiantTemperature ( ) const
```

Definition at line 134 of file Building_Zone.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.17.3.14 getNumberOfActivities()

```
int Building_Zone::getNumberOfActivities ( ) const
```

Definition at line 195 of file Building_Zone.cpp.

15.17.3.15 getOccupantFraction()

```
float Building_Zone::getOccupantFraction ( ) const
```

Definition at line 146 of file Building_Zone.cpp.

15.17.3.16 getWindowDurationOpen()

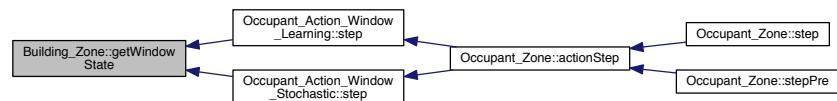
```
double Building_Zone::getWindowDurationOpen ( ) const
```

15.17.3.17 getWindowState()

```
double Building_Zone::getWindowState ( ) const
```

Definition at line 106 of file Building_Zone.cpp.

Here is the caller graph for this function:

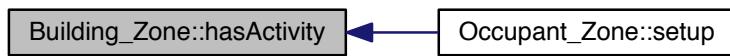


15.17.3.18 hasActivity()

```
bool Building_Zone::hasActivity (  
    int activity ) const
```

Definition at line 158 of file Building_Zone.cpp.

Here is the caller graph for this function:

**15.17.3.19 isActive()**

```
bool Building_Zone::isActive ( ) const
```

Definition at line 154 of file Building_Zone.cpp.

15.17.3.20 isNamed()

```
bool Building_Zone::isNamed (   
    const std::string & name ) const
```

Definition at line 90 of file Building_Zone.cpp.

15.17.3.21 setActive()

```
void Building_Zone::setActive (   
    bool active )
```

Definition at line 150 of file Building_Zone.cpp.

15.17.3.22 setAppFraction()

```
void Building_Zone::setAppFraction (
    double appFraction )
```

Definition at line 191 of file Building_Zone.cpp.

15.17.3.23 setBlindState()

```
void Building_Zone::setBlindState (
    double state )
```

Definition at line 187 of file Building_Zone.cpp.

15.17.3.24 setCurrentOccupantGains()

```
void Building_Zone::setCurrentOccupantGains (
    double currentOccupantGains )
```

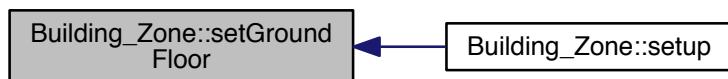
Definition at line 98 of file Building_Zone.cpp.

15.17.3.25 setGroundFloor()

```
void Building_Zone::setGroundFloor (
    bool groundFloor )
```

Definition at line 163 of file Building_Zone.cpp.

Here is the caller graph for this function:



15.17.3.26 setHeatingState()

```
void Building_Zone::setHeatingState (
    double state )
```

Definition at line 175 of file Building_Zone.cpp.

15.17.3.27 setIDString()

```
void Building_Zone::setIDString (
    const std::string & idString )
```

Definition at line 199 of file Building_Zone.cpp.

15.17.3.28 setLightState()

```
void Building_Zone::setLightState (
    bool lightState )
```

Definition at line 171 of file Building_Zone.cpp.

15.17.3.29 setName()

```
void Building_Zone::setName (
    const std::string & name )
```

Definition at line 86 of file Building_Zone.cpp.

15.17.3.30 setOccupantFraction()

```
void Building_Zone::setOccupantFraction (
    double occupantFraction )
```

Definition at line 122 of file Building_Zone.cpp.

15.17.3.31 setup()

```
void Building_Zone::setup (
    const ConfigStructZone & zoneStruct )
```

Set up the zone from the configuration struct.

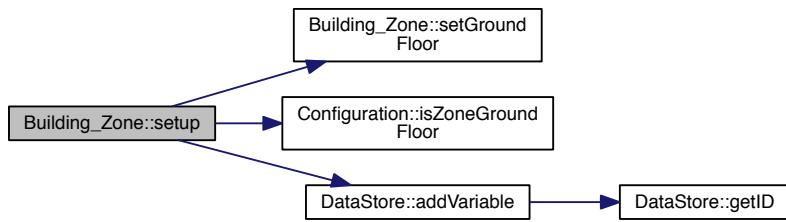
Setup the zone, adds the zone values to the data store

Parameters

<code>zoneStruct</code>	The configuration struct which values have been populated from configuration file
-------------------------	---

Definition at line 19 of file Building_Zone.cpp.

Here is the call graph for this function:



15.17.3.32 setWindowDurationOpen()

```
void Building_Zone::setWindowDurationOpen (
    double windowDurationOpen )
```

15.17.3.33 setWindowState()

```
void Building_Zone::setWindowState (
    bool windowState )
```

Definition at line 183 of file Building_Zone.cpp.

15.17.3.34 step()

```
void Building_Zone::step ( )
```

the zones step function

saves the zone values to the data store at each timestep

Definition at line 66 of file Building_Zone.cpp.

Here is the call graph for this function:



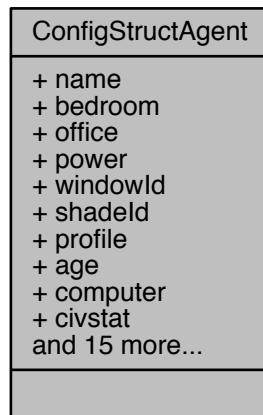
The documentation for this class was generated from the following files:

- Building_Zone.hpp
- Building_Zone.cpp

15.18 ConfigStructAgent Struct Reference

```
#include <Configuration.hpp>
```

Collaboration diagram for ConfigStructAgent:



Public Attributes

- std::string **name**
- std::string **bedroom**
- std::string **office**
- double **power**
- int **windowId**

- int `shadeld`
- std::map< int, std::string > `profile`
- std::string `age`
- std::string `computer`
- std::string `civstat`
- std::string `unemp`
- std::string `retired`
- std::string `edtry`
- std::string `famstat`
- std::string `sex`
- double `ShadeClosedDuringSleep` = 0
- double `ShadeClosedDuringWashing` = 0
- double `ShadeDuringNight` = 0
- double `ShadeDuringAudioVisual` = 0
- double `LightOffDuringAudioVisual` = 0
- double `LightOffDuringSleep` = 0
- double `WindowOpenDuringCooking` = 0
- double `WindowOpenDuringWashing` = 0
- double `WindowOpenDuringSleeping` = 0
- double `ApplianceDuringDay` = 0

15.18.1 Detailed Description

Definition at line 29 of file Configuration.hpp.

15.18.2 Member Data Documentation

15.18.2.1 age

```
std::string ConfigStructAgent::age
```

Definition at line 38 of file Configuration.hpp.

15.18.2.2 ApplianceDuringDay

```
double ConfigStructAgent::ApplianceDuringDay = 0
```

Definition at line 59 of file Configuration.hpp.

15.18.2.3 bedroom

```
std::string ConfigStructAgent::bedroom
```

Definition at line 31 of file Configuration.hpp.

15.18.2.4 civstat

```
std::string ConfigStructAgent::civstat
```

Definition at line 40 of file Configuration.hpp.

15.18.2.5 computer

```
std::string ConfigStructAgent::computer
```

Definition at line 39 of file Configuration.hpp.

15.18.2.6 edtry

```
std::string ConfigStructAgent::edtry
```

Definition at line 43 of file Configuration.hpp.

15.18.2.7 famstat

```
std::string ConfigStructAgent::famstat
```

Definition at line 44 of file Configuration.hpp.

15.18.2.8 LightOffDuringAudioVisual

```
double ConfigStructAgent::LightOffDuringAudioVisual = 0
```

Definition at line 52 of file Configuration.hpp.

15.18.2.9 LightOffDuringSleep

```
double ConfigStructAgent::LightOffDuringSleep = 0
```

Definition at line 53 of file Configuration.hpp.

15.18.2.10 name

```
std::string ConfigStructAgent::name
```

Definition at line 30 of file Configuration.hpp.

15.18.2.11 office

```
std::string ConfigStructAgent::office
```

Definition at line 32 of file Configuration.hpp.

15.18.2.12 power

```
double ConfigStructAgent::power
```

Definition at line 33 of file Configuration.hpp.

15.18.2.13 profile

```
std::map<int, std::string> ConfigStructAgent::profile
```

Definition at line 36 of file Configuration.hpp.

15.18.2.14 retired

```
std::string ConfigStructAgent::retired
```

Definition at line 42 of file Configuration.hpp.

15.18.2.15 sex

```
std::string ConfigStructAgent::sex
```

Definition at line 45 of file Configuration.hpp.

15.18.2.16 ShadeClosedDuringSleep

```
double ConfigStructAgent::ShadeClosedDuringSleep = 0
```

Definition at line 47 of file Configuration.hpp.

15.18.2.17 ShadeClosedDuringWashing

```
double ConfigStructAgent::ShadeClosedDuringWashing = 0
```

Definition at line 48 of file Configuration.hpp.

15.18.2.18 ShadeDuringAudioVisual

```
double ConfigStructAgent::ShadeDuringAudioVisual = 0
```

Definition at line 50 of file Configuration.hpp.

15.18.2.19 ShadeDuringNight

```
double ConfigStructAgent::ShadeDuringNight = 0
```

Definition at line 49 of file Configuration.hpp.

15.18.2.20 shadeId

```
int ConfigStructAgent::shadeId
```

Definition at line 35 of file Configuration.hpp.

15.18.2.21 unemp

```
std::string ConfigStructAgent::unemp
```

Definition at line 41 of file Configuration.hpp.

15.18.2.22 windowId

```
int ConfigStructAgent::windowId
```

Definition at line 34 of file Configuration.hpp.

15.18.2.23 WindowOpenDuringCooking

```
double ConfigStructAgent::WindowOpenDuringCooking = 0
```

Definition at line 55 of file Configuration.hpp.

15.18.2.24 WindowOpenDuringSleeping

```
double ConfigStructAgent::WindowOpenDuringSleeping = 0
```

Definition at line 57 of file Configuration.hpp.

15.18.2.25 WindowOpenDuringWashing

```
double ConfigStructAgent::WindowOpenDuringWashing = 0
```

Definition at line 56 of file Configuration.hpp.

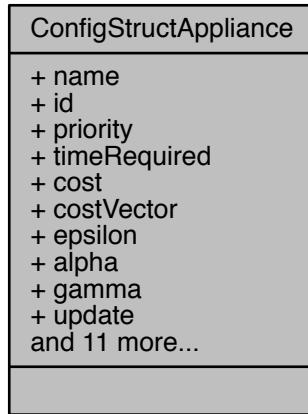
The documentation for this struct was generated from the following file:

- Configuration.hpp

15.19 ConfigStructAppliance Struct Reference

```
#include <Configuration.hpp>
```

Collaboration diagram for ConfigStructAppliance:



Public Attributes

- `std::string name`
- `int id`
- `std::vector< double > priority`
- `std::vector< double > timeRequired = {0, 0}`
- `double cost`
- `std::vector< double > costVector`
- `double epsilon = 0.1`
- `double alpha = 0.3`
- `double gamma = 0.1`
- `bool update = false`
- `std::vector< int > activities`
- `std::string fileSupply`
- `std::string fileDemand`
- `std::string fileProfile`
- `std::string WeibullParameters`
- `std::string StateProbabilities`
- `std::string Fractions`
- `std::string SumRatedPowers`
- `std::string variableName`
- `bool batteryNeighbourhoodDischarge = false`
- `bool batteryNeighbourhoodCharge = false`

15.19.1 Detailed Description

Definition at line 121 of file Configuration.hpp.

15.19.2 Member Data Documentation

15.19.2.1 activities

```
std::vector<int> ConfigStructAppliance::activities
```

Definition at line 136 of file Configuration.hpp.

15.19.2.2 alpha

```
double ConfigStructAppliance::alpha = 0.3
```

Definition at line 132 of file Configuration.hpp.

15.19.2.3 batteryNeighbourhoodCharge

```
bool ConfigStructAppliance::batteryNeighbourhoodCharge = false
```

Definition at line 150 of file Configuration.hpp.

15.19.2.4 batteryNeighbourhoodDischarge

```
bool ConfigStructAppliance::batteryNeighbourhoodDischarge = false
```

Definition at line 149 of file Configuration.hpp.

15.19.2.5 cost

```
double ConfigStructAppliance::cost
```

Definition at line 128 of file Configuration.hpp.

15.19.2.6 costVector

```
std::vector<double> ConfigStructAppliance::costVector
```

Definition at line 129 of file Configuration.hpp.

15.19.2.7 epsilon

```
double ConfigStructAppliance::epsilon = 0.1
```

Definition at line 131 of file Configuration.hpp.

15.19.2.8 fileDemand

```
std::string ConfigStructAppliance::fileDemand
```

Definition at line 139 of file Configuration.hpp.

15.19.2.9 fileProfile

```
std::string ConfigStructAppliance::fileProfile
```

Definition at line 140 of file Configuration.hpp.

15.19.2.10 fileSupply

```
std::string ConfigStructAppliance::fileSupply
```

Definition at line 138 of file Configuration.hpp.

15.19.2.11 Fractions

```
std::string ConfigStructAppliance::Fractions
```

Definition at line 144 of file Configuration.hpp.

15.19.2.12 gamma

```
double ConfigStructAppliance::gamma = 0.1
```

Definition at line 133 of file Configuration.hpp.

15.19.2.13 id

```
int ConfigStructAppliance::id
```

Definition at line 123 of file Configuration.hpp.

15.19.2.14 name

```
std::string ConfigStructAppliance::name
```

Definition at line 122 of file Configuration.hpp.

15.19.2.15 priority

```
std::vector<double> ConfigStructAppliance::priority
```

Definition at line 124 of file Configuration.hpp.

15.19.2.16 StateProbabilities

```
std::string ConfigStructAppliance::StateProbabilities
```

Definition at line 143 of file Configuration.hpp.

15.19.2.17 SumRatedPowers

```
std::string ConfigStructAppliance::SumRatedPowers
```

Definition at line 145 of file Configuration.hpp.

15.19.2.18 timeRequired

```
std::vector<double> ConfigStructAppliance::timeRequired = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

Definition at line 126 of file Configuration.hpp.

15.19.2.19 update

```
bool ConfigStructAppliance::update = false
```

Definition at line 134 of file Configuration.hpp.

15.19.2.20 variableName

```
std::string ConfigStructAppliance::variableName
```

Definition at line 147 of file Configuration.hpp.

15.19.2.21 WeibullParameters

```
std::string ConfigStructAppliance::WeibullParameters
```

Definition at line 142 of file Configuration.hpp.

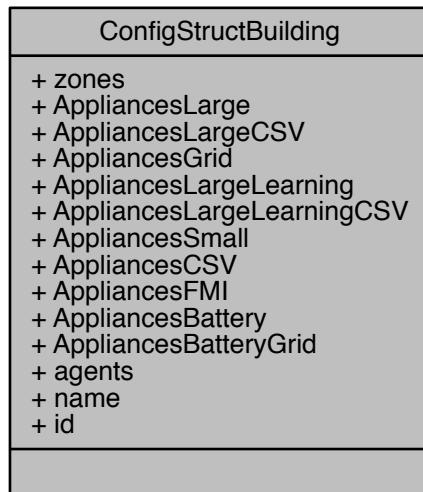
The documentation for this struct was generated from the following file:

- Configuration.hpp

15.20 ConfigStructBuilding Struct Reference

```
#include <Configuration.hpp>
```

Collaboration diagram for ConfigStructBuilding:



Public Attributes

- std::map< std::string, ConfigStructZone > zones
- std::vector< ConfigStructAppliance > AppliancesLarge
- std::vector< ConfigStructAppliance > AppliancesLargeCSV
- std::vector< ConfigStructAppliance > AppliancesGrid
- std::vector< ConfigStructAppliance > AppliancesLargeLearning
- std::vector< ConfigStructAppliance > AppliancesLargeLearningCSV
- std::vector< ConfigStructAppliance > AppliancesSmall
- std::vector< ConfigStructAppliance > AppliancesCSV
- std::vector< ConfigStructAppliance > AppliancesFMI
- std::vector< ConfigStructAppliance > AppliancesBattery
- std::vector< ConfigStructAppliance > AppliancesBatteryGrid
- std::vector< ConfigStructAgent > agents
- std::string name
- int id

15.20.1 Detailed Description

Definition at line 154 of file Configuration.hpp.

15.20.2 Member Data Documentation

15.20.2.1 agents

```
std::vector<ConfigStructAgent> ConfigStructBuilding::agents
```

Definition at line 166 of file Configuration.hpp.

15.20.2.2 AppliancesBattery

```
std::vector<ConfigStructAppliance> ConfigStructBuilding::AppliancesBattery
```

Definition at line 164 of file Configuration.hpp.

15.20.2.3 AppliancesBatteryGrid

```
std::vector<ConfigStructAppliance> ConfigStructBuilding::AppliancesBatteryGrid
```

Definition at line 165 of file Configuration.hpp.

15.20.2.4 AppliancesCSV

```
std::vector<ConfigStructAppliance> ConfigStructBuilding::AppliancesCSV
```

Definition at line 162 of file Configuration.hpp.

15.20.2.5 AppliancesFMI

```
std::vector<ConfigStructAppliance> ConfigStructBuilding::AppliancesFMI
```

Definition at line 163 of file Configuration.hpp.

15.20.2.6 AppliancesGrid

```
std::vector<ConfigStructAppliance> ConfigStructBuilding::AppliancesGrid
```

Definition at line 158 of file Configuration.hpp.

15.20.2.7 AppliancesLarge

```
std::vector<ConfigStructAppliance> ConfigStructBuilding::AppliancesLarge
```

Definition at line 156 of file Configuration.hpp.

15.20.2.8 AppliancesLargeCSV

```
std::vector<ConfigStructAppliance> ConfigStructBuilding::AppliancesLargeCSV
```

Definition at line 157 of file Configuration.hpp.

15.20.2.9 AppliancesLargeLearning

```
std::vector<ConfigStructAppliance> ConfigStructBuilding::AppliancesLargeLearning
```

Definition at line 159 of file Configuration.hpp.

15.20.2.10 AppliancesLargeLearningCSV

```
std::vector<ConfigStructAppliance> ConfigStructBuilding::AppliancesLargeLearningCSV
```

Definition at line 160 of file Configuration.hpp.

15.20.2.11 AppliancesSmall

```
std::vector<ConfigStructAppliance> ConfigStructBuilding::AppliancesSmall
```

Definition at line 161 of file Configuration.hpp.

15.20.2.12 id

```
int ConfigStructBuilding::id
```

Definition at line 168 of file Configuration.hpp.

15.20.2.13 name

```
std::string ConfigStructBuilding::name
```

Definition at line 167 of file Configuration.hpp.

15.20.2.14 zones

```
std::map<std::string, ConfigStructZone> ConfigStructBuilding::zones
```

Definition at line 155 of file Configuration.hpp.

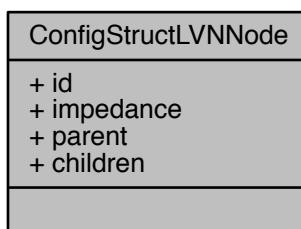
The documentation for this struct was generated from the following file:

- Configuration.hpp

15.21 ConfigStructLVNNNode Struct Reference

```
#include <Configuration.hpp>
```

Collaboration diagram for ConfigStructLVNNNode:



Public Attributes

- int `id`
- double `impedance` = 0.0047
- int `parent`
- std::vector< int > `children`

15.21.1 Detailed Description

Definition at line 13 of file Configuration.hpp.

15.21.2 Member Data Documentation

15.21.2.1 `children`

```
std::vector<int> ConfigStructLVNNNode::children
```

Definition at line 17 of file Configuration.hpp.

15.21.2.2 `id`

```
int ConfigStructLVNNNode::id
```

Definition at line 14 of file Configuration.hpp.

15.21.2.3 `impedance`

```
double ConfigStructLVNNNode::impedance = 0.0047
```

Definition at line 15 of file Configuration.hpp.

15.21.2.4 `parent`

```
int ConfigStructLVNNNode::parent
```

Definition at line 16 of file Configuration.hpp.

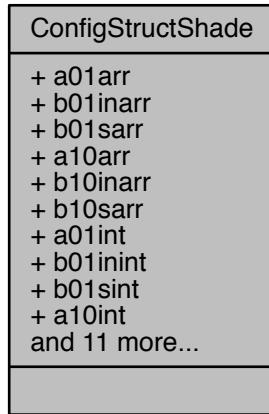
The documentation for this struct was generated from the following file:

- Configuration.hpp

15.22 ConfigStructShade Struct Reference

```
#include <Configuration.hpp>
```

Collaboration diagram for ConfigStructShade:



Public Attributes

- float [a01arr](#)
- float [b01inarr](#)
- float [b01sarr](#)
- float [a10arr](#)
- float [b10inarr](#)
- float [b10sarr](#)
- float [a01int](#)
- float [b01inint](#)
- float [b01sint](#)
- float [a10int](#)
- float [b10inint](#)
- float [b10sint](#)
- float [afullraise](#)
- float [boutfullraise](#)
- float [bsfullraise](#)
- float [bsfulllower](#)
- float [boutfulllower](#)
- float [afulllower](#)
- float [aSFlower](#)
- float [bSFlower](#)
- float [shapelower](#)

15.22.1 Detailed Description

Definition at line 62 of file Configuration.hpp.

15.22.2 Member Data Documentation

15.22.2.1 a01arr

```
float ConfigStructShade::a01arr
```

Definition at line 63 of file Configuration.hpp.

15.22.2.2 a01int

```
float ConfigStructShade::a01int
```

Definition at line 71 of file Configuration.hpp.

15.22.2.3 a10arr

```
float ConfigStructShade::a10arr
```

Definition at line 67 of file Configuration.hpp.

15.22.2.4 a10int

```
float ConfigStructShade::a10int
```

Definition at line 75 of file Configuration.hpp.

15.22.2.5 afulllower

```
float ConfigStructShade::afulllower
```

Definition at line 85 of file Configuration.hpp.

15.22.2.6 afullraise

```
float ConfigStructShade::afullraise
```

Definition at line 79 of file Configuration.hpp.

15.22.2.7 aSFlower

```
float ConfigStructShade::aSFlower
```

Definition at line 87 of file Configuration.hpp.

15.22.2.8 b01inarr

```
float ConfigStructShade::b01inarr
```

Definition at line 64 of file Configuration.hpp.

15.22.2.9 b01inint

```
float ConfigStructShade::b01inint
```

Definition at line 72 of file Configuration.hpp.

15.22.2.10 b01sarr

```
float ConfigStructShade::b01sarr
```

Definition at line 65 of file Configuration.hpp.

15.22.2.11 b01sint

```
float ConfigStructShade::b01sint
```

Definition at line 73 of file Configuration.hpp.

15.22.2.12 b10inarr

```
float ConfigStructShade::b10inarr
```

Definition at line 68 of file Configuration.hpp.

15.22.2.13 b10inint

```
float ConfigStructShade::b10inint
```

Definition at line 76 of file Configuration.hpp.

15.22.2.14 b10sarr

```
float ConfigStructShade::b10sarr
```

Definition at line 69 of file Configuration.hpp.

15.22.2.15 b10sint

```
float ConfigStructShade::b10sint
```

Definition at line 77 of file Configuration.hpp.

15.22.2.16 boutfulllower

```
float ConfigStructShade::boutfulllower
```

Definition at line 84 of file Configuration.hpp.

15.22.2.17 boutfullraise

```
float ConfigStructShade::boutfullraise
```

Definition at line 80 of file Configuration.hpp.

15.22.2.18 bSFlower

```
float ConfigStructShade::bSFlower
```

Definition at line 88 of file Configuration.hpp.

15.22.2.19 bsfulllower

```
float ConfigStructShade::bsfulllower
```

Definition at line 83 of file Configuration.hpp.

15.22.2.20 bsfullraise

```
float ConfigStructShade::bsfullraise
```

Definition at line 81 of file Configuration.hpp.

15.22.2.21 shapelow

```
float ConfigStructShade::shapelow
```

Definition at line 89 of file Configuration.hpp.

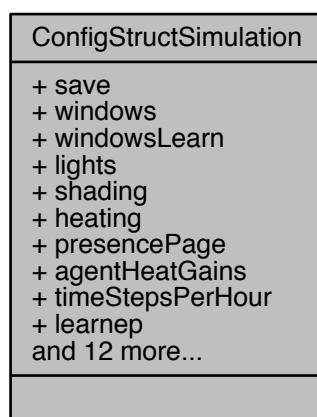
The documentation for this struct was generated from the following file:

- Configuration.hpp

15.23 ConfigStructSimulation Struct Reference

```
#include <Configuration.hpp>
```

Collaboration diagram for ConfigStructSimulation:



Public Attributes

- bool `save` = false
- bool `windows` = false
- bool `windowsLearn` = false
- bool `lights` = false
- bool `shading` = false
- bool `heating` = false
- bool `presencePage` = false
- bool `agentHeatGains` = true
- double `timeStepsPerHour`
- double `learnep`
- std::vector< double > `GridCost`
- int `learn`
- int `learnupdate`
- int `timeSteps`
- int `startDay`
- int `startMonth`
- int `startDayOfWeek` = 1
- int `endDay`
- int `endMonth`
- int `caseOrder`
- bool `ShadeClosedDuringNight`
- int `precision` = 5

15.23.1 Detailed Description

Definition at line 171 of file Configuration.hpp.

15.23.2 Member Data Documentation

15.23.2.1 `agentHeatGains`

```
bool ConfigStructSimulation::agentHeatGains = true
```

Definition at line 179 of file Configuration.hpp.

15.23.2.2 `caseOrder`

```
int ConfigStructSimulation::caseOrder
```

Definition at line 191 of file Configuration.hpp.

15.23.2.3 endDay

```
int ConfigStructSimulation::endDay
```

Definition at line 189 of file Configuration.hpp.

15.23.2.4 endMonth

```
int ConfigStructSimulation::endMonth
```

Definition at line 190 of file Configuration.hpp.

15.23.2.5 GridCost

```
std::vector<double> ConfigStructSimulation::GridCost
```

Definition at line 182 of file Configuration.hpp.

15.23.2.6 heating

```
bool ConfigStructSimulation::heating = false
```

Definition at line 177 of file Configuration.hpp.

15.23.2.7 learn

```
int ConfigStructSimulation::learn
```

Definition at line 183 of file Configuration.hpp.

15.23.2.8 learnep

```
double ConfigStructSimulation::learnep
```

Definition at line 181 of file Configuration.hpp.

15.23.2.9 learnupdate

```
int ConfigStructSimulation::learnupdate
```

Definition at line 184 of file Configuration.hpp.

15.23.2.10 lights

```
bool ConfigStructSimulation::lights = false
```

Definition at line 175 of file Configuration.hpp.

15.23.2.11 precision

```
int ConfigStructSimulation::precision = 5
```

Definition at line 193 of file Configuration.hpp.

15.23.2.12 presencePage

```
bool ConfigStructSimulation::presencePage = false
```

Definition at line 178 of file Configuration.hpp.

15.23.2.13 save

```
bool ConfigStructSimulation::save = false
```

Definition at line 172 of file Configuration.hpp.

15.23.2.14 ShadeClosedDuringNight

```
bool ConfigStructSimulation::ShadeClosedDuringNight
```

Definition at line 192 of file Configuration.hpp.

15.23.2.15 shading

```
bool ConfigStructSimulation::shading = false
```

Definition at line 176 of file Configuration.hpp.

15.23.2.16 startDay

```
int ConfigStructSimulation::startDay
```

Definition at line 186 of file Configuration.hpp.

15.23.2.17 startDayOfWeek

```
int ConfigStructSimulation::startDayOfWeek = 1
```

Definition at line 188 of file Configuration.hpp.

15.23.2.18 startMonth

```
int ConfigStructSimulation::startMonth
```

Definition at line 187 of file Configuration.hpp.

15.23.2.19 timeSteps

```
int ConfigStructSimulation::timeSteps
```

Definition at line 185 of file Configuration.hpp.

15.23.2.20 timeStepsPerHour

```
double ConfigStructSimulation::timeStepsPerHour
```

Definition at line 180 of file Configuration.hpp.

15.23.2.21 windows

```
bool ConfigStructSimulation::windows = false
```

Definition at line 173 of file Configuration.hpp.

15.23.2.22 windowsLearn

```
bool ConfigStructSimulation::windowsLearn = false
```

Definition at line 174 of file Configuration.hpp.

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

- Configuration.hpp

15.24 ConfigStructWindow Struct Reference

```
#include <Configuration.hpp>
```

Collaboration diagram for ConfigStructWindow:



Public Attributes

- double aop
- double bopout
- double shapeop
- double a01arr
- double b01inarr
- double b01outarr
- double b01absprevarr
- double b01rnarr
- double a01int
- double b01inint
- double b01outint
- double b01presint
- double b01rnint
- double a01dep
- double b01outdep
- double b01absdep
- double b01gddep
- double a10dep
- double b10indep
- double b10outdep
- double b10absdep
- double b10gddep

15.24.1 Detailed Description

Definition at line 92 of file Configuration.hpp.

15.24.2 Member Data Documentation

15.24.2.1 a01arr

```
double ConfigStructWindow::a01arr
```

Definition at line 97 of file Configuration.hpp.

15.24.2.2 a01dep

```
double ConfigStructWindow::a01dep
```

Definition at line 109 of file Configuration.hpp.

15.24.2.3 a01int

```
double ConfigStructWindow::a01int
```

Definition at line 103 of file Configuration.hpp.

15.24.2.4 a10dep

```
double ConfigStructWindow::a10dep
```

Definition at line 114 of file Configuration.hpp.

15.24.2.5 aop

```
double ConfigStructWindow::aop
```

Definition at line 93 of file Configuration.hpp.

15.24.2.6 b01absdep

```
double ConfigStructWindow::b01absdep
```

Definition at line 111 of file Configuration.hpp.

15.24.2.7 b01absprevarr

```
double ConfigStructWindow::b01absprevarr
```

Definition at line 100 of file Configuration.hpp.

15.24.2.8 b01gddep

```
double ConfigStructWindow::b01gddep
```

Definition at line 112 of file Configuration.hpp.

15.24.2.9 b01inarr

```
double ConfigStructWindow::b01inarr
```

Definition at line 98 of file Configuration.hpp.

15.24.2.10 b01inint

```
double ConfigStructWindow::b01inint
```

Definition at line 104 of file Configuration.hpp.

15.24.2.11 b01outarr

```
double ConfigStructWindow::b01outarr
```

Definition at line 99 of file Configuration.hpp.

15.24.2.12 b01outdep

```
double ConfigStructWindow::b01outdep
```

Definition at line 110 of file Configuration.hpp.

15.24.2.13 b01outint

```
double ConfigStructWindow::b01outint
```

Definition at line 105 of file Configuration.hpp.

15.24.2.14 b01presint

```
double ConfigStructWindow::b01presint
```

Definition at line 106 of file Configuration.hpp.

15.24.2.15 b01rnarr

```
double ConfigStructWindow::b01rnarr
```

Definition at line 101 of file Configuration.hpp.

15.24.2.16 b01rnint

```
double ConfigStructWindow::b01rnint
```

Definition at line 107 of file Configuration.hpp.

15.24.2.17 b10absdep

```
double ConfigStructWindow::b10absdep
```

Definition at line 117 of file Configuration.hpp.

15.24.2.18 b10gddep

```
double ConfigStructWindow::b10gddep
```

Definition at line 118 of file Configuration.hpp.

15.24.2.19 b10indep

```
double ConfigStructWindow::b10indep
```

Definition at line 115 of file Configuration.hpp.

15.24.2.20 b10outdep

```
double ConfigStructWindow::b10outdep
```

Definition at line 116 of file Configuration.hpp.

15.24.2.21 bopout

```
double ConfigStructWindow::bopout
```

Definition at line 94 of file Configuration.hpp.

15.24.2.22 shapeop

```
double ConfigStructWindow::shapeop
```

Definition at line 95 of file Configuration.hpp.

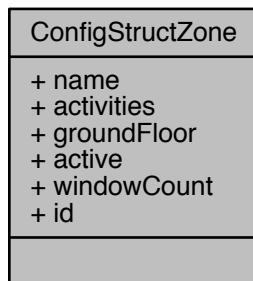
The documentation for this struct was generated from the following file:

- Configuration.hpp

15.25 ConfigStructZone Struct Reference

```
#include <Configuration.hpp>
```

Collaboration diagram for ConfigStructZone:



Public Attributes

- `std::string name`
- `std::vector< int > activities`
- `bool groundFloor = 0`
- `bool active = false`
- `int windowCount = 0`
- `int id = 0`

15.25.1 Detailed Description

Definition at line 20 of file Configuration.hpp.

15.25.2 Member Data Documentation

15.25.2.1 active

```
bool ConfigStructZone::active = false
```

Definition at line 24 of file Configuration.hpp.

15.25.2.2 activities

```
std::vector<int> ConfigStructZone::activities
```

Definition at line 22 of file Configuration.hpp.

15.25.2.3 groundFloor

```
bool ConfigStructZone::groundFloor = 0
```

Definition at line 23 of file Configuration.hpp.

15.25.2.4 id

```
int ConfigStructZone::id = 0
```

Definition at line 26 of file Configuration.hpp.

15.25.2.5 name

```
std::string ConfigStructZone::name
```

Definition at line 21 of file Configuration.hpp.

15.25.2.6 windowCount

```
int ConfigStructZone::windowCount = 0
```

Definition at line 25 of file Configuration.hpp.

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

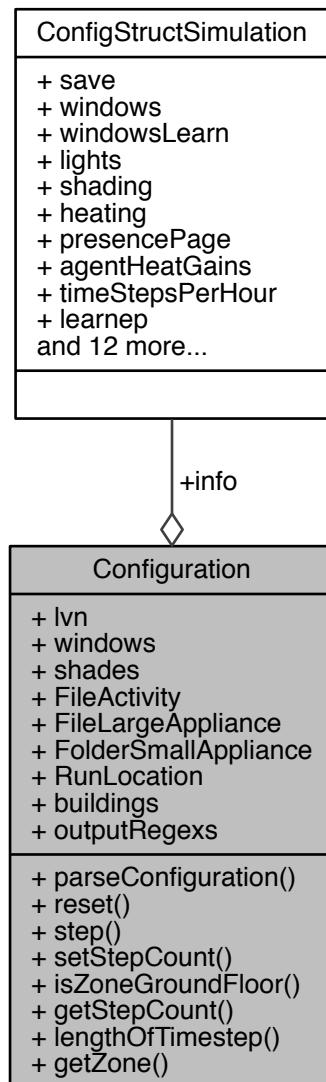
- Configuration.hpp

15.26 Configuration Class Reference

The simulation configuration reader.

```
#include <Configuration.hpp>
```

Collaboration diagram for Configuration:



Static Public Member Functions

- static void [parseConfiguration](#) (const std::string &filename)
Parses the simulation configuration file.
- static void [reset](#) ()
- static void [step](#) ()
- static void [setStepCount](#) (const int stepcount)
- static bool [isZoneGroundFloor](#) (std::string *zoneName)
- static int [getStepCount](#) ()
- static double [lengthOfTimestep](#) ()
- static [ConfigStructZone](#) [getZone](#) (std::string *zoneName)

Static Public Attributes

- static std::vector< [ConfigStructLVNNode](#) > [lvn](#)
- static std::map< int, [ConfigStructWindow](#) > [windows](#)
- static std::map< int, [ConfigStructShade](#) > [shades](#)
- static [ConfigStructSimulation](#) [info](#)
- static std::string [FileActivity](#)
- static std::string [FileLargeAppliance](#)
- static std::string [FolderSmallAppliance](#)
- static std::string [RunLocation](#)
- static std::vector< [ConfigStructBuilding](#) > [buildings](#)
- static std::vector< std::string > [outputRegexs](#)

15.26.1 Detailed Description

The simulation configuration reader.

reads the configuration files and stores the values for later retrieval

Definition at line 201 of file Configuration.hpp.

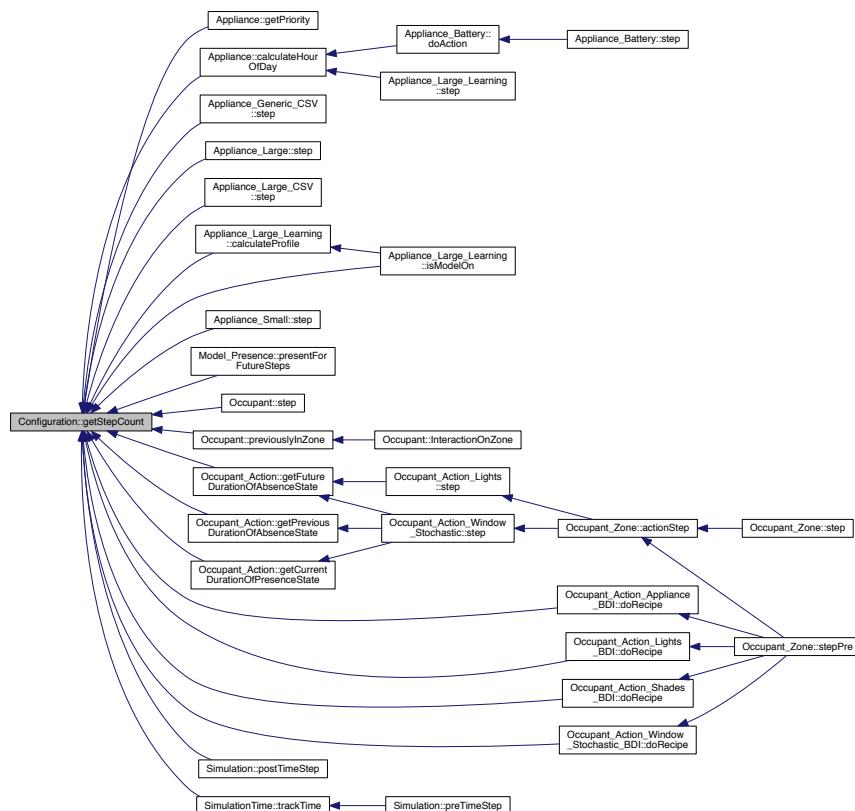
15.26.2 Member Function Documentation

15.26.2.1 getStepCount()

```
int Configuration::getStepCount ( ) [static]
```

Definition at line 614 of file Configuration.cpp.

Here is the caller graph for this function:



15.26.2.2 getZone()

```
ConfigStructZone Configuration::getZone (
    std::string * zoneName ) [static]
```

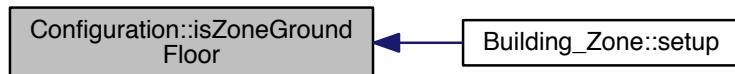
Definition at line 583 of file Configuration.cpp.

15.26.2.3 isZoneGroundFloor()

```
bool Configuration::isZoneGroundFloor (
    std::string * zoneName ) [static]
```

Definition at line 594 of file Configuration.cpp.

Here is the caller graph for this function:

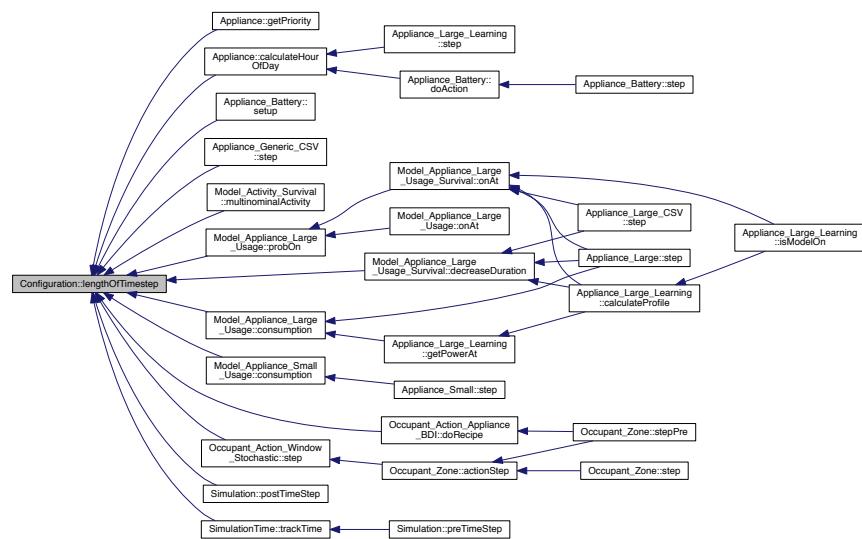


15.26.2.4 lengthOfTimestep()

```
double Configuration::lengthOfTimestep () [static]
```

Definition at line 606 of file Configuration.cpp.

Here is the caller graph for this function:



15.26.2.5 parseConfiguration()

```
void Configuration::parseConfiguration (
    const std::string & filename ) [static]
```

Parses the simulation configuration file.

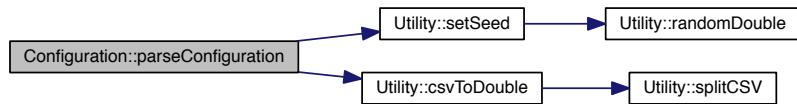
Parse the configuration file for the all the parameteres needed in the simulation

Parameters

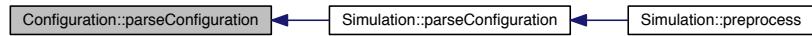
<code>filename</code>	location of the simulation file to parse.
-----------------------	---

Definition at line 525 of file Configuration.cpp.

Here is the call graph for this function:



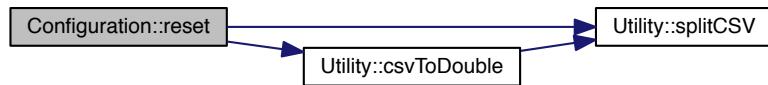
Here is the caller graph for this function:

**15.26.2.6 reset()**

```
void Configuration::reset ( ) [static]
```

Definition at line 35 of file Configuration.cpp.

Here is the call graph for this function:



15.26.2.7 setStepCount()

```
void Configuration::setStepCount ( const int stepcount ) [static]
```

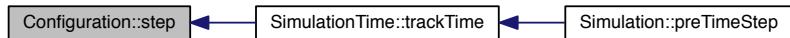
Definition at line 618 of file Configuration.cpp.

15.26.2.8 step()

```
void Configuration::step ( ) [static]
```

Definition at line 610 of file Configuration.cpp.

Here is the caller graph for this function:



15.26.3 Member Data Documentation

15.26.3.1 buildings

```
std::vector< ConfigStructBuilding > Configuration::buildings [static]
```

Definition at line 220 of file Configuration.hpp.

15.26.3.2 FileActivity

```
std::string Configuration::FileActivity [static]
```

Definition at line 216 of file Configuration.hpp.

15.26.3.3 FileLargeAppliance

```
std::string Configuration::FileLargeAppliance [static]
```

Definition at line 217 of file Configuration.hpp.

15.26.3.4 FolderSmallAppliance

```
std::string Configuration::FolderSmallAppliance [static]
```

Definition at line 218 of file Configuration.hpp.

15.26.3.5 info

```
ConfigStructSimulation Configuration::info [static]
```

Definition at line 215 of file Configuration.hpp.

15.26.3.6 lvn

```
std::vector< ConfigStructLVNNode > Configuration::lvn [static]
```

Definition at line 212 of file Configuration.hpp.

15.26.3.7 outputRegexs

```
std::vector< std::string > Configuration::outputRegexs [static]
```

Definition at line 221 of file Configuration.hpp.

15.26.3.8 RunLocation

```
std::string Configuration::RunLocation [static]
```

Definition at line 219 of file Configuration.hpp.

15.26.3.9 shades

```
std::map< int, ConfigStructShade > Configuration::shades [static]
```

Definition at line 214 of file Configuration.hpp.

15.26.3.10 windows

```
std::map< int, ConfigStructWindow > Configuration::windows [static]
```

Definition at line 213 of file Configuration.hpp.

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

- Configuration.hpp
- Configuration.cpp

15.27 Contract Struct Reference

A contract submitted from an appliance.

```
#include <Contract.hpp>
```

Collaboration diagram for Contract:

Contract
+ id + buildingID + requested + received + receivedCost + supplied + suppliedCost + suppliedLeft + priority

Public Attributes

- int **id**
- int **buildingID**
- double **requested** = 0.0
- double **received** = 0.0
- double **receivedCost** = 0.0
- double **supplied** = 0.0
- double **suppliedCost** = 0.0
- double **suppliedLeft** = 0.0
- double **priority** = 0.0

15.27.1 Detailed Description

A contract submitted from an appliance.

A contract that the appliances submit to the [Contract_Negotiation](#) class

Definition at line 12 of file [Contract.hpp](#).

15.27.2 Member Data Documentation

15.27.2.1 buildingID

```
int Contract::buildingID
```

Definition at line 14 of file [Contract.hpp](#).

15.27.2.2 id

```
int Contract::id
```

Definition at line 13 of file [Contract.hpp](#).

15.27.2.3 priority

```
double Contract::priority = 0.0
```

Definition at line 21 of file [Contract.hpp](#).

15.27.2.4 received

```
double Contract::received = 0.0
```

Definition at line 16 of file [Contract.hpp](#).

15.27.2.5 receivedCost

```
double Contract::receivedCost = 0.0
```

Definition at line 17 of file Contract.hpp.

15.27.2.6 requested

```
double Contract::requested = 0.0
```

Definition at line 15 of file Contract.hpp.

15.27.2.7 supplied

```
double Contract::supplied = 0.0
```

Definition at line 18 of file Contract.hpp.

15.27.2.8 suppliedCost

```
double Contract::suppliedCost = 0.0
```

Definition at line 19 of file Contract.hpp.

15.27.2.9 suppliedLeft

```
double Contract::suppliedLeft = 0.0
```

Definition at line 20 of file Contract.hpp.

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

- Contract.hpp

15.28 Contract_Negotiation Class Reference

Manages the negotiation between the appliance contracts.

```
#include <Contract_Negotiation.hpp>
```

Collaboration diagram for Contract_Negotiation:



Public Member Functions

- `Contract_Negotiation ()`
- void `submit (const Contract &c)`
- void `process ()`
- const `Contract getContract (const int buildingID, const int id) const`
- double `getReceivedPowerForContract (const int buildingID, const int id)`
- double `getCostOfPowerForContract (const int buildingID, const int id)`
- void `clear ()`
- double `getDifference () const`

15.28.1 Detailed Description

Manages the negotiation between the appliance contracts.

Manages the negotiation between the appliance contracts

Definition at line 18 of file Contract_Negotiation.hpp.

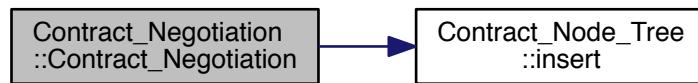
15.28.2 Constructor & Destructor Documentation

15.28.2.1 Contract_Negotiation()

```
Contract_Negotiation::Contract_Negotiation ( )
```

Definition at line 9 of file Contract_Negotiation.cpp.

Here is the call graph for this function:



15.28.3 Member Function Documentation

15.28.3.1 clear()

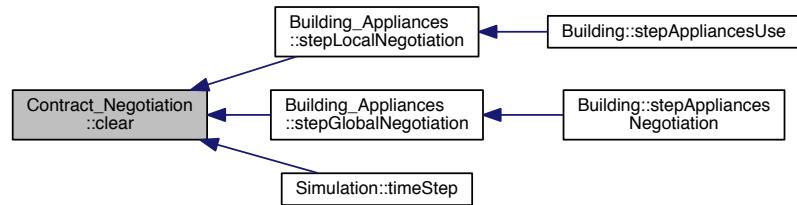
```
void Contract_Negotiation::clear ( )
```

Definition at line 104 of file Contract_Negotiation.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

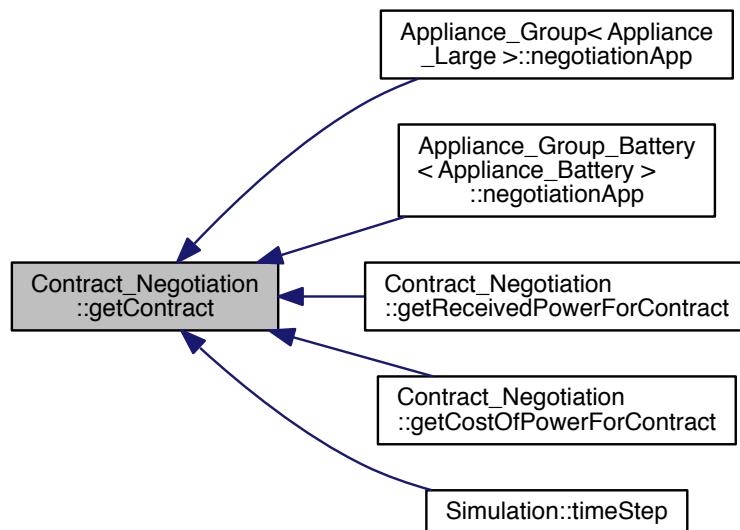


15.28.3.2 getContract()

```
const Contract Contract_Negotiation::getContract (
    const int buildingID,
    const int id ) const
```

Definition at line 89 of file Contract_Negotiation.cpp.

Here is the caller graph for this function:

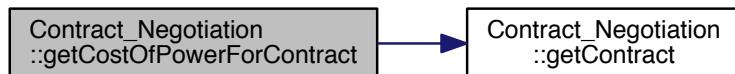


15.28.3.3 getCostOfPowerForContract()

```
double Contract_Negotiation::getCostOfPowerForContract (
    const int buildingID,
    const int id )
```

Definition at line 99 of file Contract_Negotiation.cpp.

Here is the call graph for this function:

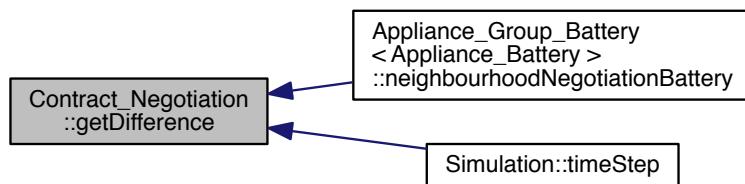


15.28.3.4 getDifference()

```
double Contract_Negotiation::getDifference ( ) const
```

Definition at line 37 of file Contract_Negotiation.cpp.

Here is the caller graph for this function:

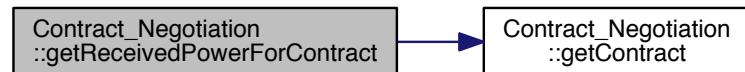


15.28.3.5 getReceivedPowerForContract()

```
double Contract_Negotiation::getReceivedPowerForContract (
    const int buildingID,
    const int id )
```

Definition at line 94 of file Contract_Negotiation.cpp.

Here is the call graph for this function:

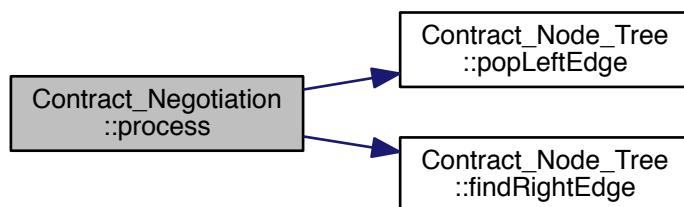


15.28.3.6 process()

```
void Contract_Negotiation::process ( )
```

Definition at line 41 of file Contract_Negotiation.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

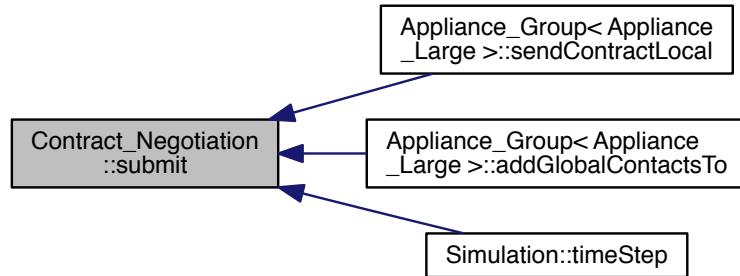


15.28.3.7 submit()

```
void Contract_Negotiation::submit ( const Contract & c )
```

Definition at line 29 of file Contract_Negotiation.cpp.

Here is the caller graph for this function:



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

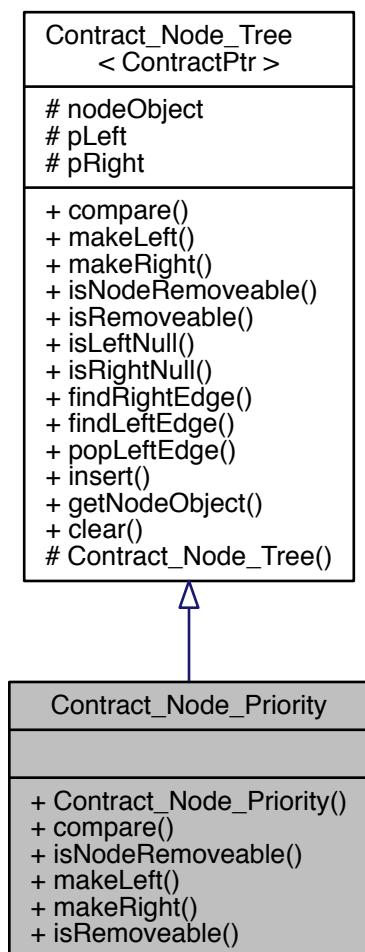
- `Contract_Negotiation.hpp`
- `Contract_Negotiation.cpp`

15.29 Contract_Node_Priority Class Reference

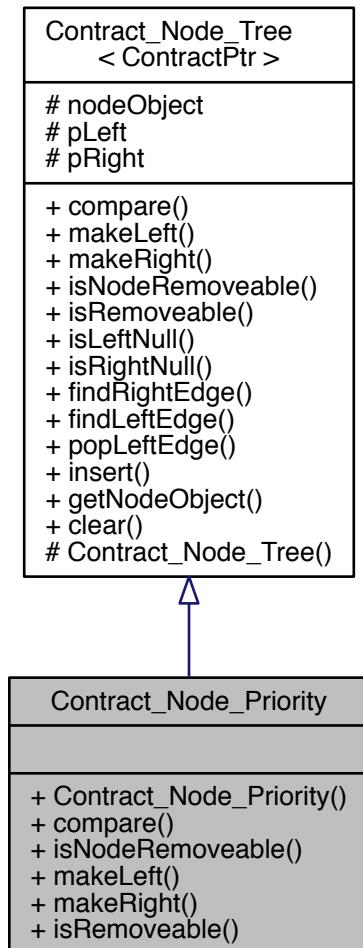
Tree of contracts sorted by the priority.

```
#include <Contract_Node_Priority.hpp>
```

Inheritance diagram for Contract_Node_Priority:



Collaboration diagram for Contract_Node_Priority:



Public Member Functions

- `Contract_Node_Priority ()`
- `bool compare (const ContractPtr &insert) const`
- `bool isNodeRemoveable (const std::shared_ptr< Contract_Node_Tree< ContractPtr >> &ptr) const`
- `void makeLeft ()`
- `void makeRight ()`
- `bool isRemoveable () const`

Additional Inherited Members

15.29.1 Detailed Description

Tree of contracts sorted by the priority.

Tree of contracts sorted by the priority of the contracts

Definition at line 14 of file `Contract_Node_Priority.hpp`.

15.29.2 Constructor & Destructor Documentation

15.29.2.1 Contract_Node_Priority()

```
Contract_Node_Priority::Contract_Node_Priority ( )
```

Definition at line 6 of file Contract_Node_Priority.cpp.

15.29.3 Member Function Documentation

15.29.3.1 compare()

```
bool Contract_Node_Priority::compare (
    const ContractPtr & insert ) const [virtual]
```

Implements [Contract_Node_Tree<ContractPtr>](#).

Definition at line 8 of file Contract_Node_Priority.cpp.

15.29.3.2 isNodeRemoveable()

```
bool Contract_Node_Priority::isNodeRemoveable (
    const std::shared_ptr< Contract\_Node\_Tree<ContractPtr> > & ptr ) const [virtual]
```

Implements [Contract_Node_Tree<ContractPtr>](#).

Definition at line 12 of file Contract_Node_Priority.cpp.

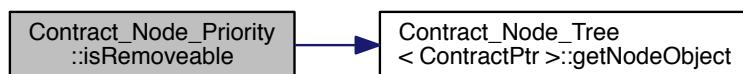
15.29.3.3 isRemoveable()

```
bool Contract_Node_Priority::isRemoveable ( ) const [virtual]
```

Implements [Contract_Node_Tree<ContractPtr>](#).

Definition at line 30 of file Contract_Node_Priority.cpp.

Here is the call graph for this function:



15.29.3.4 makeLeft()

```
void Contract_Node_Priority::makeLeft ( ) [virtual]
```

Implements [Contract_Node_Tree<ContractPtr>](#).

Definition at line 22 of file [Contract_Node_Priority.cpp](#).

15.29.3.5 makeRight()

```
void Contract_Node_Priority::makeRight ( ) [virtual]
```

Implements [Contract_Node_Tree<ContractPtr>](#).

Definition at line 26 of file [Contract_Node_Priority.cpp](#).

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

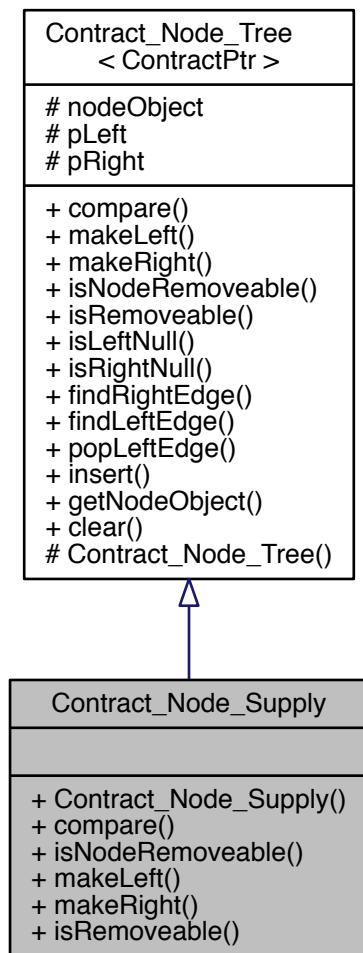
- [Contract_Node_Priority.hpp](#)
- [Contract_Node_Priority.cpp](#)

15.30 Contract_Node_Supply Class Reference

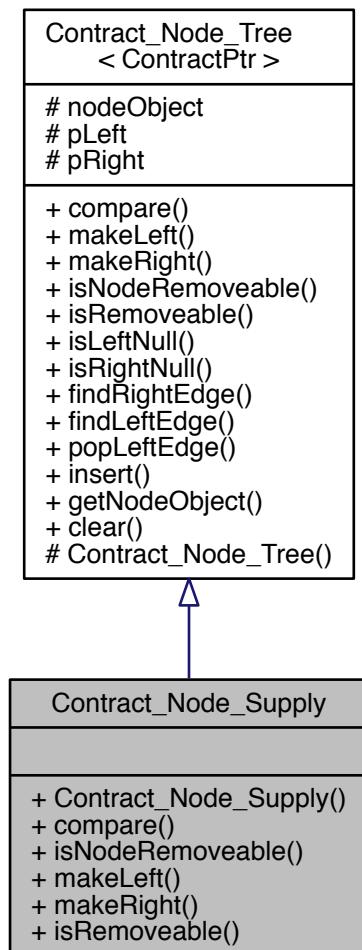
Tree of contracts sorted by the supplied energy left.

```
#include <Contract_Node_Supply.hpp>
```

Inheritance diagram for Contract_Node_Supply:



Collaboration diagram for Contract_Node_Supply:



Public Member Functions

- `Contract_Node_Supply ()`
- `bool compare (const ContractPtr &insert) const`
- `bool isNodeRemoveable (const std::shared_ptr< Contract_Node_Tree< ContractPtr >> &ptr) const`
- `void makeLeft ()`
- `void makeRight ()`
- `bool isRemoveable () const`

Additional Inherited Members

15.30.1 Detailed Description

Tree of contracts sorted by the supplied energy left.

Tree of contracts sorted by the supplied energy left in the contracts

Definition at line 14 of file `Contract_Node_Supply.hpp`.

15.30.2 Constructor & Destructor Documentation

15.30.2.1 Contract_Node_Supply()

```
Contract_Node_Supply::Contract_Node_Supply ( )
```

Definition at line 5 of file Contract_Node_Supply.cpp.

15.30.3 Member Function Documentation

15.30.3.1 compare()

```
bool Contract_Node_Supply::compare (
    const ContractPtr & insert ) const [virtual]
```

Implements [Contract_Node_Tree<ContractPtr>](#).

Definition at line 7 of file Contract_Node_Supply.cpp.

15.30.3.2 isNodeRemoveable()

```
bool Contract_Node_Supply::isNodeRemoveable (
    const std::shared_ptr< Contract\_Node\_Tree<ContractPtr> > & ptr ) const [virtual]
```

Implements [Contract_Node_Tree<ContractPtr>](#).

Definition at line 11 of file Contract_Node_Supply.cpp.

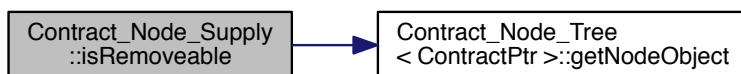
15.30.3.3 isRemoveable()

```
bool Contract_Node_Supply::isRemoveable ( ) const [virtual]
```

Implements [Contract_Node_Tree<ContractPtr>](#).

Definition at line 24 of file Contract_Node_Supply.cpp.

Here is the call graph for this function:



15.30.3.4 makeLeft()

```
void Contract_Node_Supply::makeLeft ( ) [virtual]
```

Implements [Contract_Node_Tree<ContractPtr>](#).

Definition at line 16 of file [Contract_Node_Supply.cpp](#).

15.30.3.5 makeRight()

```
void Contract_Node_Supply::makeRight ( ) [virtual]
```

Implements [Contract_Node_Tree<ContractPtr>](#).

Definition at line 20 of file [Contract_Node_Supply.cpp](#).

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

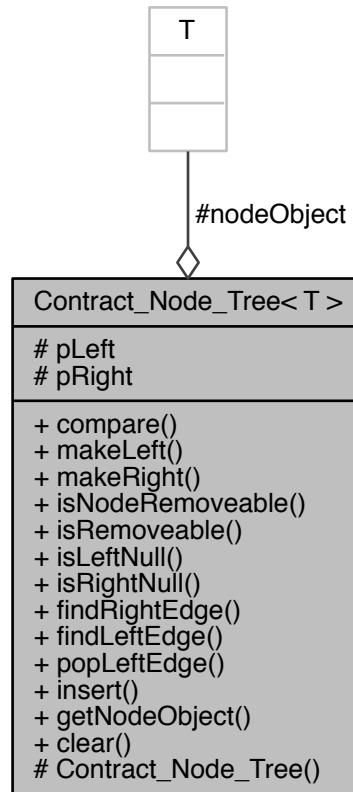
- [Contract_Node_Supply.hpp](#)
- [Contract_Node_Supply.cpp](#)

15.31 [Contract_Node_Tree< T >](#) Class Template Reference

Template class of the contract tree.

```
#include <Contract_Node_Tree.hpp>
```

Collaboration diagram for Contract_Node_Tree< T >:



Public Member Functions

- virtual bool `compare` (const T &`insert`) const =0
- virtual void `makeLeft` ()=0
- virtual void `makeRight` ()=0
- virtual bool `isNodeRemoveable` (const std::shared_ptr< `Contract_Node_Tree`< T >> &`ptr`) const =0
- virtual bool `isRemoveable` () const =0
- bool `isLeftNull` ()
- bool `isRightNull` ()
- T `findRightEdge` ()
- T `findLeftEdge` ()
- T `popLeftEdge` ()
- void `insert` (const T &`insert`, double `value`)
- const T `getNodeObject` () const
- void `clear` ()

Protected Member Functions

- `Contract_Node_Tree` ()

Protected Attributes

- T `nodeObject`
- std::shared_ptr< Contract_Node_Tree< T > > `pLeft`
- std::shared_ptr< Contract_Node_Tree< T > > `pRight`

15.31.1 Detailed Description

```
template<class T>
class Contract_Node_Tree< T >
```

Template class of the contract tree.

Template class of the contract tree

Definition at line 15 of file Contract_Node_Tree.hpp.

15.31.2 Constructor & Destructor Documentation

15.31.2.1 Contract_Node_Tree()

```
template<class T>
Contract_Node_Tree< T >::Contract_Node_Tree ( ) [inline], [protected]
```

Definition at line 135 of file Contract_Node_Tree.hpp.

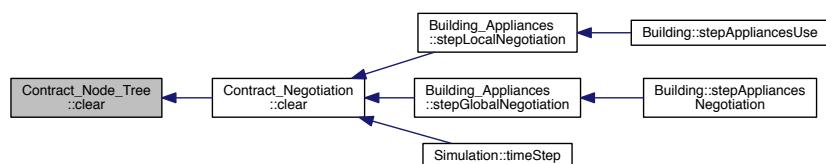
15.31.3 Member Function Documentation

15.31.3.1 clear()

```
template<class T>
void Contract_Node_Tree< T >::clear ( ) [inline]
```

Definition at line 126 of file Contract_Node_Tree.hpp.

Here is the caller graph for this function:



15.31.3.2 compare()

```
template<class T>
virtual bool Contract_Node_Tree< T >::compare (
    const T & insert ) const [pure virtual]
```

Implemented in [Contract_Node_Priority](#), and [Contract_Node_Supply](#).

15.31.3.3 findLeftEdge()

```
template<class T>
T Contract_Node_Tree< T >::findLeftEdge ( ) [inline]
```

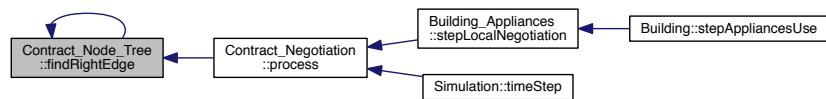
Definition at line 58 of file [Contract_Node_Tree.hpp](#).

15.31.3.4 findRightEdge()

```
template<class T>
T Contract_Node_Tree< T >::findRightEdge ( ) [inline]
```

Definition at line 32 of file [Contract_Node_Tree.hpp](#).

Here is the caller graph for this function:



15.31.3.5 getNodeObject()

```
template<class T>
const T Contract_Node_Tree< T >::getNodeObject ( ) const [inline]
```

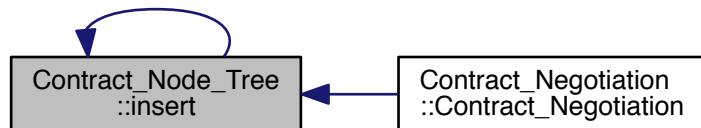
Definition at line 122 of file [Contract_Node_Tree.hpp](#).

15.31.3.6 insert()

```
template<class T>
void Contract_Node_Tree< T >::insert (
    const T & insert,
    double value ) [inline]
```

Definition at line 102 of file Contract_Node_Tree.hpp.

Here is the caller graph for this function:



15.31.3.7 isLeftNull()

```
template<class T>
bool Contract_Node_Tree< T >::isLeftNull () [inline]
```

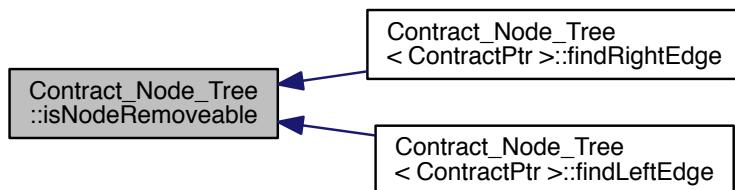
Definition at line 24 of file Contract_Node_Tree.hpp.

15.31.3.8 isNodeRemoveable()

```
template<class T>
virtual bool Contract_Node_Tree< T >::isNodeRemoveable (
    const std::shared_ptr< Contract_Node_Tree< T >> & ptr ) const [pure virtual]
```

Implemented in [Contract_Node_Priority](#), and [Contract_Node_Supply](#).

Here is the caller graph for this function:

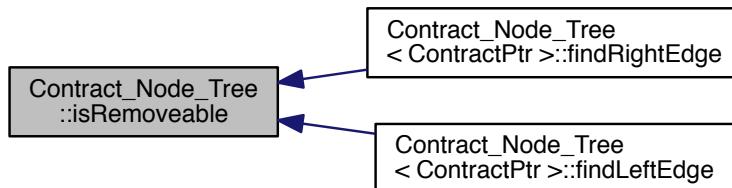


15.31.3.9 isRemoveable()

```
template<class T>
virtual bool Contract_Node_Tree< T >::isRemoveable ( ) const [pure virtual]
```

Implemented in [Contract_Node_Priority](#), and [Contract_Node_Supply](#).

Here is the caller graph for this function:



15.31.3.10 isRightNull()

```
template<class T>
bool Contract_Node_Tree< T >::isRightNull ( ) [inline]
```

Definition at line 28 of file [Contract_Node_Tree.hpp](#).

15.31.3.11 makeLeft()

```
template<class T>
virtual void Contract_Node_Tree< T >::makeLeft ( ) [pure virtual]
```

Implemented in [Contract_Node_Priority](#), and [Contract_Node_Supply](#).

Here is the caller graph for this function:



15.31.3.12 makeRight()

```
template<class T>
virtual void Contract\_Node\_Tree< T >::makeRight ( ) [pure virtual]
```

Implemented in [Contract_Node_Priority](#), and [Contract_Node_Supply](#).

Here is the caller graph for this function:

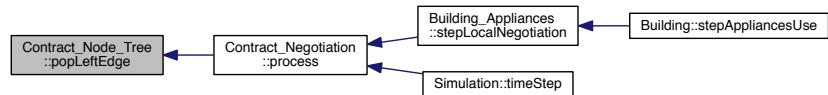


15.31.3.13 popLeftEdge()

```
template<class T>
T Contract\_Node\_Tree< T >::popLeftEdge ( ) [inline]
```

Definition at line 80 of file [Contract_Node_Tree.hpp](#).

Here is the caller graph for this function:



15.31.4 Member Data Documentation

15.31.4.1 nodeObject

```
template<class T>
T Contract\_Node\_Tree< T >::nodeObject [protected]
```

Definition at line 134 of file [Contract_Node_Tree.hpp](#).

15.31.4.2 pLeft

```
template<class T>
std::shared_ptr<Contract_Node_Tree<T>> Contract_Node_Tree< T >::pLeft [protected]
```

Definition at line 136 of file Contract_Node_Tree.hpp.

15.31.4.3 pRight

```
template<class T>
std::shared_ptr<Contract_Node_Tree<T>> Contract_Node_Tree< T >::pRight [protected]
```

Definition at line 137 of file Contract_Node_Tree.hpp.

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

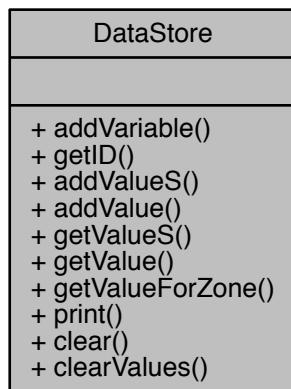
- Contract_Node_Tree.hpp

15.32 DataStore Class Reference

Stores all the data at each timestep.

```
#include <DataStore.hpp>
```

Collaboration diagram for DataStore:



Static Public Member Functions

- static int [addVariable](#) (const std::string &name)
- static int [getID](#) (const std::string &name)
- static void [addValueS](#) (const std::string &name, const float val)
- static void [addValue](#) (const int &id, const float val)
- static float [getValueS](#) (const std::string &name)
- static float [getValue](#) (const int &id)
- static float [getValueForZone](#) (const std::string &name, const std::string &zoneName)
- static void [print](#) ()
- static void [clear](#) ()
- static void [clearValues](#) ()

15.32.1 Detailed Description

Stores all the data at each timestep.

Stores all the data at each timestep to be written to a file later in the process

Definition at line 15 of file DataStore.hpp.

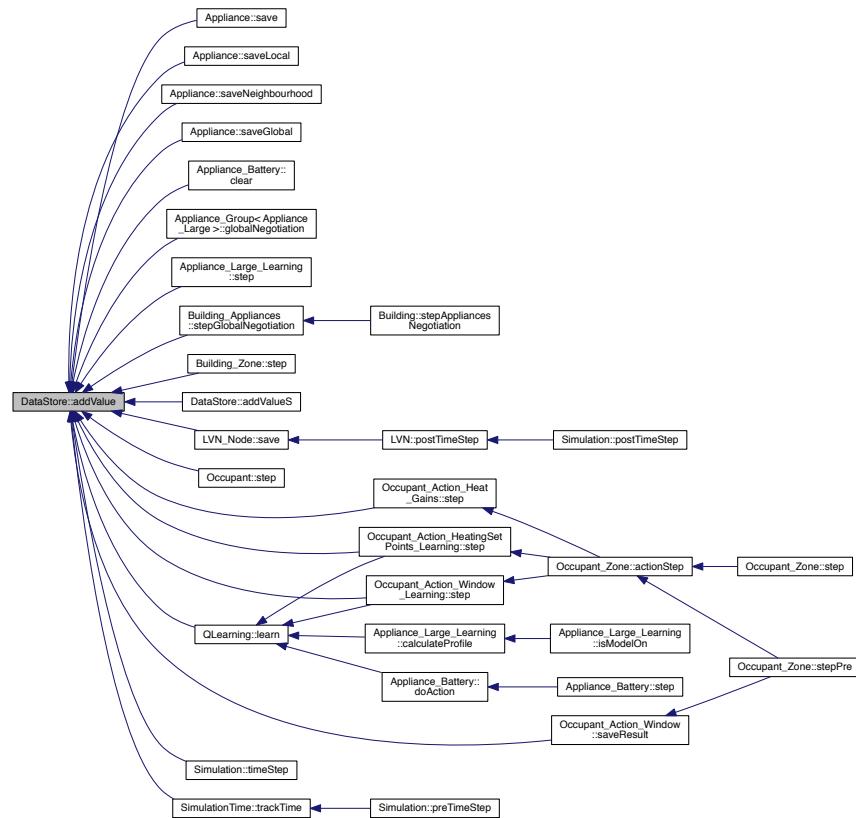
15.32.2 Member Function Documentation

15.32.2.1 `addValue()`

```
void DataStore::addValue (
    const int & id,
    const float val ) [static]
```

Definition at line 60 of file DataStore.cpp.

Here is the caller graph for this function:



15.32.2.2 addValueS()

```

void DataStore::addValueS (
    const std::string & name,
    const float val ) [static]

```

Definition at line 50 of file DataStore.cpp.

Here is the call graph for this function:

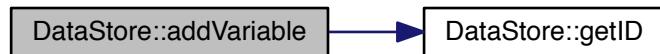


15.32.2.3 addVariable()

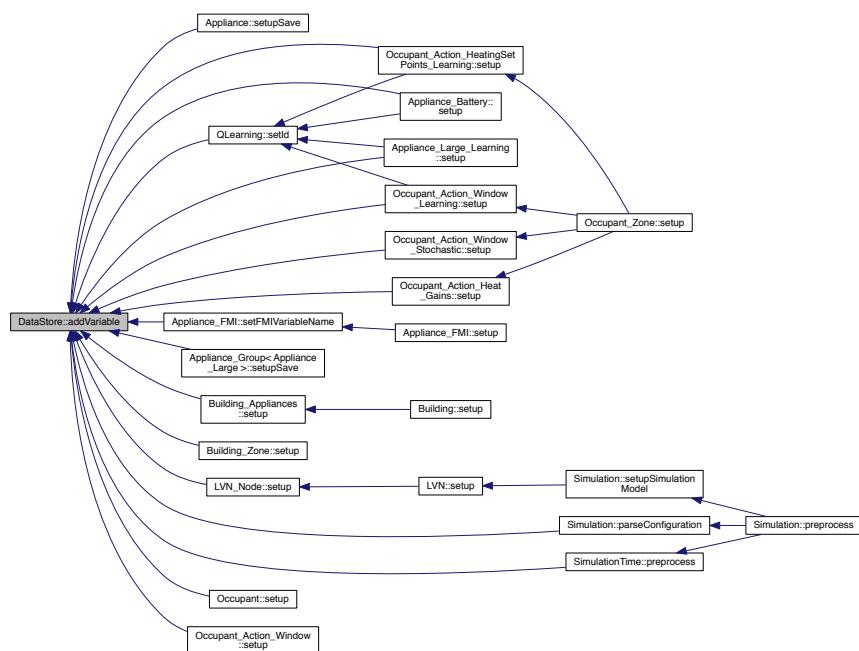
```
int DataStore::addVariable (
    const std::string & name ) [static]
```

Definition at line 26 of file DataStore.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.32.2.4 clear()

```
void DataStore::clear ( ) [static]
```

Definition at line 106 of file DataStore.cpp.

15.32.2.5 clearValues()

```
void DataStore::clearValues ( ) [static]
```

Definition at line 98 of file DataStore.cpp.

Here is the caller graph for this function:

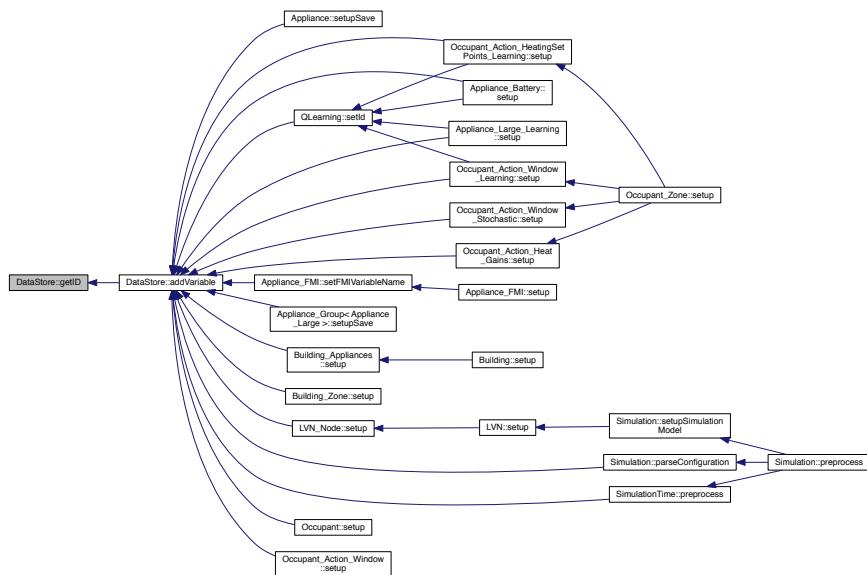


15.32.2.6 getID()

```
int DataStore::getID (
    const std::string & name ) [static]
```

Definition at line 21 of file DataStore.cpp.

Here is the caller graph for this function:

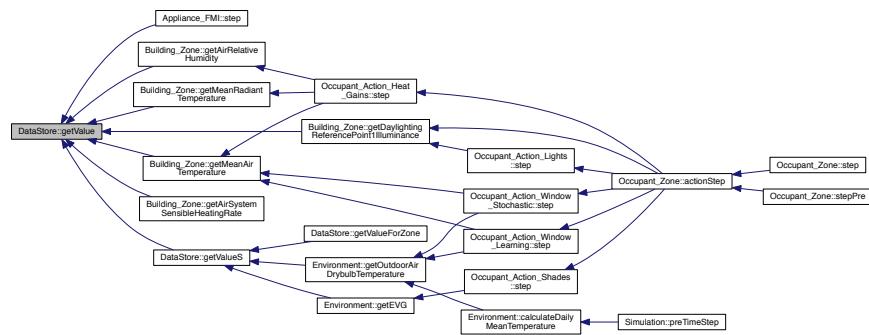


15.32.2.7 getValue()

```
float DataStore::getValue (
    const int & id ) [static]
```

Definition at line 90 of file DataStore.cpp.

Here is the caller graph for this function:



15.32.2.8 getValueForZone()

```
float DataStore::getValueForZone (
    const std::string & name,
    const std::string & zoneName ) [static]
```

Definition at line 67 of file DataStore.cpp.

Here is the call graph for this function:



15.32.2.9 `getValueS()`

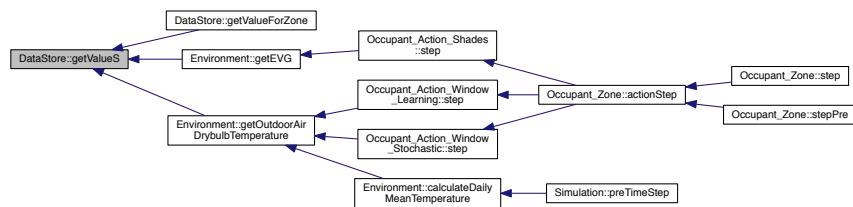
```
float DataStore::getValueS (
    const std::string & name ) [static]
```

Definition at line 72 of file DataStore.cpp.

Here is the call graph for this function:



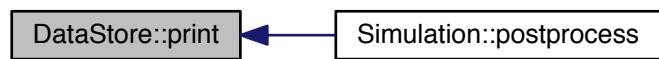
Here is the caller graph for this function:

15.32.2.10 `print()`

```
void DataStore::print ( ) [static]
```

Definition at line 113 of file DataStore.cpp.

Here is the caller graph for this function:



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

- DataStore.hpp
- DataStore.cpp

15.33 Environment Class Reference

Stores environmental parameters.

```
#include <Environment.hpp>
```

Collaboration diagram for Environment:

Environment
+ dailyMeanTemperature
+ calculateDailyMeanTemperature() + getDailyMeanTemperature() + getEVG() + getOutdoorAirDrybulbTemperature()

Static Public Member Functions

- static void [calculateDailyMeanTemperature \(\)](#)
- static double [getDailyMeanTemperature \(\)](#)
- static double [getEVG \(\)](#)
- static double [getOutdoorAirDrybulbTemperature \(\)](#)

Static Public Attributes

- static double [dailyMeanTemperature = 0](#)

15.33.1 Detailed Description

Stores environmental parameters.

Stores environmental parameters such as extrenal weather conditions

Definition at line 13 of file Environment.hpp.

15.33.2 Member Function Documentation

15.33.2.1 calculateDailyMeanTemperature()

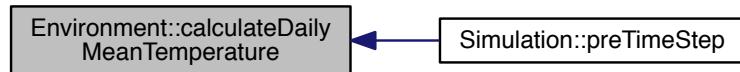
```
void Environment::calculateDailyMeanTemperature ( ) [static]
```

Definition at line 17 of file Environment.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

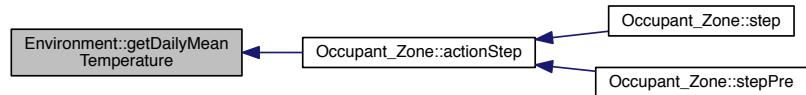


15.33.2.2 getDailyMeanTemperature()

```
double Environment::getDailyMeanTemperature ( ) [static]
```

Definition at line 13 of file Environment.cpp.

Here is the caller graph for this function:

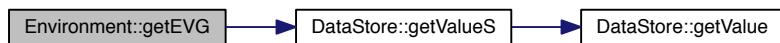


15.33.2.3 getEVG()

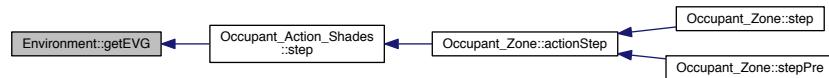
```
double Environment::getEVG ( ) [static]
```

Definition at line 36 of file Environment.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.33.2.4 getOutdoorAirDrybulbTemperature()

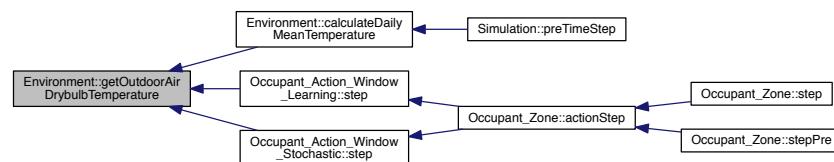
```
double Environment::getOutdoorAirDrybulbTemperature ( ) [static]
```

Definition at line 41 of file Environment.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.33.3 Member Data Documentation

15.33.3.1 dailyMeanTemperature

```
double Environment::dailyMeanTemperature = 0 [static]
```

Definition at line 15 of file Environment.hpp.

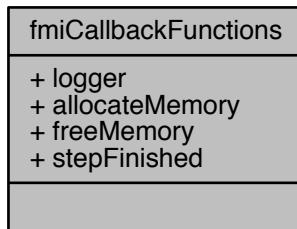
The documentation for this class was generated from the following files:

- Environment.hpp
- Environment.cpp

15.34 fmiCallbackFunctions Struct Reference

```
#include <fmiFunctions.hpp>
```

Collaboration diagram for fmiCallbackFunctions:



Public Attributes

- fmiCallbackLogger [logger](#)
- fmiCallbackAllocateMemory [allocateMemory](#)
- fmiCallbackFreeMemory [freeMemory](#)
- fmiStepFinished [stepFinished](#)

15.34.1 Detailed Description

Definition at line 154 of file fmiFunctions.hpp.

15.34.2 Member Data Documentation

15.34.2.1 allocateMemory

```
fmiCallbackAllocateMemory fmiCallbackFunctions::allocateMemory
```

Definition at line 156 of file fmiFunctions.hpp.

15.34.2.2 freeMemory

```
fmiCallbackFreeMemory fmiCallbackFunctions::freeMemory
```

Definition at line 157 of file fmiFunctions.hpp.

15.34.2.3 logger

```
fmiCallbackLogger fmiCallbackFunctions::logger
```

Definition at line 155 of file fmiFunctions.hpp.

15.34.2.4 stepFinished

```
fmiStepFinished fmiCallbackFunctions::stepFinished
```

Definition at line 158 of file fmiFunctions.hpp.

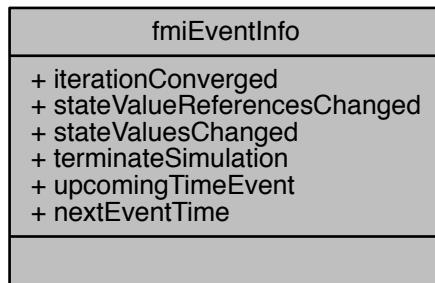
The documentation for this struct was generated from the following file:

- fmiFunctions.hpp

15.35 fmiEventInfo Struct Reference

```
#include <fmiFunctions.hpp>
```

Collaboration diagram for fmiEventInfo:



Public Attributes

- fmiBoolean [iterationConverged](#)
- fmiBoolean [stateValueReferencesChanged](#)
- fmiBoolean [stateValuesChanged](#)
- fmiBoolean [terminateSimulation](#)
- fmiBoolean [upcomingTimeEvent](#)
- fmiReal [nextEventTime](#)

15.35.1 Detailed Description

Definition at line 161 of file fmiFunctions.hpp.

15.35.2 Member Data Documentation

15.35.2.1 iterationConverged

```
fmiBoolean fmiEventInfo::iterationConverged
```

Definition at line 162 of file fmiFunctions.hpp.

15.35.2.2 nextEventTime

```
fmiReal fmiEventInfo::nextEventTime
```

Definition at line 167 of file fmiFunctions.hpp.

15.35.2.3 stateValueReferencesChanged

```
fmiBoolean fmiEventInfo::stateValueReferencesChanged
```

Definition at line 163 of file fmiFunctions.hpp.

15.35.2.4 stateValuesChanged

```
fmiBoolean fmiEventInfo::stateValuesChanged
```

Definition at line 164 of file fmiFunctions.hpp.

15.35.2.5 terminateSimulation

```
fmiBoolean fmiEventInfo::terminateSimulation
```

Definition at line 165 of file fmiFunctions.hpp.

15.35.2.6 upcomingTimeEvent

```
fmiBoolean fmiEventInfo::upcomingTimeEvent
```

Definition at line 166 of file fmiFunctions.hpp.

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

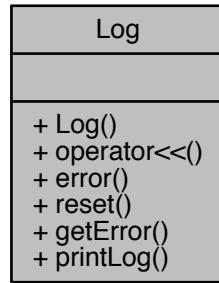
- fmiFunctions.hpp

15.36 Log Class Reference

Logs the error messages.

```
#include <Log.hpp>
```

Collaboration diagram for Log:



Public Member Functions

- [Log \(\)](#)
- template<typename T >
[Log & operator<< \(const T &t\)](#)
- [void error \(\)](#)
- [void reset \(\)](#)
- [bool getError \(\)](#)

Static Public Member Functions

- [static void printLog \(\)](#)

15.36.1 Detailed Description

Logs the error messages.

Logs the error messages for later writing to file

Definition at line 16 of file Log.hpp.

15.36.2 Constructor & Destructor Documentation

15.36.2.1 Log()

```
Log::Log ( )
```

Definition at line 11 of file Log.cpp.

15.36.3 Member Function Documentation

15.36.3.1 error()

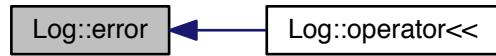
```
void Log::error ( )
```

Definition at line 35 of file Log.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.36.3.2 getError()

```
bool Log::getError ( )
```

Definition at line 44 of file Log.cpp.

Here is the caller graph for this function:

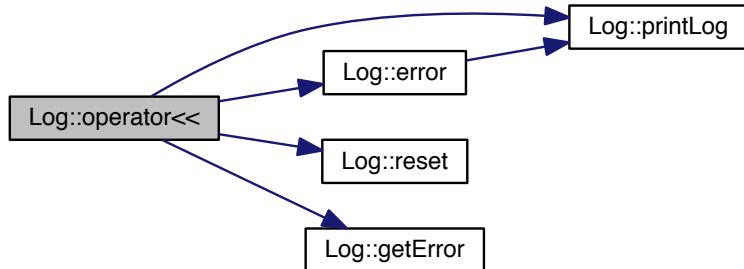


15.36.3.3 operator<<()

```
template<typename T >
Log& Log::operator<< (
    const T & t ) [inline]
```

Definition at line 21 of file Log.hpp.

Here is the call graph for this function:

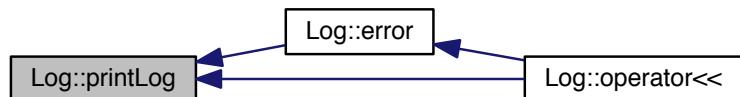


15.36.3.4 printLog()

```
void Log::printLog ( ) [static]
```

Definition at line 13 of file Log.cpp.

Here is the caller graph for this function:



15.36.3.5 reset()

```
void Log::reset ( )
```

Definition at line 40 of file Log.cpp.

Here is the caller graph for this function:



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

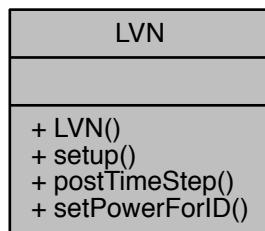
- Log.hpp
- Log.cpp

15.37 LVN Class Reference

Models the survival function for the activity model.

```
#include <LVN.hpp>
```

Collaboration diagram for LVN:



Public Member Functions

- [LVN \(\)](#)
- [void setup \(\)](#)
- [void postTimeStep \(\)](#)
- [void setPowerForID \(const double power, const int id\)](#)

15.37.1 Detailed Description

Models the survival function for the activity model.

The low voltage network used to calculate cable losses

Sancho-Tomás, A., Chapman, J., & Robinson, D. (2017). Extending No-MASS: Multi-Agent Stochastic Simulation for Demand Response of residential appliances. In Building Simulation 2017.

Definition at line 15 of file LVN.hpp.

15.37.2 Constructor & Destructor Documentation

15.37.2.1 LVN()

```
LVN::LVN ( )
```

Definition at line 7 of file LVN.cpp.

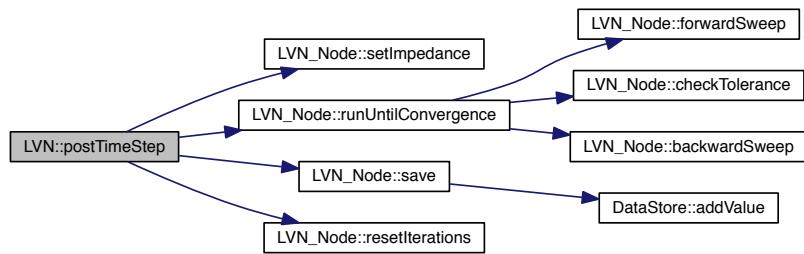
15.37.3 Member Function Documentation

15.37.3.1 postTimeStep()

```
void LVN::postTimeStep( )
```

Definition at line 25 of file LVN.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

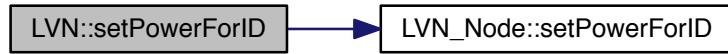


15.37.3.2 setPowerForID()

```
void LVN::setPowerForID(
    const double power,
    const int id )
```

Definition at line 35 of file LVN.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

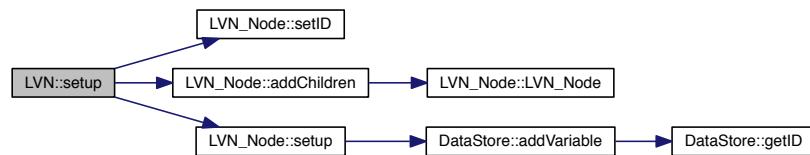


15.37.3.3 setup()

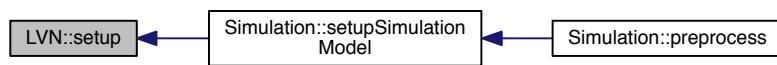
```
void LVN::setup ( )
```

Definition at line 11 of file LVN.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



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

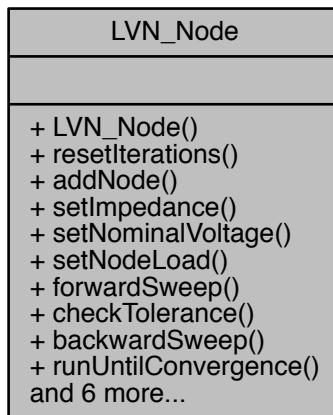
- LVN.hpp
- LVN.cpp

15.38 LVN_Node Class Reference

Low voltage network for power flow analysis.

```
#include <LVN_Node.hpp>
```

Collaboration diagram for LVN_Node:



Public Member Functions

- [LVN_Node \(\)](#)
- void [resetIterations \(\)](#)

For each time step that we want to achieve convergence, we need to reset the iteration counter. The iteration counter controls when the terminal nodes are assumed to have nominal voltage.
- void [addNode \(const LVN_Node &node\)](#)

Add node to object node, to create a network.
- void [setImpedance \(const std::complex< double > &value\)](#)

Set value of the cable impedance between self node and the parent node.
- void [setNominalVoltage \(const std::complex< double > &value\)](#)

Set value of the nominal voltage.
- void [setNodeLoad \(const std::complex< double > &nodeLoad\)](#)

Assign the corresponding NodeLoad to the node, such as HouseLoad, ApplianceLoad, etc.
- void [forwardSweep \(\)](#)

On the forward sweep currents of the network are updated.
- double [checkTolerance \(\)](#)

Evaluates error between voltage of rootNode (closest one to source) and nominal voltage.
- void [backwardSweep \(const std::complex< double > &parent_voltage\)](#)

On the backward sweep, voltages of the network are updated.
- void [runUntilConvergence \(double tolerance\)](#)

runUntilConvergence should only be run on the rootNode.
- bool [setPowerForID \(const std::complex< double > &power, const int id\)](#)

set the power for node at the ID.

- void `setID` (const int id)
- void `setup` ()
- int `getID` () const
- void `addChildren` (const std::vector< int > &ids)
- void `save` ()

15.38.1 Detailed Description

Low voltage network for power flow analysis.

Script copied from original Node class in Documents > LVNmodelling > powerFlowAnalysis. Details on how to use the class for power flow analysis in there. The class Node has been updated to take NodeLoad objects for definition of Complex Power. NodeLoad objects inherit to HouseLoad, ApplianceLoad, ElectricVehicleLoad, PVLoad or any other, so that Node class only needs to deal with NodeLoad objects.

Converted from Python by Jacob Chapman

Created by Ana Sancho on 12 Apr 2016.

Definition at line 20 of file LVN_Node.hpp.

15.38.2 Constructor & Destructor Documentation

15.38.2.1 LVN_Node()

```
LVN_Node::LVN_Node ( )
```

Definition at line 9 of file LVN_Node.cpp.

Here is the caller graph for this function:



15.38.3 Member Function Documentation

15.38.3.1 addChildren()

```
void LVN_Node::addChildren (
    const std::vector< int > & ids )
```

Definition at line 212 of file LVN_Node.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.38.3.2 addNode()

```
void LVN_Node::addNode (
    const LVN_Node & node )
```

Add node to object node, to create a network.

Parameters

<i>node</i>	node to be added.
-------------	-------------------

Definition at line 47 of file LVN_Node.cpp.

15.38.3.3 backwardSweep()

```
void LVN_Node::backwardSweep (
    const std::complex< double > & parent_voltage )
```

On the backward sweep, voltages of the network are updated.

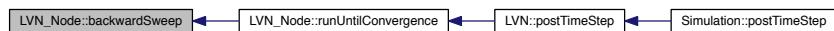
Parameters

<i>parent_voltage</i>	As input needs to be the voltage of the root node (set to nominal voltage before starting backward sweep. After, recursively corresponds to $V[j-1]$).
-----------------------	---

$$V[j] = V[j-1] - l_{\text{line}}[j]*z[j] \text{ unless you are root, and then: } V[j-1] = V_{\text{nominal}}$$

Definition at line 144 of file LVN_Node.cpp.

Here is the caller graph for this function:

**15.38.3.4 checkTolerance()**

```
double LVN_Node::checkTolerance ( )
```

Evaluates error between voltage of rootNode (closest one to source) and nominal voltage.

Returns

error

What we need to calculate is:

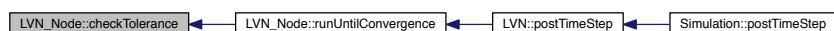
error = nominalVoltage - Vslack (see figure attached in report)

error = abs(nominalVoltage - voltage) // this is wrong,

since the voltage drop between Vroot and Vslack is not being considered.

Definition at line 130 of file LVN_Node.cpp.

Here is the caller graph for this function:



15.38.3.5 forwardSweep()

```
void LVN_Node::forwardSweep ( )
```

On the forward sweep currents of the network are updated.

$$V[j] = V[j+1] + Iline[j+1]*Z[j+1]$$

with only taking one branch ($j+1$) from all the k branches attached to $j-1$.

$$Iload[j] = (S[j]/V[j])* // is complex conjugate$$

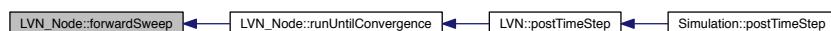
$Iline[j] = Iload[j] + \sum_k Iline[k]*Z[k]$, for k being all child branches coming from node j .

(All currents arriving to j)

If node is a terminal node, then voltage must be assumed (on the first iteration) to be the nominal voltage.

Definition at line 93 of file LVN_Node.cpp.

Here is the caller graph for this function:



15.38.3.6 getID()

```
int LVN_Node::getID ( ) const
```

Definition at line 231 of file LVN_Node.cpp.

15.38.3.7 resetIterations()

```
void LVN_Node::resetIterations ( )
```

For each time step that we want to achieve convergence, we need to reset the iteration counter. The iteration counter controls when the terminal nodes are assumed to have nominal voltage.

Definition at line 39 of file LVN_Node.cpp.

Here is the caller graph for this function:



15.38.3.8 runUntilConvergence()

```
void LVN_Node::runUntilConvergence (
    double tolerance )
```

runUntilConvergence should only be run on the rootNode.

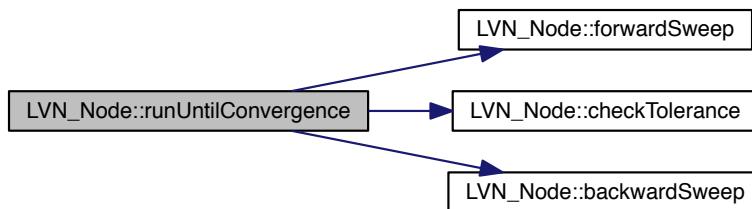
Parameters

<i>tolerance</i>	tolerance to be admissible for the error.
------------------	---

Runs a forward sweep. Calculates error.
 If error is smaller than tolerance, convergence is achieved.
 Otherwise starts backward sweep. And again.
 voltages and currents of convergence are saved in voltage and currentLoad.

Definition at line 162 of file LVN_Node.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.38.3.9 save()

```
void LVN_Node::save( )
```

Definition at line 175 of file LVN_Node.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

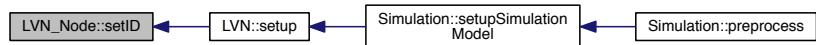


15.38.3.10 setID()

```
void LVN_Node::setID (
    const int id )
```

Definition at line 227 of file LVN_Node.cpp.

Here is the caller graph for this function:



15.38.3.11 setImpedance()

```
void LVN_Node::setImpedance (
    const std::complex< double > & value )
```

Set value of the cable impedance between self node and the parent node.

Parameters

<code>value</code>	value of impedance in Ohms
--------------------	----------------------------

Definition at line 55 of file LVN_Node.cpp.

Here is the caller graph for this function:



15.38.3.12 setNodeLoad()

```
void LVN_Node::setNodeLoad (
    const std::complex< double > & nodeLoad )
```

Assign the corresponding NodeLoad to the node, such as HouseLoad, ApplianceLoad, etc.

Parameters

<i>NodeLoad</i>	NodeLoad object contains NodeLoad.profile, an array of the power profile.
-----------------	---

Definition at line 71 of file LVN_Node.cpp.

15.38.3.13 setNominalVoltage()

```
void LVN_Node::setNominalVoltage (
    const std::complex< double > & value )
```

Set value of the nominal voltage.

Parameters

<i>value</i>	value of nominal voltage in Volts.
--------------	------------------------------------

Definition at line 63 of file LVN_Node.cpp.

15.38.3.14 setPowerForID()

```
bool LVN_Node::setPowerForID (
    const std::complex< double > & power,
    const int id )
```

set the power for node at the ID.

Parameters

<i>power</i>	the nodes power.
<i>id</i>	of the node we are searching for.

Returns

true if node is found.

Definition at line 197 of file LVN_Node.cpp.

Here is the caller graph for this function:

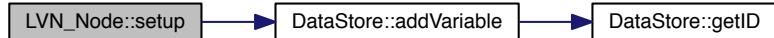


15.38.3.15 setup()

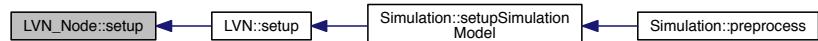
```
void LVN_Node::setup( )
```

Definition at line 20 of file LVN_Node.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



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

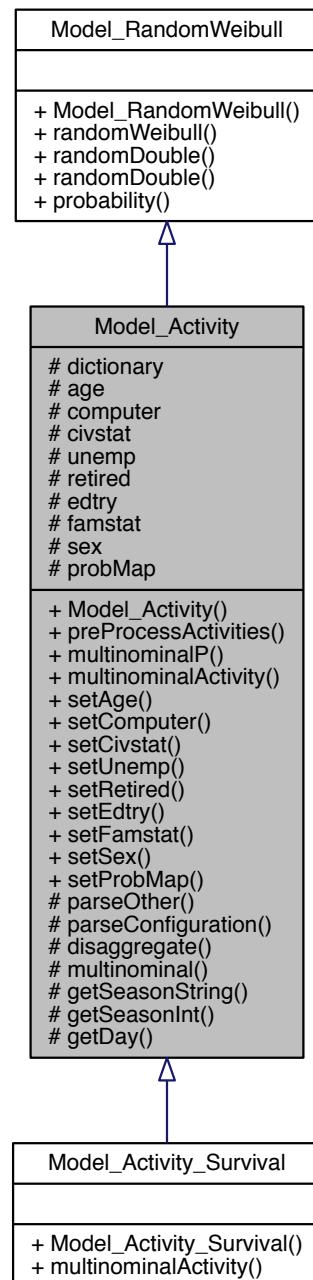
- LVN_Node.hpp
- LVN_Node.cpp

15.39 Model_Activity Class Reference

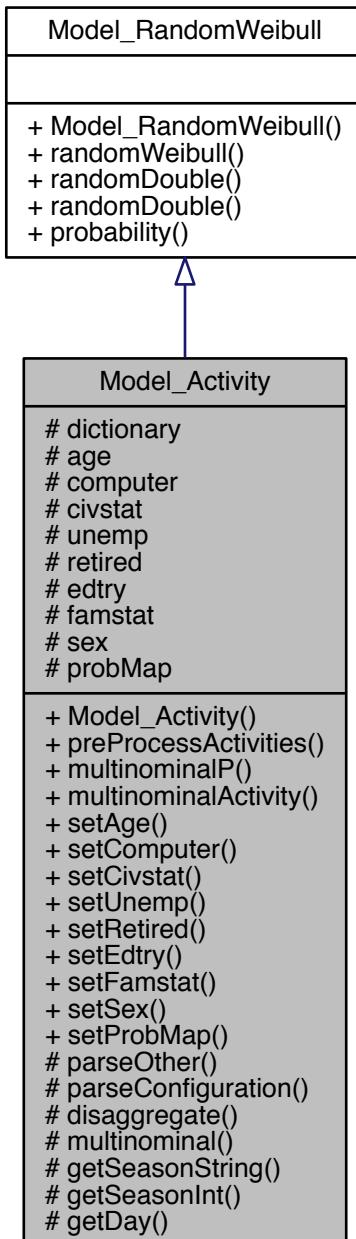
Models the large appliance with survival.

```
#include <Model_Activity.hpp>
```

Inheritance diagram for Model_Activity:



Collaboration diagram for Model_Activity:



Public Member Functions

- [Model_Activity \(\)](#)
- std::vector< double > [preProcessActivities \(\)](#)
- void [multinomialP \(double p\[4\]\[7\]\[24\]\[10\]\) const](#)
- virtual int [multinomialActivity \(const double p\[24\]\[10\], const int hourCount\)](#)
- void [setAge \(const std::string &age\)](#)

- void `setComputer` (const std::string &`computer`)
- void `setCivstat` (const std::string &`civstat`)
- void `setUnemp` (const std::string &`unemp`)
- void `setRetired` (const std::string &`retired`)
- void `setEdtry` (const std::string &`edtry`)
- void `setFamstat` (const std::string &`famstat`)
- void `setSex` (const std::string &`sex`)
- void `setProbMap` (const std::map< int, std::string > &`probMap`)

Protected Member Functions

- virtual void `parseOther` (rapidxml::xml_node<> *`node`)
- void `parseConfiguration` (const std::string &`filename`)
- std::vector< double > `disaggregate` () const
- std::vector< double > `multinomial` ()
- std::string `getSeasonString` (const int `month`) const
- int `getSeasonInt` (const int `month`) const
- std::string `getDay` (const int `day`) const

Protected Attributes

- std::map< int, std::map< std::string, std::vector< double > > > `dictionary`
- std::string `age`
- std::string `computer`
- std::string `civstat`
- std::string `unemp`
- std::string `retired`
- std::string `edtry`
- std::string `famstat`
- std::string `sex`
- std::map< int, std::string > `probMap`

Additional Inherited Members

15.39.1 Detailed Description

Models the large appliance with survival.

Models activity model for prediction of activity at each timestep adapted from Jaboob, S. (2015). Stochastic Modelling of Occupants' Activities and Related Behaviours. The University of Nottingham.

Definition at line 20 of file Model_Activity.hpp.

15.39.2 Constructor & Destructor Documentation

15.39.2.1 Model_Activity()

```
Model_Activity::Model_Activity ( )
```

sleep passive Audio / visul IT Cooking Cleaning Other Washing Metabolic Washing appliance out

Definition at line 35 of file Model_Activity.cpp.

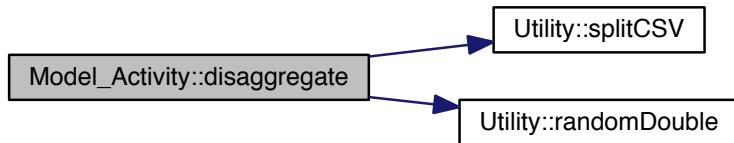
15.39.3 Member Function Documentation

15.39.3.1 disaggregate()

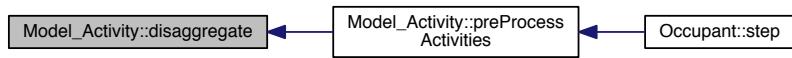
```
std::vector< double > Model_Activity::disaggregate ( ) const [protected]
```

Definition at line 222 of file Model_Activity.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.39.3.2 getDay()

```
std::string Model_Activity::getDay (
    const int day ) const [protected]
```

Definition at line 89 of file Model_Activity.cpp.

Here is the caller graph for this function:

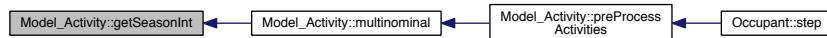


15.39.3.3 getSeasonInt()

```
int Model_Activity::getSeasonInt (
    const int month ) const [protected]
```

Definition at line 65 of file Model_Activity.cpp.

Here is the caller graph for this function:

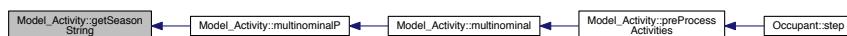


15.39.3.4 getSeasonString()

```
std::string Model_Activity::getSeasonString (
    const int month ) const [protected]
```

Definition at line 47 of file Model_Activity.cpp.

Here is the caller graph for this function:

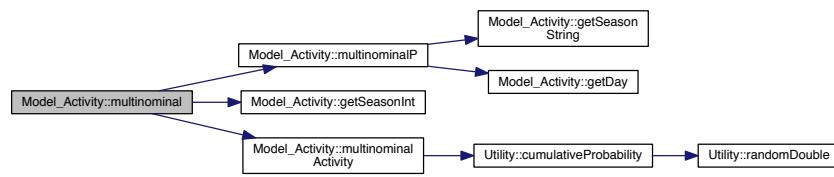


15.39.3.5 multinomial()

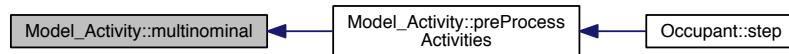
```
std::vector< double > Model_Activity::multinomial () [protected]
```

Definition at line 140 of file Model_Activity.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



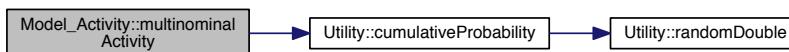
15.39.3.6 multinomialActivity()

```
int Model_Activity::multinomialActivity (
    const double p[24][10],
    const int hourCount ) [virtual]
```

Reimplemented in [Model_Activity_Survival](#).

Definition at line 93 of file Model_Activity.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

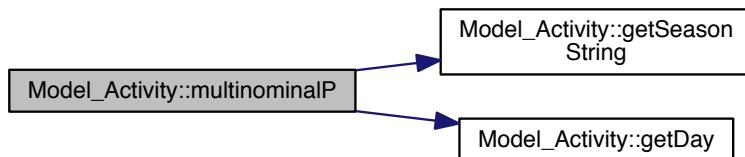


15.39.3.7 multinomialP()

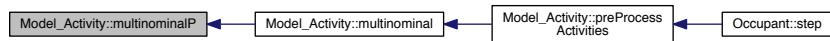
```
void Model_Activity::multinomialP (
    double p[4][7][24][10] ) const
```

Definition at line 98 of file Model_Activity.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

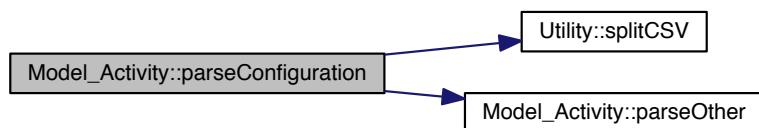


15.39.3.8 parseConfiguration()

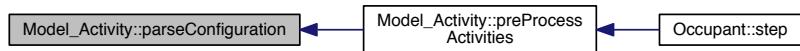
```
void Model_Activity::parseConfiguration (
    const std::string & filename ) [protected]
```

Definition at line 183 of file Model_Activity.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

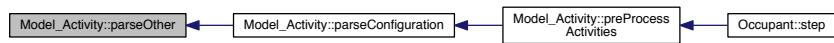


15.39.3.9 parseOther()

```
void Model_Activity::parseOther (
    rapidxml::xml_node<> * node ) [protected], [virtual]
```

Definition at line 219 of file Model_Activity.cpp.

Here is the caller graph for this function:

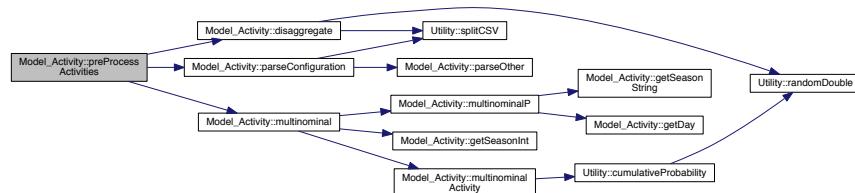


15.39.3.10 preprocessActivities()

```
std::vector< double > Model_Activity::preprocessActivities ( )
```

Definition at line 37 of file Model_Activity.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.39.3.11 setAge()

```
void Model_Activity::setAge ( const std::string & age )
```

Definition at line 259 of file Model_Activity.cpp.

Here is the caller graph for this function:



15.39.3.12 setCivstat()

```
void Model_Activity::setCivstat ( const std::string & civstat )
```

Definition at line 267 of file Model_Activity.cpp.

Here is the caller graph for this function:



15.39.3.13 setComputer()

```
void Model_Activity::setComputer (
    const std::string & computer )
```

Definition at line 263 of file Model_Activity.cpp.

Here is the caller graph for this function:



15.39.3.14 setEdtry()

```
void Model_Activity::setEdtry (
    const std::string & edtry )
```

Definition at line 279 of file Model_Activity.cpp.

Here is the caller graph for this function:



15.39.3.15 setFamstat()

```
void Model_Activity::setFamstat (
    const std::string & famstat )
```

Definition at line 283 of file Model_Activity.cpp.

Here is the caller graph for this function:



15.39.3.16 setProbMap()

```
void Model_Activity::setProbMap (const std::map< int, std::string > & probMap )
```

Definition at line 291 of file Model_Activity.cpp.

Here is the caller graph for this function:

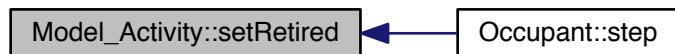


15.39.3.17 setRetired()

```
void Model_Activity::setRetired (const std::string & retired )
```

Definition at line 275 of file Model_Activity.cpp.

Here is the caller graph for this function:



15.39.3.18 setSex()

```
void Model_Activity::setSex (const std::string & sex )
```

Definition at line 287 of file Model_Activity.cpp.

Here is the caller graph for this function:



15.39.3.19 setUnemp()

```
void Model_Activity::setUnemp (const std::string & unemp )
```

Definition at line 271 of file Model_Activity.cpp.

Here is the caller graph for this function:



15.39.4 Member Data Documentation

15.39.4.1 age

```
std::string Model_Activity::age [protected]
```

Definition at line 48 of file Model_Activity.hpp.

15.39.4.2 civstat

```
std::string Model_Activity::civstat [protected]
```

Definition at line 50 of file Model_Activity.hpp.

15.39.4.3 computer

```
std::string Model_Activity::computer [protected]
```

Definition at line 49 of file Model_Activity.hpp.

15.39.4.4 dictionary

```
std::map<int, std::map<std::string, std::vector<double> > > Model_Activity::dictionary [protected]
```

Definition at line 46 of file Model_Activity.hpp.

15.39.4.5 edtry

```
std::string Model_Activity::edtry [protected]
```

Definition at line 53 of file Model_Activity.hpp.

15.39.4.6 famstat

```
std::string Model_Activity::famstat [protected]
```

Definition at line 54 of file Model_Activity.hpp.

15.39.4.7 probMap

```
std::map<int, std::string> Model_Activity::probMap [protected]
```

Definition at line 56 of file Model_Activity.hpp.

15.39.4.8 retired

```
std::string Model_Activity::retired [protected]
```

Definition at line 52 of file Model_Activity.hpp.

15.39.4.9 sex

```
std::string Model_Activity::sex [protected]
```

Definition at line 55 of file Model_Activity.hpp.

15.39.4.10 unemp

```
std::string Model_Activity::unemp [protected]
```

Definition at line 51 of file Model_Activity.hpp.

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

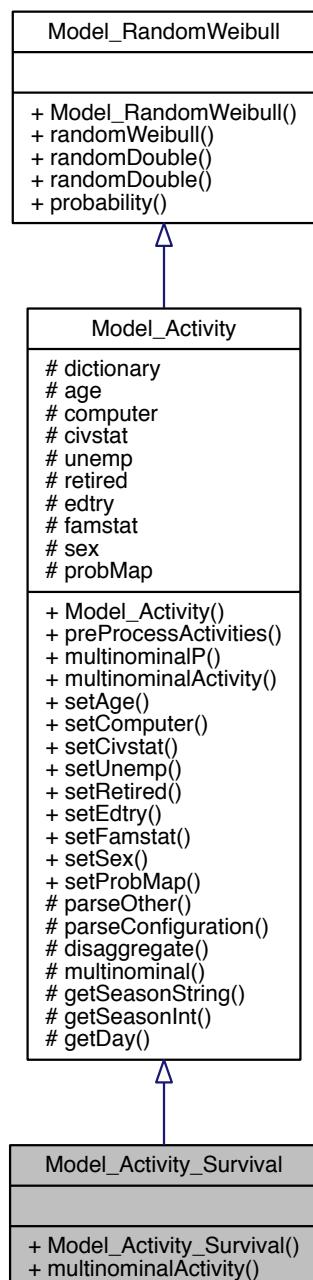
- Model_Activity.hpp
- Model_Activity.cpp

15.40 Model_Activity_Survival Class Reference

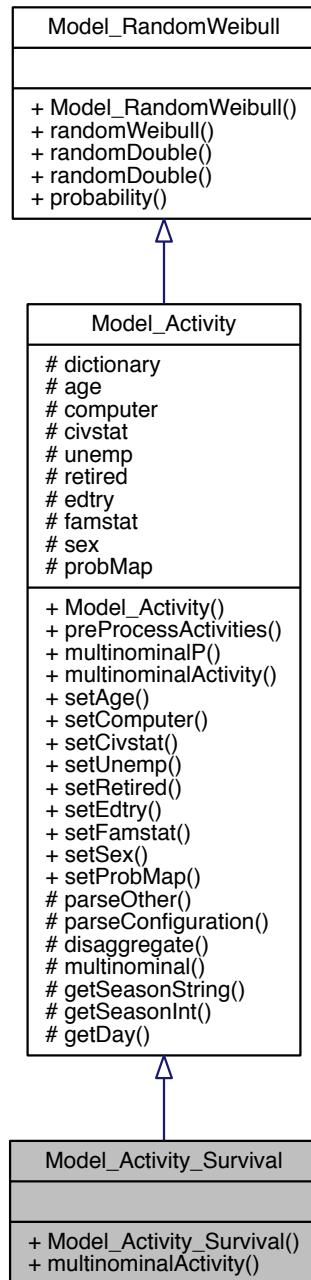
Models the survival function for the activity model.

```
#include <Model_Activity_Survival.hpp>
```

Inheritance diagram for Model_Activity_Survival:



Collaboration diagram for Model_Activity_Survival:



Public Member Functions

- [Model_Activity_Survival \(\)](#)
- int [multinomialActivity \(const double p\[24\]\[10\], const int hourCount\)](#)

Additional Inherited Members

15.40.1 Detailed Description

Models the survival function for the activity model.

Models the survival function for the activity model adapted from
 Jaboob, S. (2015). Stochastic Modelling of Occupants' Activities and Related Behaviours. The University of Nottingham.

Definition at line 17 of file Model_Activity_Survival.hpp.

15.40.2 Constructor & Destructor Documentation

15.40.2.1 Model_Activity_Survival()

```
Model_Activity_Survival::Model_Activity_Survival ( )
```

sleep passive Audio / visual IT Cooking Cleaning Other Washing Metabolic Washing appliance out

Definition at line 24 of file Model_Activity_Survival.cpp.

15.40.3 Member Function Documentation

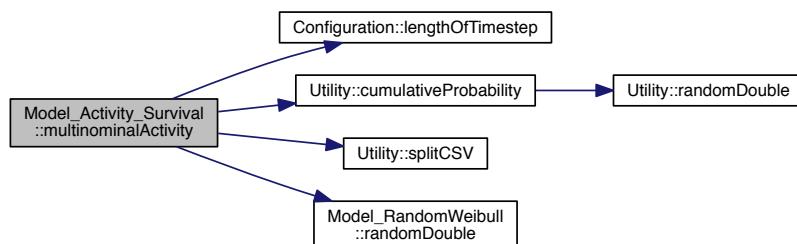
15.40.3.1 multinomialActivity()

```
int Model_Activity_Survival::multinomialActivity (
    const double p[24][10],
    const int hourCount ) [virtual]
```

Reimplemented from [Model_Activity](#).

Definition at line 28 of file Model_Activity_Survival.cpp.

Here is the call graph for this function:



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

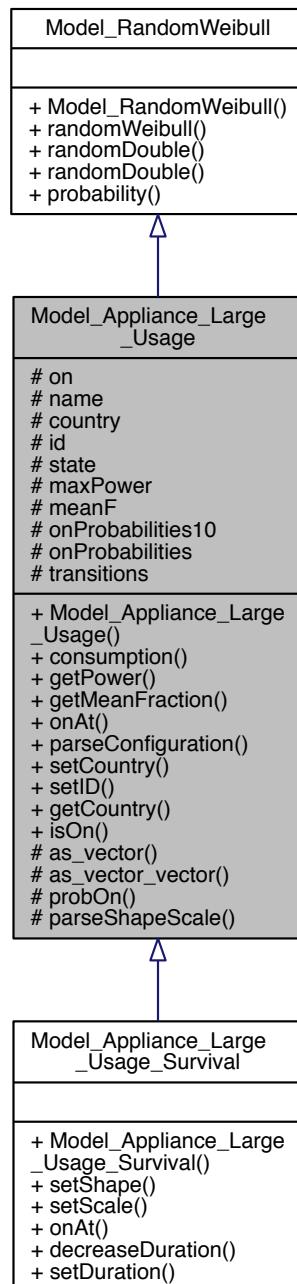
- Model_Activity_Survival.hpp
- Model_Activity_Survival.cpp

15.41 Model_Appliance_Large_Usage Class Reference

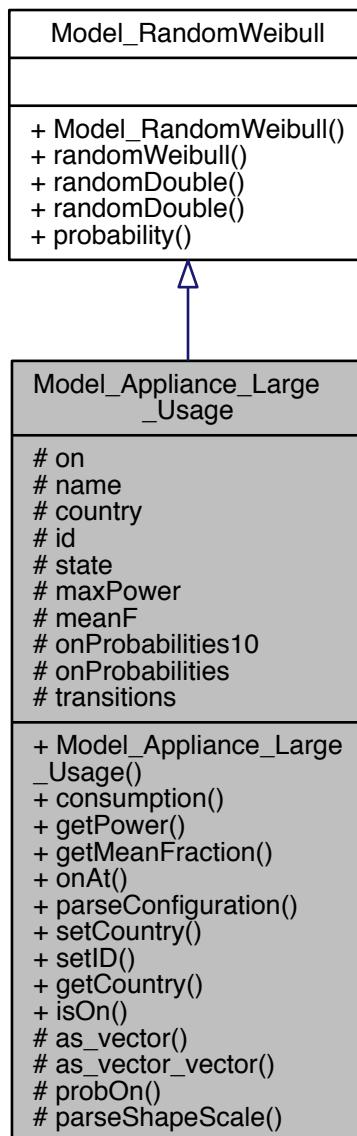
Models the large appliance.

```
#include <Model_Appliance_Large_Usage.hpp>
```

Inheritance diagram for Model_Appliance_Large_Usage:



Collaboration diagram for Model_Appliance_Large_Usage:



Public Member Functions

- `Model_Appliance_Large_Usage ()`
- `double consumption (const int timeStep)`
- `double getPower ()`
- `double getMeanFraction ()`
- `virtual bool onAt (const int timeStep)`
- `void parseConfiguration (const std::string &filename)`
- `void setCountry (const std::string &name)`
- `void setID (const int id)`
- `std::string getCountry ()`
- `bool isOn () const`

Protected Member Functions

- template<typename T >
std::vector< T > **as_vector** (rapidxml::xml_node<> *node)
- template<typename T >
std::vector< std::vector< T > > **as_vector_vector** (rapidxml::xml_node<> *node)
- double **probOn** (int timestep) const
- virtual void **parseShapeScale** (rapidxml::xml_node<> *node)

Protected Attributes

- bool **on**
- std::string **name**
- std::string **country**
- int **id**
- int **state**
- double **maxPower**
- std::vector< double > **meanF**
- std::vector< double > **onProbabilities10**
- std::vector< double > **onProbabilities**
- std::vector< std::vector< double > > **transitions**

Additional Inherited Members

15.41.1 Detailed Description

Models the large appliance.

Models large appliance for prediction of appliance use at each timestep adapted from Jaboob, S. (2015). Stochastic Modelling of Occupants' Activities and Related Behaviours. The University of Nottingham.

Definition at line 19 of file Model_Appliance_Large_Usage.hpp.

15.41.2 Constructor & Destructor Documentation

15.41.2.1 Model_Appliance_Large_Usage()

```
Model_Appliance_Large_Usage::Model_Appliance_Large_Usage ( )
```

Definition at line 11 of file Model_Appliance_Large_Usage.cpp.

Here is the call graph for this function:



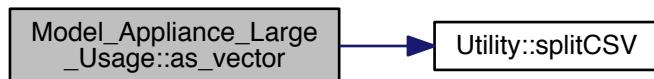
15.41.3 Member Function Documentation

15.41.3.1 as_vector()

```
template<typename T >
std::vector< T > Model_Appliance_Large_Usage::as_vector (
    rapidxml::xml_node<> * node ) [protected]
```

Definition at line 73 of file Model_Appliance_Large_Usage.cpp.

Here is the call graph for this function:



15.41.3.2 as_vector_vector()

```
template<typename T >
std::vector< std::vector< T > > Model_Appliance_Large_Usage::as_vector_vector (
    rapidxml::xml_node<> * node ) [protected]
```

Definition at line 87 of file Model_Appliance_Large_Usage.cpp.

15.41.3.3 consumption()

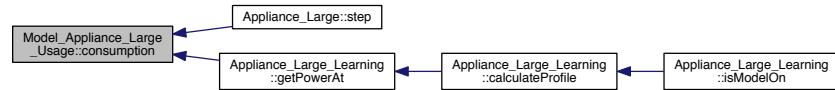
```
double Model_Appliance_Large_Usage::consumption (
    const int timestep )
```

Definition at line 40 of file Model_Appliance_Large_Usage.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.41.3.4 getCountry()

```
std::string Model_Appliance_Large_Usage::getCountry( )
```

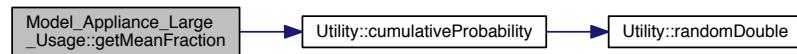
Definition at line 23 of file Model_Appliance_Large_Usage.cpp.

15.41.3.5 getMeanFraction()

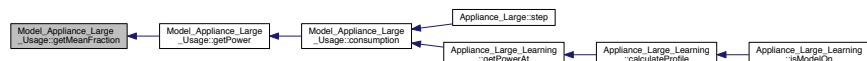
```
double Model_Appliance_Large_Usage::getMeanFraction( )
```

Definition at line 56 of file Model_Appliance_Large_Usage.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

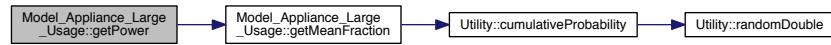


15.41.3.6 getPower()

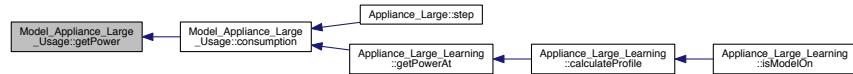
```
double Model_Appliance_Large_Usage::getPower ( )
```

Definition at line 52 of file Model_Appliance_Large_Usage.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

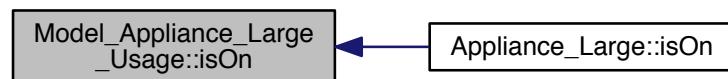


15.41.3.7 isOn()

```
bool Model_Appliance_Large_Usage::isOn ( ) const
```

Definition at line 61 of file Model_Appliance_Large_Usage.cpp.

Here is the caller graph for this function:



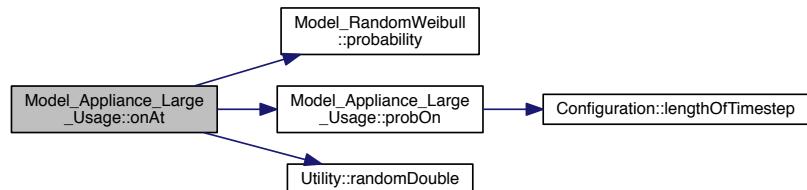
15.41.3.8 onAt()

```
bool Model_Appliance_Large_Usage::onAt (
    const int timeStep ) [virtual]
```

Reimplemented in [Model_Appliance_Large_Usage_Survival](#).

Definition at line 65 of file [Model_Appliance_Large_Usage.cpp](#).

Here is the call graph for this function:



15.41.3.9 parseConfiguration()

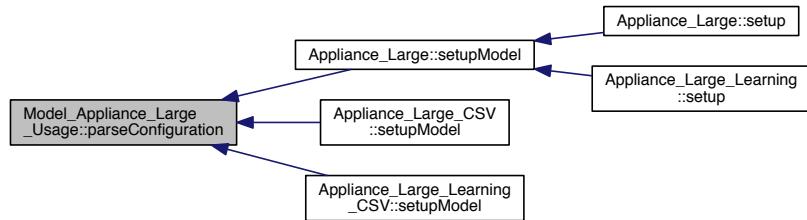
```
void Model_Appliance_Large_Usage::parseConfiguration (
    const std::string & filename )
```

Definition at line 98 of file [Model_Appliance_Large_Usage.cpp](#).

Here is the call graph for this function:



Here is the caller graph for this function:

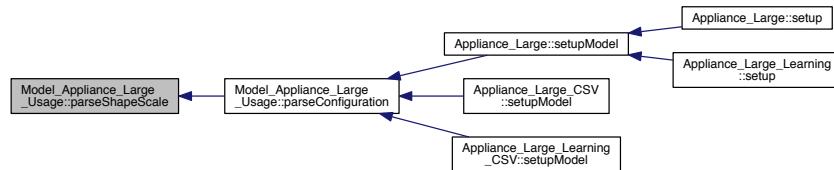


15.41.3.10 parseShapeScale()

```
void Model_Appliance_Large_Usage::parseShapeScale (
    rapidxml::xml_node<> * node ) [protected], [virtual]
```

Definition at line 148 of file Model_Appliance_Large_Usage.cpp.

Here is the caller graph for this function:

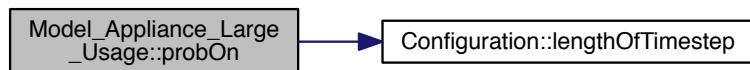


15.41.3.11 probOn()

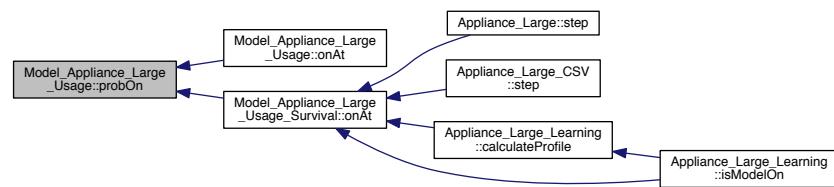
```
double Model_Appliance_Large_Usage::probOn (
    int timestep ) const [protected]
```

Definition at line 27 of file Model_Appliance_Large_Usage.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.41.3.12 setCountry()

```
void Model_Appliance_Large_Usage::setCountry (
    const std::string & name )
```

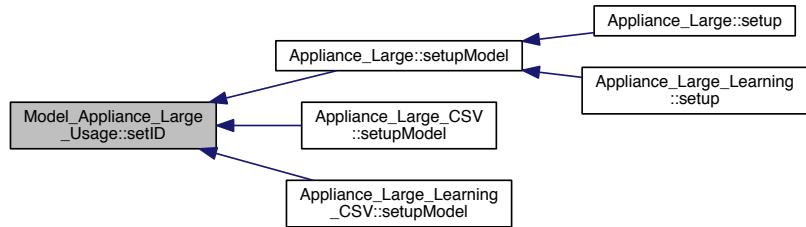
Definition at line 15 of file Model_Appliance_Large_Usage.cpp.

15.41.3.13 setId()

```
void Model_Appliance_Large_Usage::setId (
    const int id )
```

Definition at line 19 of file Model_Appliance_Large_Usage.cpp.

Here is the caller graph for this function:



15.41.4 Member Data Documentation

15.41.4.1 country

```
std::string Model_Appliance_Large_Usage::country [protected]
```

Definition at line 43 of file Model_Appliance_Large_Usage.hpp.

15.41.4.2 id

```
int Model_Appliance_Large_Usage::id [protected]
```

Definition at line 44 of file Model_Appliance_Large_Usage.hpp.

15.41.4.3 maxPower

```
double Model_Appliance_Large_Usage::maxPower [protected]
```

Definition at line 46 of file Model_Appliance_Large_Usage.hpp.

15.41.4.4 meanF

```
std::vector<double> Model_Appliance_Large_Usage::meanF [protected]
```

Definition at line 47 of file Model_Appliance_Large_Usage.hpp.

15.41.4.5 name

```
std::string Model_Appliance_Large_Usage::name [protected]
```

Definition at line 42 of file Model_Appliance_Large_Usage.hpp.

15.41.4.6 on

```
bool Model_Appliance_Large_Usage::on [protected]
```

Definition at line 41 of file Model_Appliance_Large_Usage.hpp.

15.41.4.7 onProbabilities

```
std::vector<double> Model_Appliance_Large_Usage::onProbabilities [protected]
```

Definition at line 49 of file Model_Appliance_Large_Usage.hpp.

15.41.4.8 onProbabilities10

```
std::vector<double> Model_Appliance_Large_Usage::onProbabilities10 [protected]
```

Definition at line 48 of file Model_Appliance_Large_Usage.hpp.

15.41.4.9 state

```
int Model_Appliance_Large_Usage::state [protected]
```

Definition at line 45 of file Model_Appliance_Large_Usage.hpp.

15.41.4.10 transitions

```
std::vector<std::vector<double>> Model_Appliance_Large_Usage::transitions [protected]
```

Definition at line 50 of file Model_Appliance_Large_Usage.hpp.

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

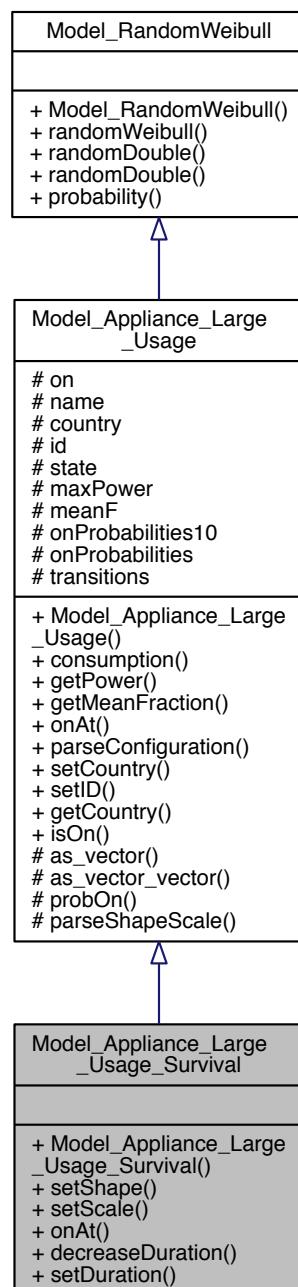
- Model_Appliance_Large_Usage.hpp
- Model_Appliance_Large_Usage.cpp

15.42 Model_Appliance_Large_Usage_Survival Class Reference

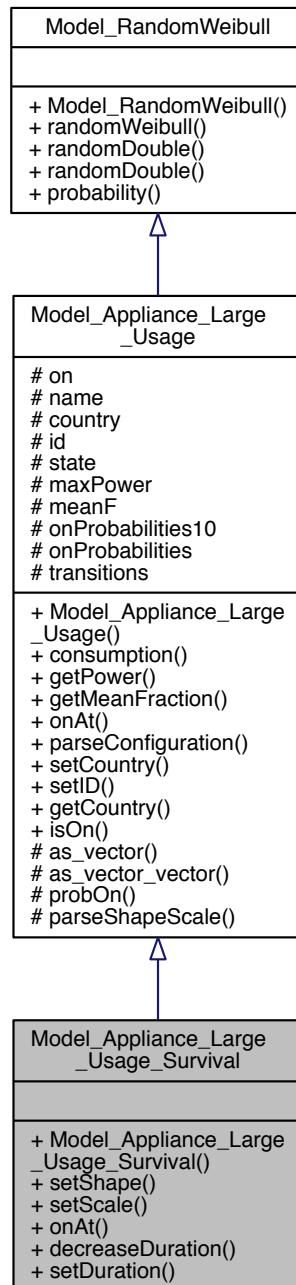
Models the large appliance with survival.

```
#include <Model_Appliance_Large_Usage_Survival.hpp>
```

Inheritance diagram for Model_Appliance_Large_Usage_Survival:



Collaboration diagram for Model_Appliance_Large_Usage_Survival:



Public Member Functions

- [Model_Appliance_Large_Usage_Survival \(\)](#)
- void [setShape](#) (double shape)
- void [setScale](#) (double scale)
- bool [onAt](#) (const int timeStep)
- void [decreaseDuration](#) ()
- void [setDuration](#) (double duration)

Additional Inherited Members

15.42.1 Detailed Description

Models the large appliance with survival.

Models large appliance model for prediction of appliance use at each timestep with appliance survival in on state
Jaboor, S. (2015). Stochastic Modelling of Occupants' Activities and Related Behaviours. The University of Nottingham.

Definition at line 17 of file Model_Appliance_Large_Usage_Survival.hpp.

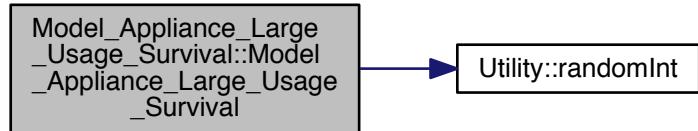
15.42.2 Constructor & Destructor Documentation

15.42.2.1 Model_Appliance_Large_Usage_Survival()

```
Model_Appliance_Large_Usage_Survival::Model_Appliance_Large_Usage_Survival ( )
```

Definition at line 11 of file Model_Appliance_Large_Usage_Survival.cpp.

Here is the call graph for this function:



15.42.3 Member Function Documentation

15.42.3.1 decreaseDuration()

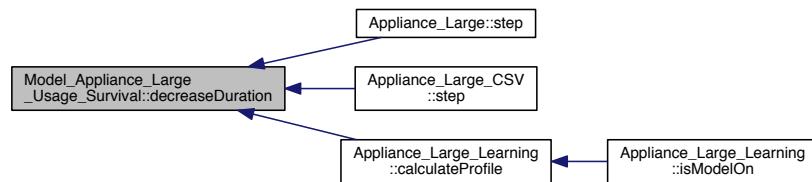
```
void Model_Appliance_Large_Usage_Survival::decreaseDuration ( )
```

Definition at line 15 of file Model_Appliance_Large_Usage_Survival.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



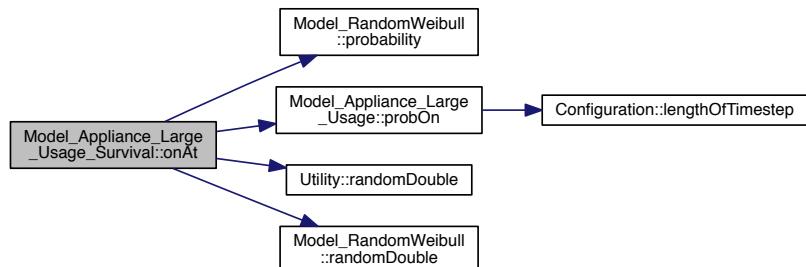
15.42.3.2 onAt()

```
bool Model_Appliance_Large_Usage_Survival::onAt (
    const int timeStep ) [virtual]
```

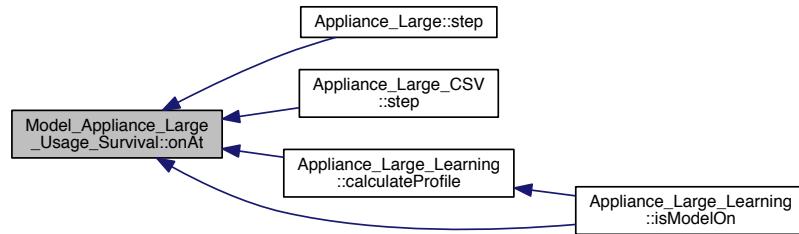
Reimplemented from [Model_Appliance_Large_Usage](#).

Definition at line 19 of file Model_Appliance_Large_Usage_Survival.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

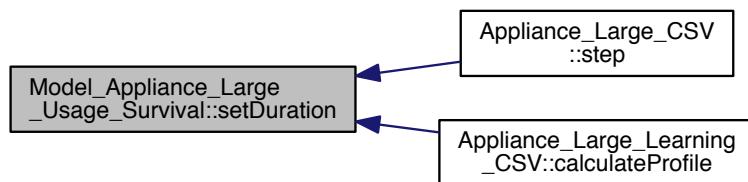


15.42.3.3 setDuration()

```
void Model_Appliance_Large_Usage_Survival::setDuration (
    double duration )
```

Definition at line 61 of file `Model_Appliance_Large_Usage_Survival.cpp`.

Here is the caller graph for this function:



15.42.3.4 setScale()

```
void Model_Appliance_Large_Usage_Survival::setScale (
    double scale )
```

Definition at line 48 of file `Model_Appliance_Large_Usage_Survival.cpp`.

15.42.3.5 setShape()

```
void Model_Appliance_Large_Usage_Survival::setShape (
    double shape )
```

Definition at line 45 of file Model_Appliance_Large_Usage_Survival.cpp.

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

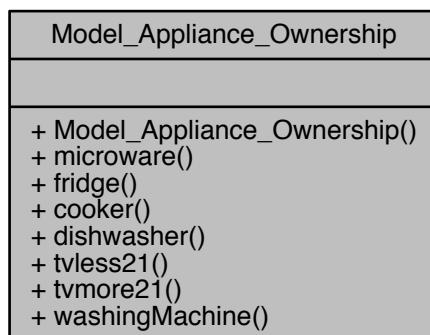
- Model_Appliance_Large_Usage_Survival.hpp
- Model_Appliance_Large_Usage_Survival.cpp

15.43 Model_Appliance_Ownership Class Reference

Models appliance ownership.

```
#include <Model_Appliance_Ownership.hpp>
```

Collaboration diagram for Model_Appliance_Ownership:



Public Member Functions

- [Model_Appliance_Ownership \(\)](#)
- [double microware \(\) const](#)
- [double fridge \(\) const](#)
- [double cooker \(const int age, const int employment, const int bedrooms, const int education\) const](#)
- [double dishwasher \(const int socclass, const int internet, const int cohabitants, const int hometype, const int ownrent, const int bedrooms, const int internetreg, const int houseyear\) const](#)
- [double tvless21 \(const int sex, const int internet, const int cohabitants, const int over15, const int education\) const](#)
- [double tvmore21 \(const int sex, const int employment, const int cohabitants, const int hometype, const int ownrent, const int bedrooms, const int education, const int internetregother\) const](#)
- [double washingMachine \(const int sex, const int ownrent, const int bedrooms, const int internetregother\) const](#)

15.43.1 Detailed Description

Models appliance ownership.

Models appliance ownership, based on inputs provide in configuration file adapted from Jaboob, S. (2015). Stochastic Modelling of Occupants' Activities and Related Behaviours. The University of Nottingham.

Definition at line 12 of file Model_Appliance_Ownership.hpp.

15.43.2 Constructor & Destructor Documentation

15.43.2.1 Model_Appliance_Ownership()

```
Model_Appliance_Ownership::Model_Appliance_Ownership ( )
```

Definition at line 5 of file Model_Appliance_Ownership.cpp.

15.43.3 Member Function Documentation

15.43.3.1 cooker()

```
double Model_Appliance_Ownership::cooker (
    const int age,
    const int employment,
    const int bedrooms,
    const int education ) const
```

Definition at line 68 of file Model_Appliance_Ownership.cpp.

15.43.3.2 dishwasher()

```
double Model_Appliance_Ownership::dishwasher (
    const int soclass,
    const int internet,
    const int cohabitants,
    const int hometype,
    const int ownrent,
    const int bedrooms,
    const int internetreg,
    const int houseyear ) const
```

Definition at line 7 of file Model_Appliance_Ownership.cpp.

15.43.3.3 `fridge()`

```
double Model_Appliance_Ownership::fridge ( ) const
```

Definition at line 81 of file Model_Appliance_Ownership.cpp.

15.43.3.4 `microwave()`

```
double Model_Appliance_Ownership::microwave ( ) const
```

Definition at line 63 of file Model_Appliance_Ownership.cpp.

15.43.3.5 `tvless21()`

```
double Model_Appliance_Ownership::tvless21 (
    const int sex,
    const int internet,
    const int cohabitants,
    const int over15,
    const int education ) const
```

Definition at line 26 of file Model_Appliance_Ownership.cpp.

15.43.3.6 `tvmore21()`

```
double Model_Appliance_Ownership::tvmore21 (
    const int sex,
    const int employment,
    const int cohabitants,
    const int hometype,
    const int ownrent,
    const int bedrooms,
    const int education,
    const int internetregother ) const
```

Definition at line 42 of file Model_Appliance_Ownership.cpp.

15.43.3.7 washingMachine()

```
double Model_Appliance_Ownership::washingMachine ( const int sex, const int ownrent, const int bedrooms, const int internetregother ) const
```

Definition at line 86 of file Model_Appliance_Ownership.cpp.

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

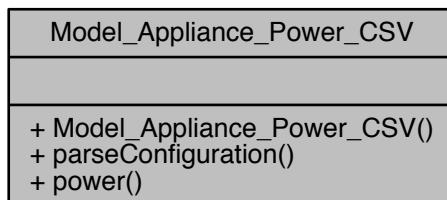
- Model_Appliance_Ownership.hpp
- Model_Appliance_Ownership.cpp

15.44 Model_Appliance_Power_CSV Class Reference

Appliance modeled from a CSV file.

```
#include <Model_Appliance_Power_CSV.hpp>
```

Collaboration diagram for Model_Appliance_Power_CSV:



Public Member Functions

- [Model_Appliance_Power_CSV \(\)](#)
- void [parseConfiguration \(const std::string &filename\)](#)
- double [power \(const int dayOfYear, const int minuteOfDay\)](#)

15.44.1 Detailed Description

Appliance modeled from a CSV file.

Models appliance usage at each timestep from a CSV file

Definition at line 14 of file Model_Appliance_Power_CSV.hpp.

15.44.2 Constructor & Destructor Documentation

15.44.2.1 Model_Appliance_Power_CSV()

```
Model_Appliance_Power_CSV::Model_Appliance_Power_CSV ( )
```

Definition at line 7 of file Model_Appliance_Power_CSV.cpp.

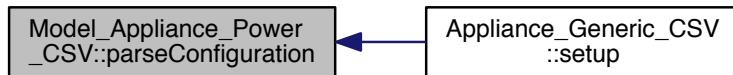
15.44.3 Member Function Documentation

15.44.3.1 parseConfiguration()

```
void Model_Appliance_Power_CSV::parseConfiguration (
    const std::string & filename )
```

Definition at line 9 of file Model_Appliance_Power_CSV.cpp.

Here is the caller graph for this function:



15.44.3.2 power()

```
double Model_Appliance_Power_CSV::power (
    const int dayOfYear,
    const int minuteOfDay )
```

Definition at line 14 of file Model_Appliance_Power_CSV.cpp.

Here is the caller graph for this function:



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

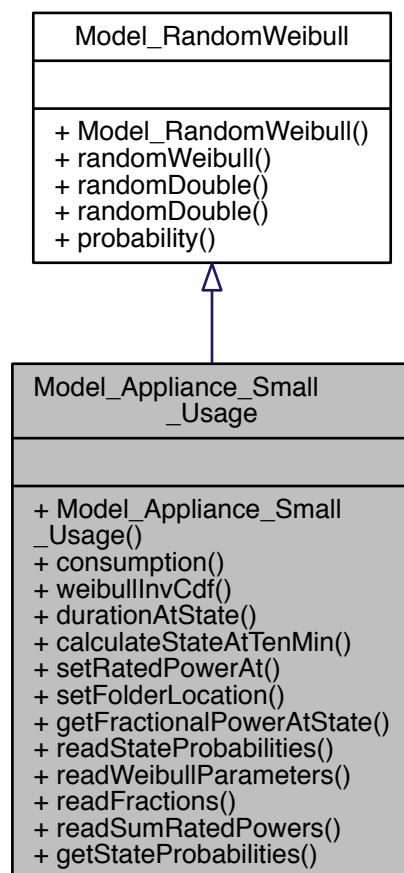
- Model_Appliance_Power_CSV.hpp
- Model_Appliance_Power_CSV.cpp

15.45 Model_Appliance_Small_Usage Class Reference

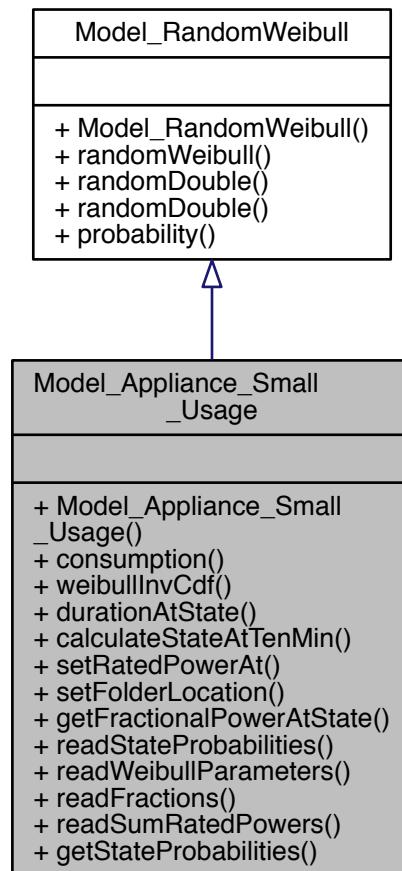
Models small appliance.

```
#include <Model_Appliance_Small_Usage.hpp>
```

Inheritance diagram for Model_Appliance_Small_Usage:



Collaboration diagram for Model_Appliance_Small_Usage:



Public Member Functions

- `Model_Appliance_Small_Usage ()`
- `double consumption (const int timeStep)`
- `double weibullInvCdf (float loc, float shape, float scale) const`
- `double durationAtState (int state) const`
- `int calculateStateAtTenMin (int timeAsInt) const`
- `void setRatedPowerAt (const int i)`
- `void setFolderLocation (const std::string &folderLocation)`
- `double getFractionalPowerAtState (int state) const`
- `void readStateProbabilities (const std::string &file)`
- `void readWeibullParameters (const std::string &file)`
- `void readFractions (const std::string &file)`
- `void readSumRatedPowers (const std::string &file)`
- `Utility::uTable< double > getStateProbabilities () const`

Additional Inherited Members

15.45.1 Detailed Description

Models small appliance.

Models small appliance for prediction of appliance use at each timestep

Sancho-Tomás, A., Chapman, J., & Robinson, D. (2017). Extending No-MASS: Multi-Agent Stochastic Simulation for Demand Response of residential appliances. In Building Simulation 2017.

Definition at line 17 of file Model_Appliance_Small_Usage.hpp.

15.45.2 Constructor & Destructor Documentation

15.45.2.1 Model_Appliance_Small_Usage()

```
Model_Appliance_Small_Usage::Model_Appliance_Small_Usage ( )
```

Definition at line 12 of file Model_Appliance_Small_Usage.cpp.

15.45.3 Member Function Documentation

15.45.3.1 calculateStateAtTenMin()

```
int Model_Appliance_Small_Usage::calculateStateAtTenMin (
    int timeAsInt ) const
```

Definition at line 49 of file Model_Appliance_Small_Usage.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

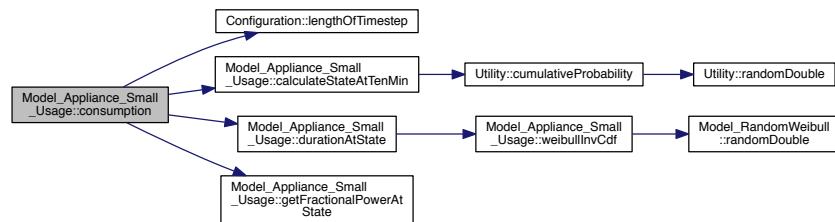


15.45.3.2 consumption()

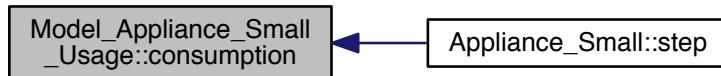
```
double Model_Appliance_Small_Usage::consumption (
    const int timestep )
```

Definition at line 16 of file Model_Appliance_Small_Usage.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.45.3.3 durationAtState()

```
double Model_Appliance_Small_Usage::durationAtState (
    int state ) const
```

Definition at line 44 of file Model_Appliance_Small_Usage.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.45.3.4 getFractionalPowerAtState()

```
double Model_Appliance_Small_Usage::getFractionalPowerAtState (
    int state ) const
```

Definition at line 35 of file Model_Appliance_Small_Usage.cpp.

Here is the caller graph for this function:



15.45.3.5 getStateProbabilities()

```
Utility::uTable<double> Model_Appliance_Small_Usage::getStateProbabilities ( ) const
```

15.45.3.6 readFractions()

```
void Model_Appliance_Small_Usage::readFractions (
    const std::string & file )
```

Definition at line 53 of file Model_Appliance_Small_Usage.cpp.

Here is the caller graph for this function:



15.45.3.7 `readStateProbabilities()`

```
void Model_Appliance_Small_Usage::readStateProbabilities (
    const std::string & file )
```

Definition at line 69 of file `Model_Appliance_Small_Usage.cpp`.

Here is the caller graph for this function:



15.45.3.8 `readSumRatedPowers()`

```
void Model_Appliance_Small_Usage::readSumRatedPowers (
    const std::string & file )
```

Definition at line 61 of file `Model_Appliance_Small_Usage.cpp`.

Here is the caller graph for this function:



15.45.3.9 `readWeibullParameters()`

```
void Model_Appliance_Small_Usage::readWeibullParameters (
    const std::string & file )
```

Definition at line 75 of file `Model_Appliance_Small_Usage.cpp`.

Here is the caller graph for this function:

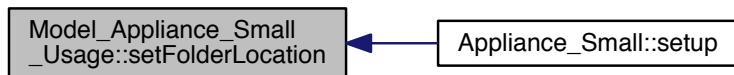


15.45.3.10 setFolderLocation()

```
void Model_Appliance_Small_Usage::setFolderLocation ( const std::string & folderLocation )
```

Definition at line 39 of file Model_Appliance_Small_Usage.cpp.

Here is the caller graph for this function:

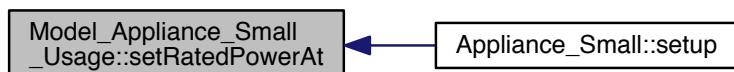


15.45.3.11 setRatedPowerAt()

```
void Model_Appliance_Small_Usage::setRatedPowerAt ( const int i )
```

Definition at line 84 of file Model_Appliance_Small_Usage.cpp.

Here is the caller graph for this function:

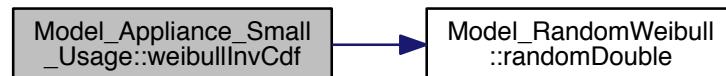


15.45.3.12 weibullInvCdf()

```
double Model_Appliance_Small_Usage::weibullInvCdf (
    float loc,
    float shape,
    float scale ) const
```

Definition at line 88 of file Model_Appliance_Small_Usage.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



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

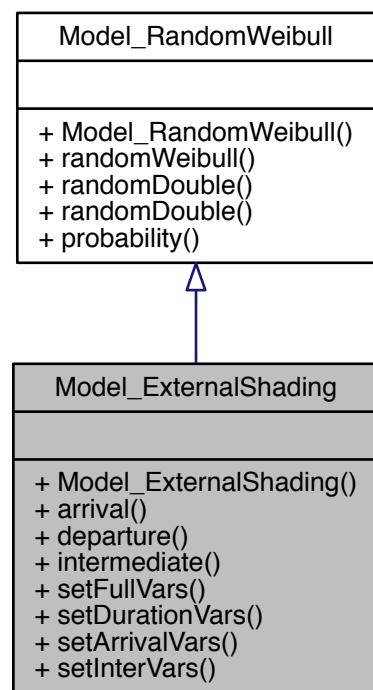
- Model_Appliance_Small_Usage.hpp
- Model_Appliance_Small_Usage.cpp

15.46 Model_ExternalShading Class Reference

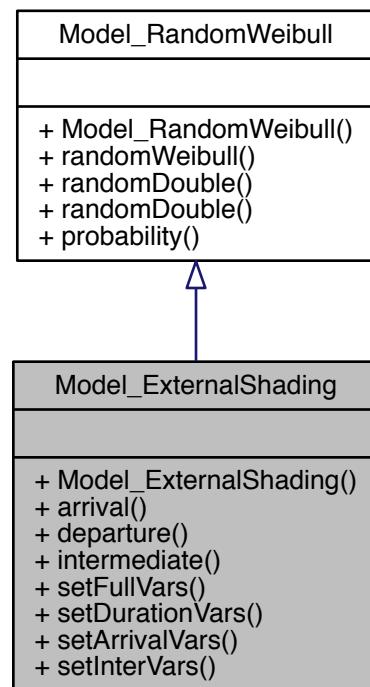
Models the prediction of external shades.

```
#include <Model_ExternalShading.hpp>
```

Inheritance diagram for Model_ExternalShading:



Collaboration diagram for Model_ExternalShading:



Public Member Functions

- `Model_ExternalShading ()`
- `double arrival (double state, double Lumint, double Evg)`
- `double departure (double state, double Lumint, double Evg)`
- `double intermediate (bool state, double Lumint, double Evg)`
- `void setFullVars (float afullraise, float boutfullraise, float bsfullraise, float bsfulllower, float boutfulllower, float afulllower)`
- `void setDurationVars (float aSFlower, float bSFlower, float shapeflower)`
- `void setArrivalVars (float a01arr, float b01inarr, float b01sarr, float a10arr, float b10inarr, float b10sarr)`
- `void setInterVars (float a01int, float b01inint, float b01sint, float a10int, float b10inint, float b10sint)`

Additional Inherited Members

15.46.1 Detailed Description

Models the prediction of external shades.

Models the prediction of external shades adapted from
Haldi, F., & Robinson, D. (2010). Adaptive actions on shading devices in response to local visual stimuli. Journal of Building Performance Simulation, 3(2), 135–153. <https://doi.org/10.1080/19401490903580759>

Definition at line 14 of file `Model_ExternalShading.hpp`.

15.46.2 Constructor & Destructor Documentation

15.46.2.1 Model_ExternalShading()

```
Model_ExternalShading::Model_ExternalShading ()
```

Definition at line 8 of file Model_ExternalShading.cpp.

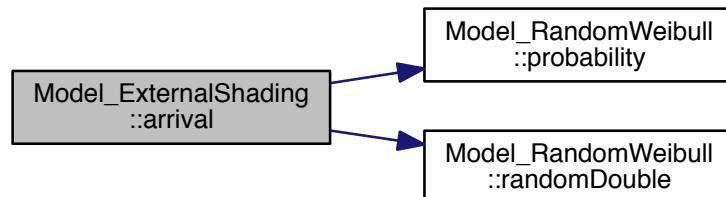
15.46.3 Member Function Documentation

15.46.3.1 arrival()

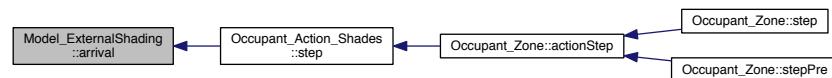
```
double Model_ExternalShading::arrival (
    double state,
    double Lumint,
    double Evg )
```

Definition at line 81 of file Model_ExternalShading.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

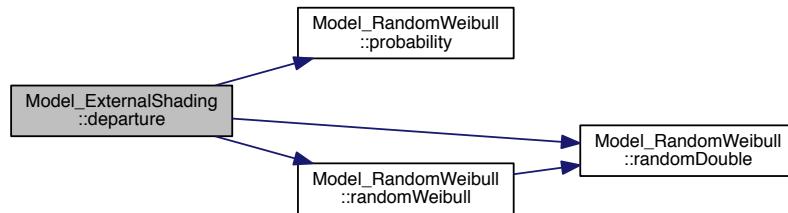


15.46.3.2 departure()

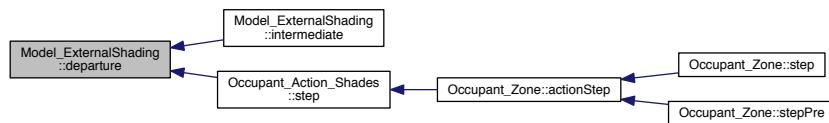
```
double Model_ExternalShading::departure (
    double state,
    double Lumint,
    double Evg )
```

Definition at line 119 of file Model_ExternalShading.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

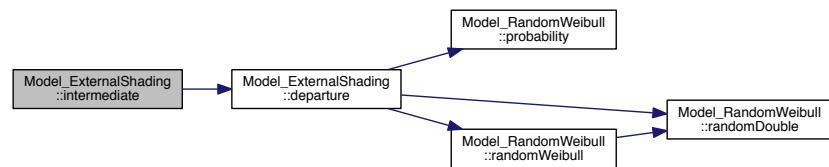


15.46.3.3 intermediate()

```
double Model_ExternalShading::intermediate (
    bool state,
    double Lumint,
    double Evg )
```

Definition at line 114 of file Model_ExternalShading.cpp.

Here is the call graph for this function:

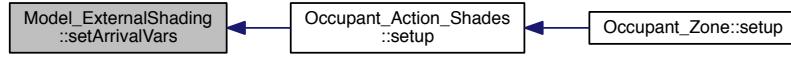


15.46.3.4 setArrivalVars()

```
void Model_ExternalShading::setArrivalVars (
    float a01arr,
    float b01inarr,
    float b01sarr,
    float a10arr,
    float b10inarr,
    float b10sarr )
```

Definition at line 58 of file Model_ExternalShading.cpp.

Here is the caller graph for this function:

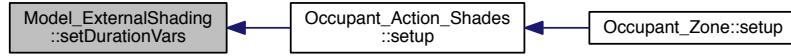


15.46.3.5 setDurationVars()

```
void Model_ExternalShading::setDurationVars (
    float aSFlower,
    float bSFlower,
    float shapeflower )
```

Definition at line 51 of file Model_ExternalShading.cpp.

Here is the caller graph for this function:

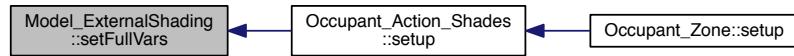


15.46.3.6 setFullVars()

```
void Model_ExternalShading::setFullVars (
    float afullraise,
    float boutfullraise,
    float bfullraise,
    float bfulllower,
    float boutfulllower,
    float afulllower )
```

Definition at line 38 of file Model_ExternalShading.cpp.

Here is the caller graph for this function:

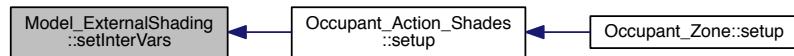


15.46.3.7 setInterVars()

```
void Model_ExternalShading::setInterVars (
    float a01int,
    float b01inint,
    float b01sint,
    float a10int,
    float b10inint,
    float b10sint )
```

Definition at line 69 of file Model_ExternalShading.cpp.

Here is the caller graph for this function:



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

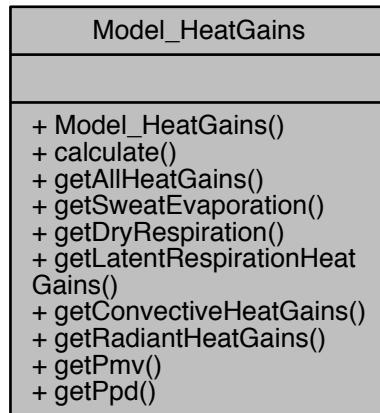
- Model_ExternalShading.hpp
- Model_ExternalShading.cpp

15.47 Model_HeatGains Class Reference

Models the occupant heat gains and PMV.

```
#include <Model_HeatGains.hpp>
```

Collaboration diagram for Model_HeatGains:



Public Member Functions

- [Model_HeatGains \(\)](#)
- void [calculate](#) (double metabolicRate, double reativeHumidity, double meanRadiantTemperature, double externalWork, double ta, double clo, double airVelocity)
- double [getAllHeatGains \(\)](#)
- double [getSweatEvaporation \(\) const](#)
- double [getDryRespiration \(\) const](#)
- double [getLatentRespirationHeatGains \(\) const](#)
- double [getConvectiveHeatGains \(\) const](#)
- double [getRadiantHeatGains \(\) const](#)
- double [getPmv \(\) const](#)
- double [getPpd \(\) const](#)

15.47.1 Detailed Description

Models the occupant heat gains and PMV.

Models the occupant heat gains and PMV adapted from ISO. (2005). ISO 7730: Ergonomics of the thermal environment Analytical determination and interpretation of thermal comfort using calculation of the PMV and PPD indices and local thermal comfort criteria. Management (Vol. 3). <https://doi.org/10.1016/j.soildyn.2004.11.005>

Definition at line 12 of file Model_HeatGains.hpp.

15.47.2 Constructor & Destructor Documentation

15.47.2.1 Model_HeatGains()

```
Model_HeatGains::Model_HeatGains ( )
```

Definition at line 6 of file Model_HeatGains.cpp.

15.47.3 Member Function Documentation

15.47.3.1 calculate()

```
void Model_HeatGains::calculate (
    double metabolicRate,
    double reativeHumidity,
    double meanRadiantTemperature,
    double externalWork,
    double ta,
    double clo,
    double airVelocity )
```

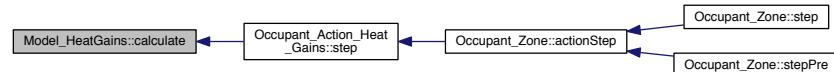
Calculates the Fanger pmv and sets the instance variables related to results.

Parameters

<i>metabolicRate</i>	Metabolic Rate
<i>partialWaterPressure</i>	partial water vapour kPa
<i>meanRadiantTemperature</i>	mean radiant temperature C
<i>externalWork</i>	external work
<i>ta</i>	air temperature
<i>clo</i>	Clothing value
<i>airVelocity</i>	Air velocity

Definition at line 104 of file Model_HeatGains.cpp.

Here is the caller graph for this function:



15.47.3.2 getAllHeatGains()

```
double Model_HeatGains::getAllHeatGains ( )
```

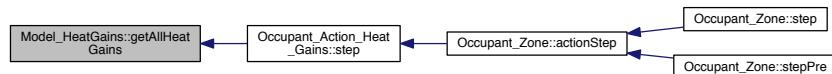
Radiant heat gains (all the sensible radiant stuff).

Returns

Radiant heat gains (all the sensible radiant stuff)

Definition at line 14 of file Model_HeatGains.cpp.

Here is the caller graph for this function:



15.47.3.3 getConvectiveHeatGains()

```
double Model_HeatGains::getConvectiveHeatGains ( ) const
```

Definition at line 35 of file Model_HeatGains.cpp.

15.47.3.4 getDryRespiration()

```
double Model_HeatGains::getDryRespiration ( ) const
```

Definition at line 27 of file Model_HeatGains.cpp.

15.47.3.5 getLatentRespirationHeatGains()

```
double Model_HeatGains::getLatentRespirationHeatGains ( ) const
```

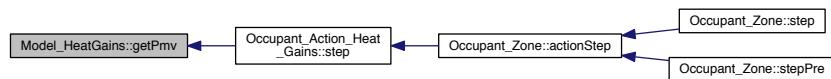
Definition at line 31 of file Model_HeatGains.cpp.

15.47.3.6 getPmv()

```
double Model_HeatGains::getPmv ( ) const
```

Definition at line 43 of file Model_HeatGains.cpp.

Here is the caller graph for this function:

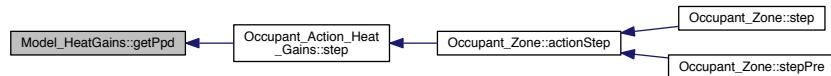


15.47.3.7 getPpd()

```
double Model_HeatGains::getPpd ( ) const
```

Definition at line 47 of file Model_HeatGains.cpp.

Here is the caller graph for this function:



15.47.3.8 getRadiantHeatGains()

```
double Model_HeatGains::getRadiantHeatGains ( ) const
```

Definition at line 39 of file Model_HeatGains.cpp.

15.47.3.9 getSweatEvaporation()

```
double Model_HeatGains::getSweatEvaporation ( ) const
```

Definition at line 23 of file Model_HeatGains.cpp.

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

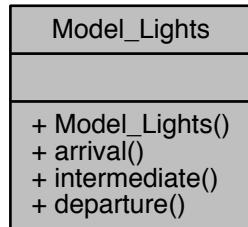
- Model_HeatGains.hpp
- Model_HeatGains.cpp

15.48 Model_Lights Class Reference

Models the lighting interactions.

```
#include <Model_Lights.hpp>
```

Collaboration diagram for Model_Lights:



Public Member Functions

- [Model_Lights \(\)](#)
- bool [arrival](#) (bool state, double Lumint)
- bool [intermediate](#) (bool state, double Lumint)
- bool [departure](#) (bool state, double futureDuration)

15.48.1 Detailed Description

Models the lighting interactions.

Models lighting interactions for an occupant adapted from
 Reinhart, C. (2004). Lightswitch-2002: a model for manual and automated control of electric lighting and blinds. *Solar Energy*, 77(1), 15–28. Retrieved from <http://www.sciencedirect.com/science/article/pii/S0038092X04000702>

Definition at line 12 of file Model_Lights.hpp.

15.48.2 Constructor & Destructor Documentation

15.48.2.1 Model_Lights()

```
Model_Lights::Model_Lights ( )
```

Definition at line 7 of file Model_Lights.cpp.

15.48.3 Member Function Documentation

15.48.3.1 arrival()

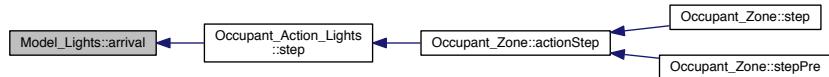
```
bool Model_Lights::arrival (
    bool state,
    double Lumint )
```

Definition at line 23 of file Model_Lights.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.48.3.2 departure()

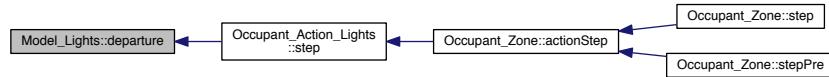
```
bool Model_Lights::departure (
    bool state,
    double futureDuration )
```

Definition at line 89 of file Model_Lights.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

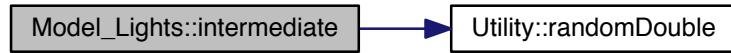


15.48.3.3 intermediate()

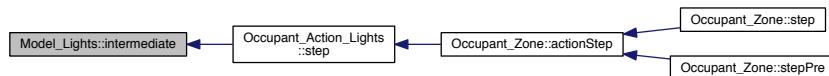
```
bool Model_Lights::intermediate (
    bool state,
    double Lumint )
```

Definition at line 60 of file Model_Lights.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



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

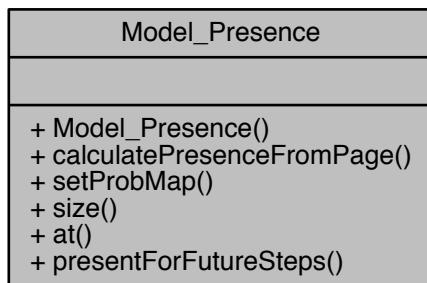
- `Model_Lights.hpp`
- `Model_Lights.cpp`

15.49 Model_Presence Class Reference

Models the Presence of an occupant.

```
#include <Model_Presence.hpp>
```

Collaboration diagram for Model_Presence:



Public Member Functions

- `Model_Presence ()`
- `void calculatePresenceFromPage ()`
- `void setProbMap (const std::map< int, std::string > &probMap)`
- `unsigned int size () const`
- `bool at (const int i) const`
- `int presentForFutureSteps () const`

15.49.1 Detailed Description

Models the Presence of an occupant.

Models the Presence of an occupant adapted from

Page, J., Robinson, D., Morel, N., & Scartezzini, J.-L. (2008). A generalised stochastic model for the simulation of occupant presence. Energy and Buildings, 40(2), 83–98. <https://doi.org/10.1016/j.enbuild.2007.01.018>

Definition at line 16 of file Model_Presence.hpp.

15.49.2 Constructor & Destructor Documentation

15.49.2.1 Model_Presence()

```
Model_Presence::Model_Presence ( )
```

Definition at line 13 of file Model_Presence.cpp.

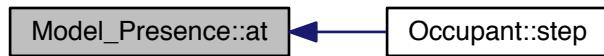
15.49.3 Member Function Documentation

15.49.3.1 at()

```
bool Model_Presence::at ( const int i ) const
```

Definition at line 157 of file Model_Presence.cpp.

Here is the caller graph for this function:

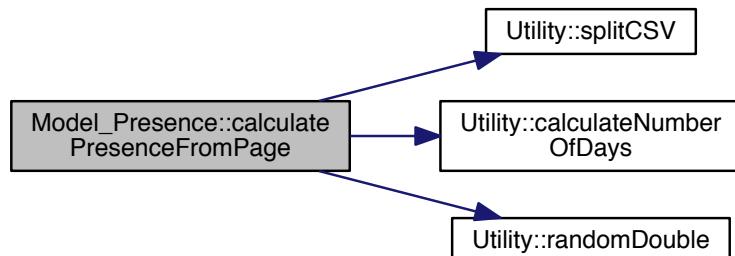


15.49.3.2 calculatePresenceFromPage()

```
void Model_Presence::calculatePresenceFromPage ( )
```

Definition at line 19 of file Model_Presence.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

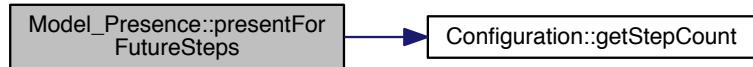


15.49.3.3 presentForFutureSteps()

```
int Model_Presence::presentForFutureSteps ( ) const
```

Definition at line 15 of file `Model_Presence.cpp`.

Here is the call graph for this function:



15.49.3.4 setProbMap()

```
void Model_Presence::setProbMap (const std::map< int, std::string > & probMap )
```

Definition at line 261 of file `Model_Presence.cpp`.

Here is the caller graph for this function:



15.49.3.5 size()

```
unsigned int Model_Presence::size () const
```

Definition at line 161 of file Model_Presence.cpp.

Here is the caller graph for this function:



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

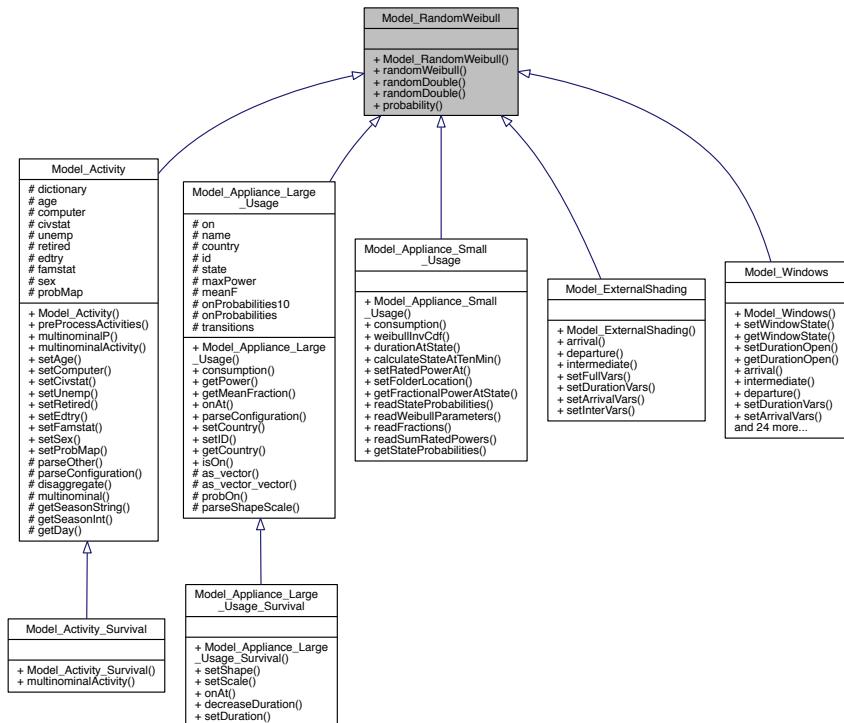
- Model_Presence.hpp
- Model_Presence.cpp

15.50 Model_RandomWeibull Class Reference

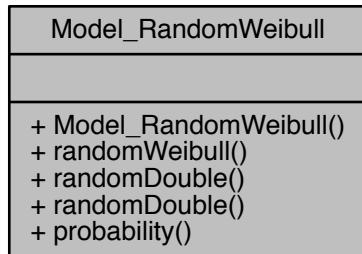
Models a weibull function.

```
#include <Model_RandomWeibull.hpp>
```

Inheritance diagram for Model_RandomWeibull:



Collaboration diagram for Model_RandomWeibull:



Public Member Functions

- [Model_RandomWeibull \(\)](#)

Static Public Member Functions

- static double [randomWeibull](#) (double scale, double shape)
- static double [randomDouble](#) ()
- static double [randomDouble](#) (double min, double max)
- static double [probability](#) (double m)

15.50.1 Detailed Description

Models a weibull function.

Models a weibull function

Definition at line 11 of file Model_RandomWeibull.hpp.

15.50.2 Constructor & Destructor Documentation

15.50.2.1 Model_RandomWeibull()

```
Model_RandomWeibull::Model_RandomWeibull ( )
```

Definition at line 8 of file Model_RandomWeibull.cpp.

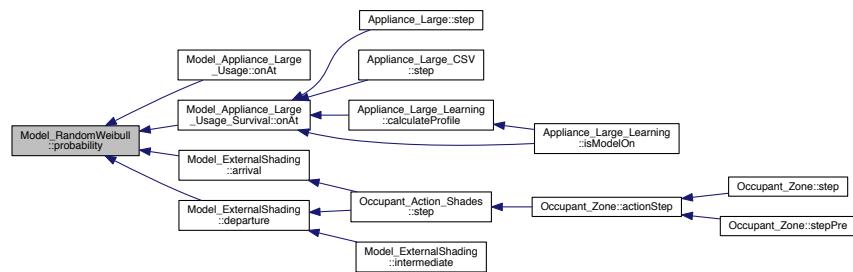
15.50.3 Member Function Documentation

15.50.3.1 probability()

```
double Model_RandomWeibull::probability (
    double m ) [static]
```

Definition at line 29 of file Model_RandomWeibull.cpp.

Here is the caller graph for this function:

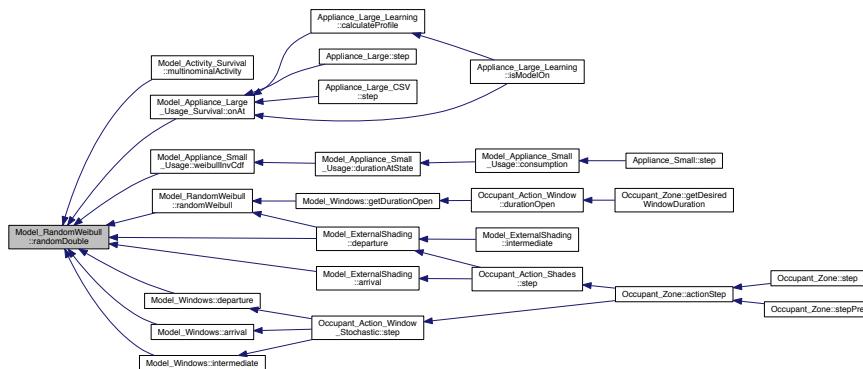


15.50.3.2 randomDouble() [1/2]

```
double Model_RandomWeibull::randomDouble () [static]
```

Definition at line 21 of file Model_RandomWeibull.cpp.

Here is the caller graph for this function:



15.50.3.3 randomDouble() [2/2]

```
double Model_RandomWeibull::randomDouble (
    double min,
    double max ) [static]
```

Definition at line 25 of file Model_RandomWeibull.cpp.

Here is the call graph for this function:



15.50.3.4 randomWeibull()

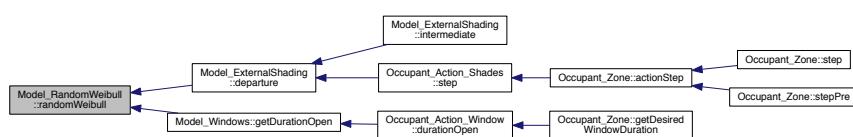
```
double Model_RandomWeibull::randomWeibull (
    double scale,
    double shape ) [static]
```

Definition at line 10 of file Model_RandomWeibull.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



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

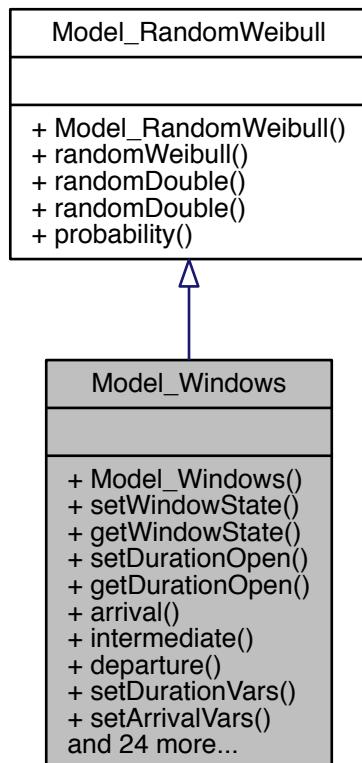
- Model_RandomWeibull.hpp
- Model_RandomWeibull.cpp

15.51 Model_Windows Class Reference

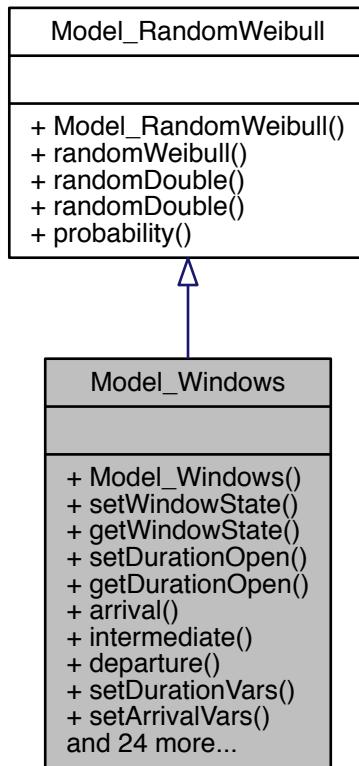
Models occupant window openings.

```
#include <Model_Windows.hpp>
```

Inheritance diagram for Model_Windows:



Collaboration diagram for Model_Windows:



Public Member Functions

- [Model_Windows \(\)](#)
- void [setWindowState](#) (bool windowState)
- bool [getWindowState](#) () const
- void [setDurationOpen](#) (int durationOpen)
- int [getDurationOpen](#) () const
- void [arrival](#) (double indoorTemperature, double outdoorTemperature, double previousDuration, bool rain, int timeStepLengthInMinutes)
- void [intermediate](#) (double indoorTemperature, double outdoorTemperature, double currentDuration, bool rain, int timeStepLengthInMinutes)
- void [departure](#) (double indoorTemperature, double dailyMeanTemperature, double futureDuration, double groundFloor)
- void [setDurationVars](#) (double aop, double bopout, double shapeop)
- void [setArrivalVars](#) (double a01arr, double b01inarr, double b01outarr, double b01absprevarr, double b01rnarr)
- void [setInterVars](#) (double a01int, double b01inint, double b01outint, double b01presint, double b01rnint)
- void [setDepartureVars](#) (double a01dep, double b01outdep, double b01absdep, double b01gddep, double a10dep, double b10indep, double b10outdep, double b10absdep, double b10gddep)
- double [getAop](#) () const
- double [getBopout](#) () const

- double `getshapeop () const`
- double `getA01arr () const`
- double `getB01inarr () const`
- double `getB01outarr () const`
- double `getB01absprevarr () const`
- double `getB01rnarr () const`
- double `getA01int () const`
- double `getB01inint () const`
- double `getB01outint () const`
- double `getB01presint () const`
- double `getB01rnint () const`
- double `getA01dep () const`
- double `getB01outdep () const`
- double `getB01absdep () const`
- double `getB01gddep () const`
- double `getA10dep () const`
- double `getB10indep () const`
- double `getB10outdep () const`
- double `getB10absdep () const`
- double `getB10gddep () const`

Additional Inherited Members

15.51.1 Detailed Description

Models occupant window openings.

Models occupant window openings adapted from

Haldi, F., & Robinson, D. (2009). Interactions with window openings by office occupants. *Building and Environment*, 44(12), 2378–2395. <https://doi.org/10.1016/j.buildenv.2009.03.025>

Definition at line 13 of file `Model_Windows.hpp`.

15.51.2 Constructor & Destructor Documentation

15.51.2.1 Model_Windows()

```
Model_Windows::Model_Windows ( )
```

Definition at line 7 of file `Model_Windows.cpp`.

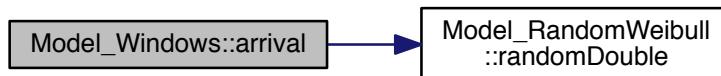
15.51.3 Member Function Documentation

15.51.3.1 arrival()

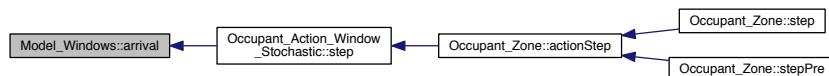
```
void Model_Windows::arrival (
    double indoorTemperature,
    double outdoorTemperature,
    double previousDuration,
    bool rain,
    int timeStepLengthInMinutes )
```

Definition at line 97 of file Model_Windows.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

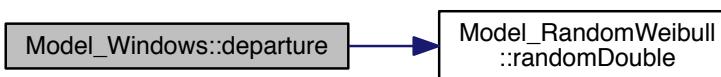


15.51.3.2 departure()

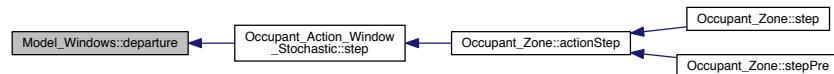
```
void Model_Windows::departure (
    double indoorTemperature,
    double dailyMeanTemperature,
    double futureDuration,
    double groundFloor )
```

Definition at line 159 of file Model_Windows.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.51.3.3 getA01arr()

```
double Model_Windows::getA01arr ( ) const
```

Definition at line 208 of file Model_Windows.cpp.

15.51.3.4 getA01dep()

```
double Model_Windows::getA01dep ( ) const
```

Definition at line 248 of file Model_Windows.cpp.

15.51.3.5 getA01int()

```
double Model_Windows::getA01int ( ) const
```

Definition at line 228 of file Model_Windows.cpp.

15.51.3.6 getA10dep()

```
double Model_Windows::getA10dep ( ) const
```

Definition at line 264 of file Model_Windows.cpp.

15.51.3.7 getAop()

```
double Model_Windows::getAop ( ) const
```

Definition at line 196 of file Model_Windows.cpp.

15.51.3.8 getB01absdep()

```
double Model_Windows::getB01absdep ( ) const
```

Definition at line 256 of file Model_Windows.cpp.

15.51.3.9 getB01absprevarr()

```
double Model_Windows::getB01absprevarr ( ) const
```

Definition at line 220 of file Model_Windows.cpp.

15.51.3.10 getB01gddep()

```
double Model_Windows::getB01gddep ( ) const
```

Definition at line 260 of file Model_Windows.cpp.

15.51.3.11 getB01inarr()

```
double Model_Windows::getB01inarr ( ) const
```

Definition at line 212 of file Model_Windows.cpp.

15.51.3.12 getB01inint()

```
double Model_Windows::getB01inint ( ) const
```

Definition at line 232 of file Model_Windows.cpp.

15.51.3.13 getB01outarr()

```
double Model_Windows::getB01outarr ( ) const
```

Definition at line 216 of file Model_Windows.cpp.

15.51.3.14 getB01outdep()

```
double Model_Windows::getB01outdep ( ) const
```

Definition at line 252 of file Model_Windows.cpp.

15.51.3.15 getB01outint()

```
double Model_Windows::getB01outint ( ) const
```

Definition at line 236 of file Model_Windows.cpp.

15.51.3.16 getB01presint()

```
double Model_Windows::getB01presint ( ) const
```

Definition at line 240 of file Model_Windows.cpp.

15.51.3.17 getB01rnarr()

```
double Model_Windows::getB01rnarr ( ) const
```

Definition at line 224 of file Model_Windows.cpp.

15.51.3.18 getB01rnint()

```
double Model_Windows::getB01rnint ( ) const
```

Definition at line 244 of file Model_Windows.cpp.

15.51.3.19 getB10absdep()

```
double Model_Windows::getB10absdep ( ) const
```

Definition at line 276 of file Model_Windows.cpp.

15.51.3.20 getB10gddep()

```
double Model_Windows::getB10gddep ( ) const
```

Definition at line 280 of file Model_Windows.cpp.

15.51.3.21 getB10indep()

```
double Model_Windows::getB10indep ( ) const
```

Definition at line 268 of file Model_Windows.cpp.

15.51.3.22 getB10outdep()

```
double Model_Windows::getB10outdep ( ) const
```

Definition at line 272 of file Model_Windows.cpp.

15.51.3.23 getBopout()

```
double Model_Windows::getBopout ( ) const
```

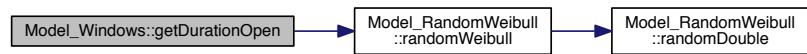
Definition at line 200 of file Model_Windows.cpp.

15.51.3.24 getDurationOpen()

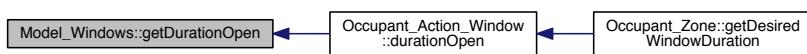
```
int Model_Windows::getDurationOpen ( ) const
```

Definition at line 89 of file Model_Windows.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.51.3.25 getshapeop()

```
double Model_Windows::getshapeop ( ) const
```

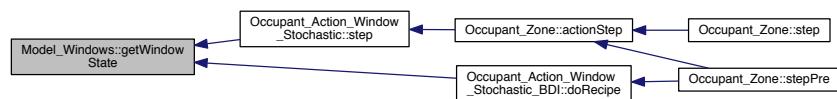
Definition at line 204 of file Model_Windows.cpp.

15.51.3.26 getWindowState()

```
bool Model_Windows::getWindowState ( ) const
```

Definition at line 81 of file Model_Windows.cpp.

Here is the caller graph for this function:

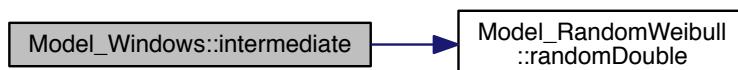


15.51.3.27 intermediate()

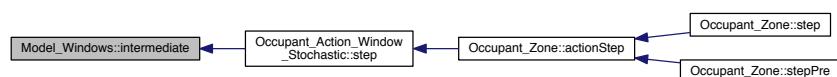
```
void Model_Windows::intermediate (
    double indoorTemperature,
    double outdoorTemperature,
    double currentDuration,
    bool rain,
    int timeStepLengthInMinutes )
```

Definition at line 129 of file Model_Windows.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.51.3.28 setArrivalVars()

```
void Model_Windows::setArrivalVars (
    double a01arr,
    double b01inarr,
    double b01outarr,
    double b01absprevarr,
    double b01rnarr )
```

Definition at line 45 of file Model_Windows.cpp.

Here is the caller graph for this function:



15.51.3.29 setDepartureVars()

```
void Model_Windows::setDepartureVars (
    double a01dep,
    double b01outdep,
    double b01absdep,
    double b01gddep,
    double a10dep,
    double b10indep,
    double b10outdep,
    double b10absdep,
    double b10gddep )
```

Definition at line 63 of file Model_Windows.cpp.

Here is the caller graph for this function:

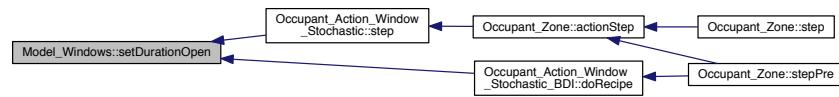


15.51.3.30 setDurationOpen()

```
void Model_Windows::setDurationOpen (
    int durationOpen )
```

Definition at line 85 of file Model_Windows.cpp.

Here is the caller graph for this function:



15.51.3.31 setDurationVars()

```
void Model_Windows::setDurationVars (
    double aop,
    double bopout,
    double shapeop )
```

Definition at line 39 of file Model_Windows.cpp.

Here is the caller graph for this function:



15.51.3.32 setInterVars()

```
void Model_Windows::setInterVars (
    double a01int,
    double b01inint,
    double b01outint,
    double b01presint,
    double b01rnint )
```

Definition at line 54 of file Model_Windows.cpp.

Here is the caller graph for this function:

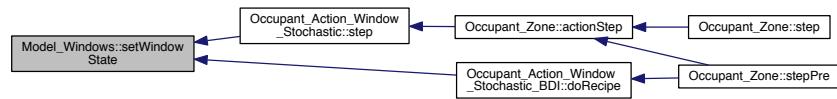


15.51.3.33 setWindowState()

```
void Model_Windows::setWindowState (
    bool windowState )
```

Definition at line 77 of file Model_Windows.cpp.

Here is the caller graph for this function:



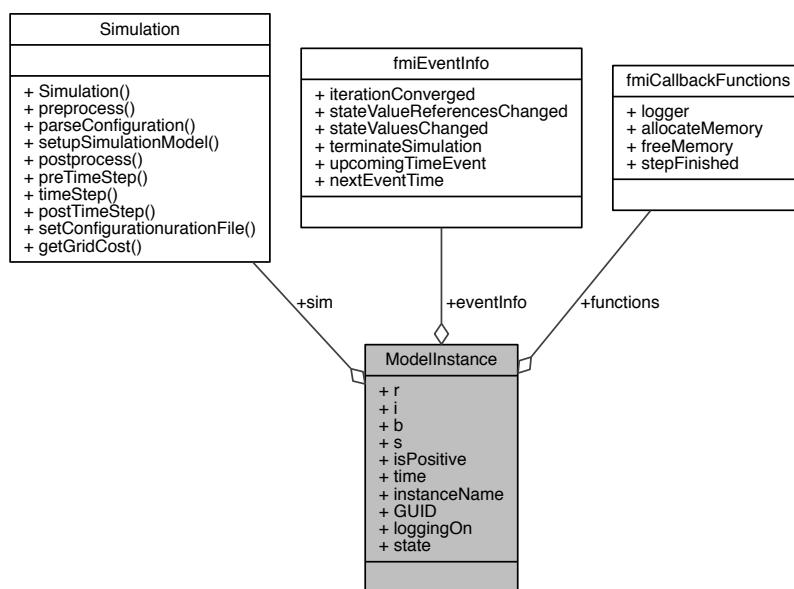
The documentation for this class was generated from the following files:

- Model_Windows.hpp
- Model_Windows.cpp

15.52 ModellInstance Struct Reference

```
#include <fmuTemplate.hpp>
```

Collaboration diagram for ModellInstance:



Public Attributes

- fmiReal * **r**
- fmiInteger * **i**
- fmiBoolean * **b**
- fmiString * **s**
- fmiBoolean * **isPositive**
- fmiReal **time**
- fmiString **instanceName**
- fmiString **GUID**
- **fmiCallbackFunctions** **functions**
- fmiBoolean **loggingOn**
- ModelState **state**
- **Simulation** **sim**
- **fmiEventInfo** **eventInfo**

15.52.1 Detailed Description

Definition at line 37 of file fmuTemplate.hpp.

15.52.2 Member Data Documentation

15.52.2.1 **b**

`fmiBoolean* ModelInstance::b`

Definition at line 40 of file fmuTemplate.hpp.

15.52.2.2 **eventInfo**

`fmiEventInfo ModelInstance::eventInfo`

Definition at line 50 of file fmuTemplate.hpp.

15.52.2.3 **functions**

`fmiCallbackFunctions ModelInstance::functions`

Definition at line 46 of file fmuTemplate.hpp.

15.52.2.4 GUID

```
fmiString ModelInstance::GUID
```

Definition at line 45 of file fmuTemplate.hpp.

15.52.2.5 i

```
fmiInteger* ModelInstance::i
```

Definition at line 39 of file fmuTemplate.hpp.

15.52.2.6 instanceName

```
fmiString ModelInstance::instanceName
```

Definition at line 44 of file fmuTemplate.hpp.

15.52.2.7 isPositive

```
fmiBoolean* ModelInstance::isPositive
```

Definition at line 42 of file fmuTemplate.hpp.

15.52.2.8 loggingOn

```
fmiBoolean ModelInstance::loggingOn
```

Definition at line 47 of file fmuTemplate.hpp.

15.52.2.9 r

```
fmiReal* ModelInstance::r
```

Definition at line 38 of file fmuTemplate.hpp.

15.52.2.10 s

```
fmiString* ModelInstance::s
```

Definition at line 41 of file fmuTemplate.hpp.

15.52.2.11 sim

```
Simulation ModelInstance::sim
```

Definition at line 49 of file fmuTemplate.hpp.

15.52.2.12 state

```
ModelState ModelInstance::state
```

Definition at line 48 of file fmuTemplate.hpp.

15.52.2.13 time

```
fmiReal ModelInstance::time
```

Definition at line 43 of file fmuTemplate.hpp.

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

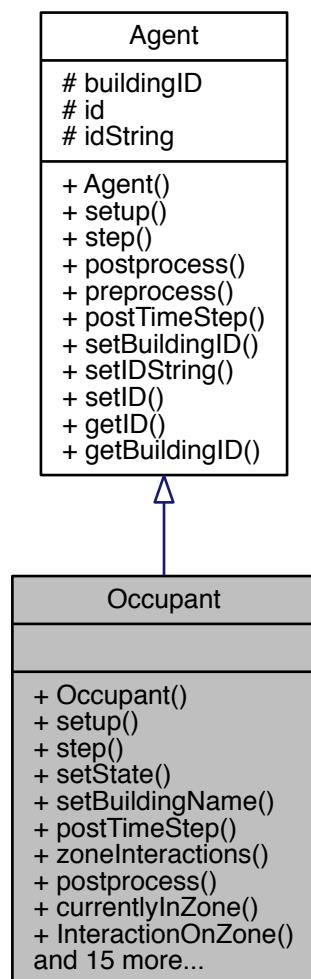
- fmuTemplate.hpp

15.53 Occupant Class Reference

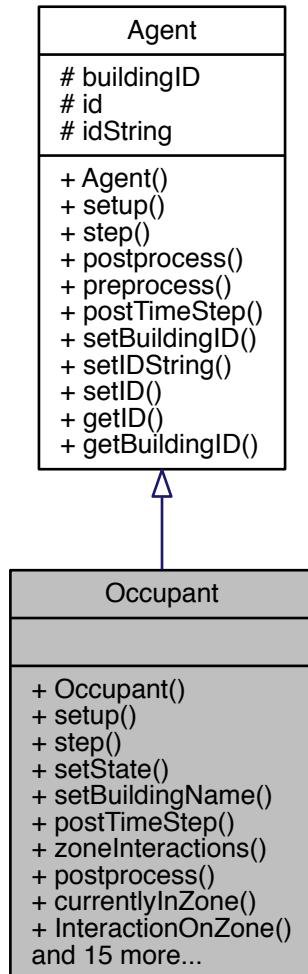
The occupant agent.

```
#include <Occupant.hpp>
```

Inheritance diagram for Occupant:



Collaboration diagram for Occupant:



Public Member Functions

- [Occupant \(\)](#)
- void [setup](#) (int `id`, const `ConfigStructAgent` &`agent`, const std::vector< std::shared_ptr< `Building_Zone` >> &`zones`)
Initialises the occupant.
- void [step \(\)](#)
the occupant timestep function moves the agent to their new state, calls the agents understanding of zone that an agent will be interacting with.
 - void [setState](#) (const `State` &`state`)
 - void [setBuildingName](#) (const std::string &`buildingName`)
 - void [postTimeStep \(\)](#)
 - void [zoneInteractions \(\)](#)
 - void [postprocess \(\)](#)
 - bool [currentlyInZone](#) (const `Building_Zone` &`zone`) const

- bool `InteractionOnZone` (const `Building_Zone` &zone) const
- bool `getDesiredLightState` (const `Building_Zone` &zone) const
- bool `getDesiredWindowState` (const `Building_Zone` &zone) const
- bool `isActionWindow` (const `Building_Zone` &zone) const
- bool `isActionLights` (const `Building_Zone` &zone) const
- bool `isActionShades` (const `Building_Zone` &zone) const
- bool `isActionHeatGains` (const `Building_Zone` &zone) const
- bool `isActionLearning` (const `Building_Zone` &zone) const
- bool `isActionAppliance` (const `Building_Zone` &zone) const
- bool `previouslyInZone` (const `Building_Zone` &zone) const
- int `getStateID` () const
- double `getDesiredShadeState` (const `Building_Zone` &zone) const
- double `getDesiredAppliance` (const `Building_Zone` &zone) const
- double `getDesiredHeatState` (const `Building_Zone` &zone) const
- double `getCurrentRadientGains` (const `Building_Zone` &zone) const
- double `getPower` () const

Additional Inherited Members

15.53.1 Detailed Description

The occupant agent.

The occupant agent

Definition at line 21 of file Occupant.hpp.

15.53.2 Constructor & Destructor Documentation

15.53.2.1 Occupant()

```
Occupant::Occupant ( )
```

Definition at line 23 of file Occupant.cpp.

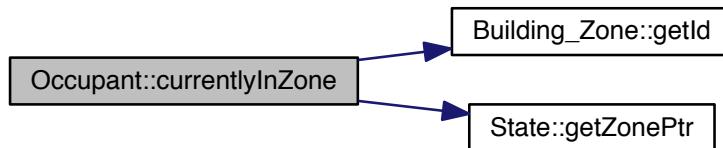
15.53.3 Member Function Documentation

15.53.3.1 currentlyInZone()

```
bool Occupant::currentlyInZone (
    const Building_Zone & zone ) const
```

Definition at line 291 of file Occupant.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.53.3.2 getCurrentRadientGains()

```
double Occupant::getCurrentRadientGains (
    const Building_Zone & zone ) const
```

Definition at line 210 of file Occupant.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.53.3.3 getDesiredAppliance()

```
double Occupant::getDesiredAppliance (
    const Building_Zone & zone ) const
```

Definition at line 258 of file Occupant.cpp.

Here is the call graph for this function:



15.53.3.4 getDesiredHeatState()

```
double Occupant::getDesiredHeatState (
    const Building_Zone & zone ) const
```

Definition at line 280 of file Occupant.cpp.

Here is the call graph for this function:

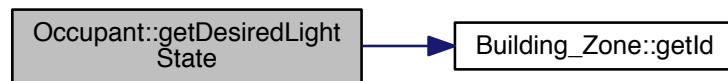


15.53.3.5 getDesiredLightState()

```
bool Occupant::getDesiredLightState (
    const Building_Zone & zone ) const
```

Definition at line 225 of file Occupant.cpp.

Here is the call graph for this function:

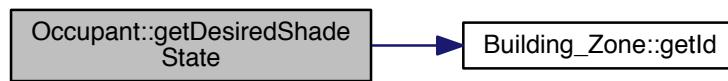


15.53.3.6 getDesiredShadeState()

```
double Occupant::getDesiredShadeState (
    const Building_Zone & zone ) const
```

Definition at line 247 of file Occupant.cpp.

Here is the call graph for this function:

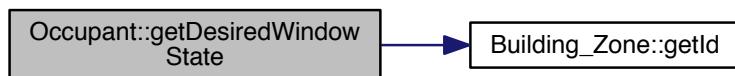


15.53.3.7 getDesiredWindowState()

```
bool Occupant::getDesiredWindowState (
    const Building_Zone & zone ) const
```

Definition at line 236 of file Occupant.cpp.

Here is the call graph for this function:



15.53.3.8 getPower()

```
double Occupant::getPower ( ) const
```

Definition at line 221 of file Occupant.cpp.

15.53.3.9 getStateID()

```
int Occupant::getStateID ( ) const
```

Definition at line 386 of file Occupant.cpp.

Here is the call graph for this function:

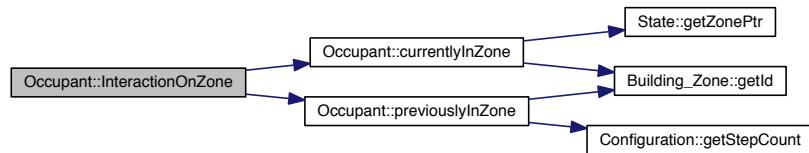


15.53.3.10 InteractionOnZone()

```
bool Occupant::InteractionOnZone (
    const Building_Zone & zone ) const
```

Definition at line 303 of file Occupant.cpp.

Here is the call graph for this function:



15.53.3.11 isActionAppliance()

```
bool Occupant::isActionAppliance (
    const Building_Zone & zone ) const
```

Definition at line 369 of file Occupant.cpp.

Here is the call graph for this function:



15.53.3.12 isActionHeatGains()

```
bool Occupant::isActionHeatGains (
    const Building_Zone & zone ) const
```

Definition at line 348 of file Occupant.cpp.

Here is the call graph for this function:



15.53.3.13 isActionLearning()

```
bool Occupant::isActionLearning (
    const Building_Zone & zone ) const
```

Definition at line 358 of file Occupant.cpp.

Here is the call graph for this function:



15.53.3.14 isActionLights()

```
bool Occupant::isActionLights (
    const Building_Zone & zone ) const
```

Definition at line 328 of file Occupant.cpp.

Here is the call graph for this function:



15.53.3.15 isActionShades()

```
bool Occupant::isActionShades (
    const Building_Zone & zone ) const
```

Definition at line 338 of file Occupant.cpp.

Here is the call graph for this function:



15.53.3.16 isActionWindow()

```
bool Occupant::isActionWindow (
    const Building_Zone & zone ) const
```

Definition at line 317 of file Occupant.cpp.

Here is the call graph for this function:



15.53.3.17 postprocess()

```
void Occupant::postprocess ( )
```

Definition at line 307 of file Occupant.cpp.

15.53.3.18 postTimeStep()

```
void Occupant::postTimeStep ( )
```

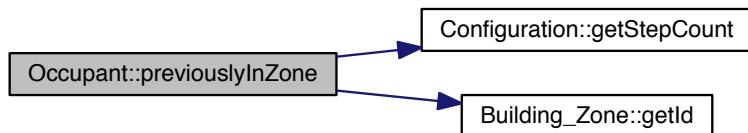
Definition at line 380 of file Occupant.cpp.

15.53.3.19 previouslyInZone()

```
bool Occupant::previouslyInZone ( const Building_Zone & zone ) const
```

Definition at line 295 of file Occupant.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.53.3.20 setBuildingName()

```
void Occupant::setBuildingName ( const std::string & buildingName )
```

Definition at line 390 of file Occupant.cpp.

15.53.3.21 setState()

```
void Occupant::setState (
    const State & state )
```

Definition at line 313 of file Occupant.cpp.

Here is the caller graph for this function:



15.53.3.22 setup()

```
void Occupant::setup (
    int id,
    const ConfigStructAgent & agent,
    const std::vector< std::shared_ptr< Building_Zone >> & zones )
```

Initialises the occupant.

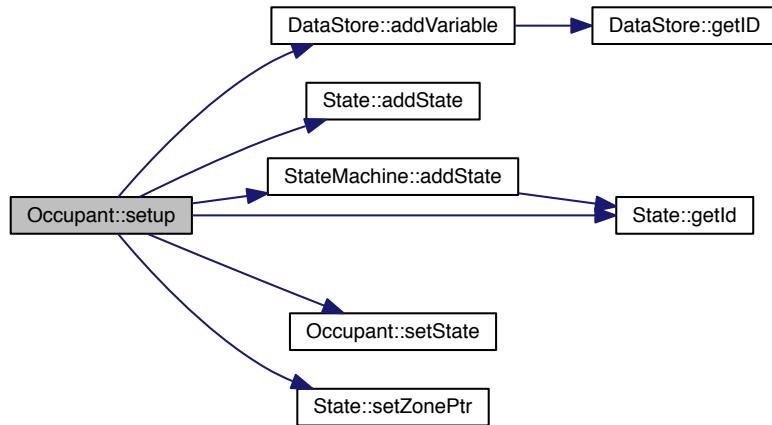
Set occupant parameters, preprocesses the activities or states and sets up the initial state of the occupant

Parameters

<i>id</i>	The occupants id
<i>agent</i>	The configuration struct, built from config file
<i>zones</i>	Zones that the agent can inhabit

Definition at line 33 of file Occupant.cpp.

Here is the call graph for this function:



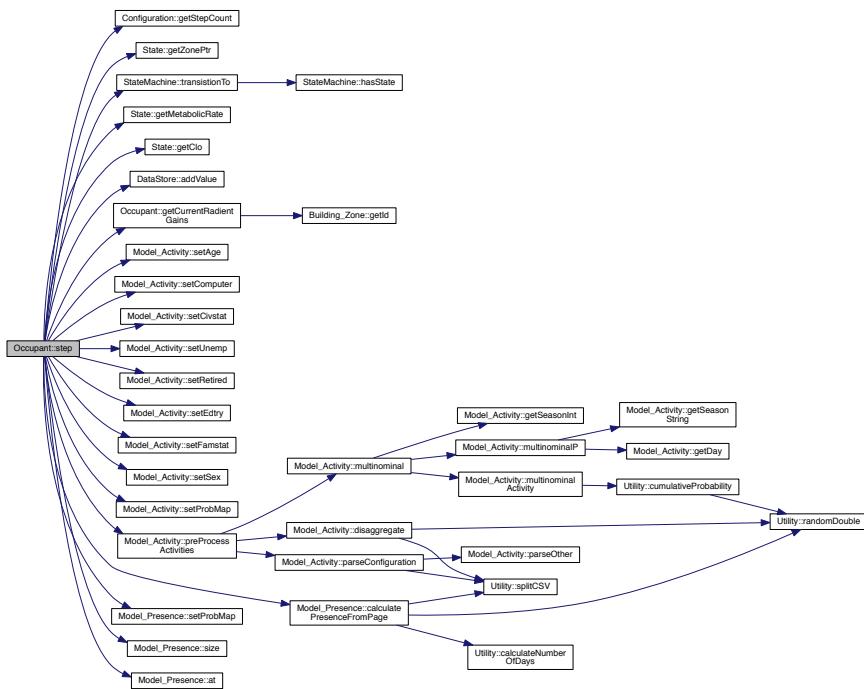
15.53.3.23 step()

```
void Occupant::step ()
```

the occupant timestep function moves the agent to their new state, calls the agents understanding of zone that an agent will be interacting with.

Definition at line 150 of file Occupant.cpp.

Here is the call graph for this function:



15.53.3.24 zoneInteractions()

```
void Occupant::zoneInteractions ( )
```

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

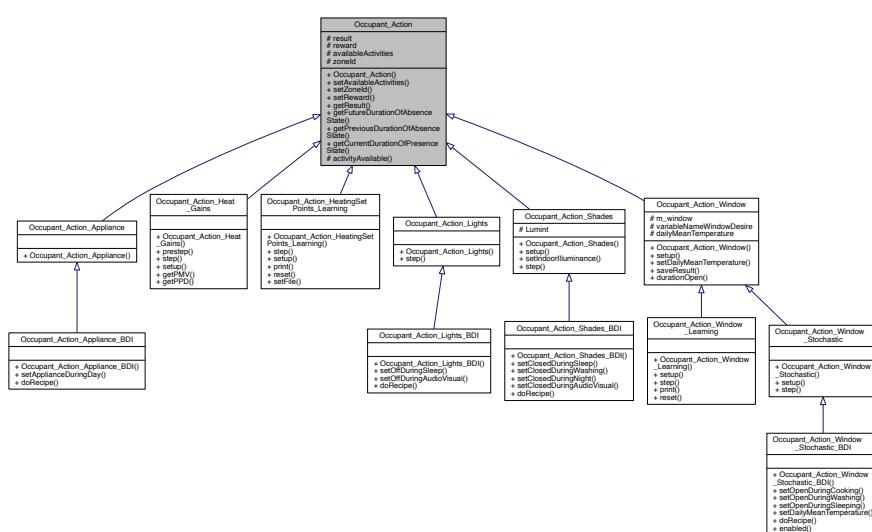
- Occupant.hpp
- Occupant.cpp

15.54 Occupant_Action Class Reference

Occupant action super class.

```
#include <Occupant_Action.hpp>
```

Inheritance diagram for Occupant_Action:



Collaboration diagram for Occupant_Action:

Occupant_Action
<pre># result # reward # availableActivities # zoneld + Occupant_Action() + setAvailableActivities() + setZoneld() + setReward() + getResult() + getFutureDurationOfAbsence State() + getPreviousDurationOfAbsence State() + getCurrentDurationOfPresence State() # activityAvailable()</pre>

Public Member Functions

- [Occupant_Action \(\)](#)
- void [setAvailableActivities \(const std::vector< int > &availableActivities\)](#)
- void [setZoneld \(const double zoneld\)](#)
- void [setReward \(const double reward\)](#)
- double [getResult \(\) const](#)
- double [getFutureDurationOfAbsenceState \(const std::vector< double > &activities\) const](#)
- double [getPreviousDurationOfAbsenceState \(const std::vector< double > &activities\) const](#)
- double [getCurrentDurationOfPresenceState \(const std::vector< double > &activities\) const](#)

Protected Member Functions

- bool [activityAvailable \(const int act\) const](#)

Protected Attributes

- double [result](#)
- double [reward](#)
- std::vector< int > [availableActivities](#)
- int [zoneld](#)

15.54.1 Detailed Description

Occupant action super class.

Occupant action super class

Definition at line 14 of file Occupant_Action.hpp.

15.54.2 Constructor & Destructor Documentation

15.54.2.1 Occupant_Action()

```
Occupant_Action::Occupant_Action ( )
```

Definition at line 9 of file Occupant_Action.cpp.

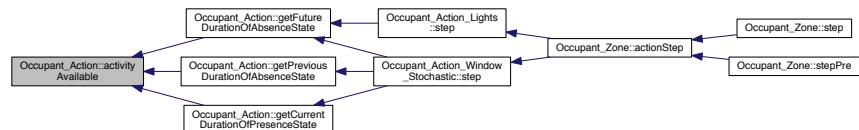
15.54.3 Member Function Documentation

15.54.3.1 activityAvailable()

```
bool Occupant_Action::activityAvailable (
    const int act ) const [protected]
```

Definition at line 20 of file Occupant_Action.cpp.

Here is the caller graph for this function:

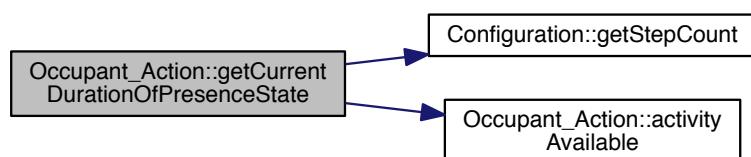


15.54.3.2 getCurrentDurationOfPresenceState()

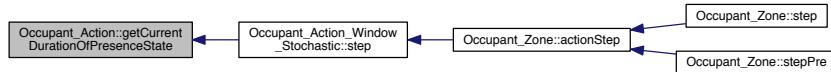
```
double Occupant_Action::getCurrentDurationOfPresenceState (
    const std::vector< double > & activities ) const
```

Definition at line 57 of file Occupant_Action.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

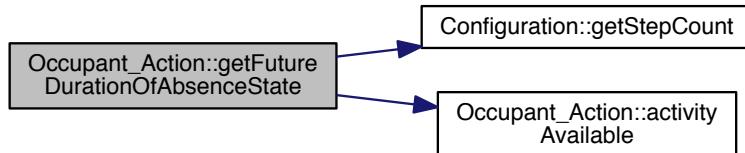


15.54.3.3 getFutureDurationOfAbsenceState()

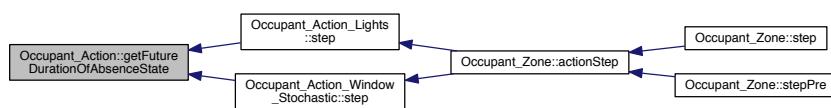
```
double Occupant_Action::getFutureDurationOfAbsenceState (
    const std::vector< double > & activities ) const
```

Definition at line 25 of file Occupant_Action.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

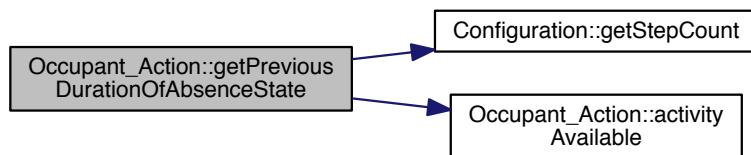


15.54.3.4 getPreviousDurationOfAbsenceState()

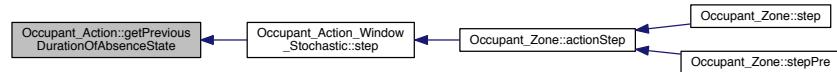
```
double Occupant_Action::getPreviousDurationOfAbsenceState (
    const std::vector< double > & activities ) const
```

Definition at line 43 of file Occupant_Action.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.54.3.5 getResult()

```
double Occupant_Action::getResult( ) const
```

Definition at line 11 of file Occupant_Action.cpp.

Here is the caller graph for this function:

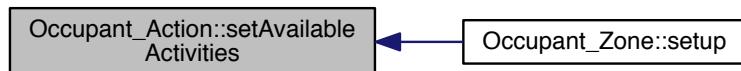


15.54.3.6 setAvailableActivities()

```
void Occupant_Action::setAvailableActivities (
    const std::vector< int > & availableActivities )
```

Definition at line 15 of file Occupant_Action.cpp.

Here is the caller graph for this function:

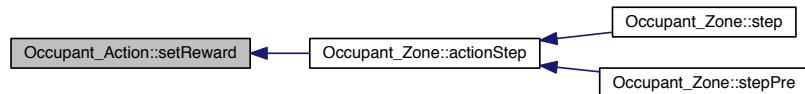


15.54.3.7 setReward()

```
void Occupant_Action::setReward (
    const double reward )
```

Definition at line 74 of file Occupant_Action.cpp.

Here is the caller graph for this function:



15.54.3.8 setZoneId()

```
void Occupant_Action::setZoneId (
    const double zoneId )
```

Definition at line 78 of file Occupant_Action.cpp.

Here is the caller graph for this function:



15.54.4 Member Data Documentation

15.54.4.1 availableActivities

```
std::vector<int> Occupant_Action::availableActivities [protected]
```

Definition at line 33 of file Occupant_Action.hpp.

15.54.4.2 result

```
double Occupant_Action::result [protected]
```

Definition at line 31 of file Occupant_Action.hpp.

15.54.4.3 reward

```
double Occupant_Action::reward [protected]
```

Definition at line 32 of file Occupant_Action.hpp.

15.54.4.4 zoneId

```
int Occupant_Action::zoneId [protected]
```

Definition at line 34 of file Occupant_Action.hpp.

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

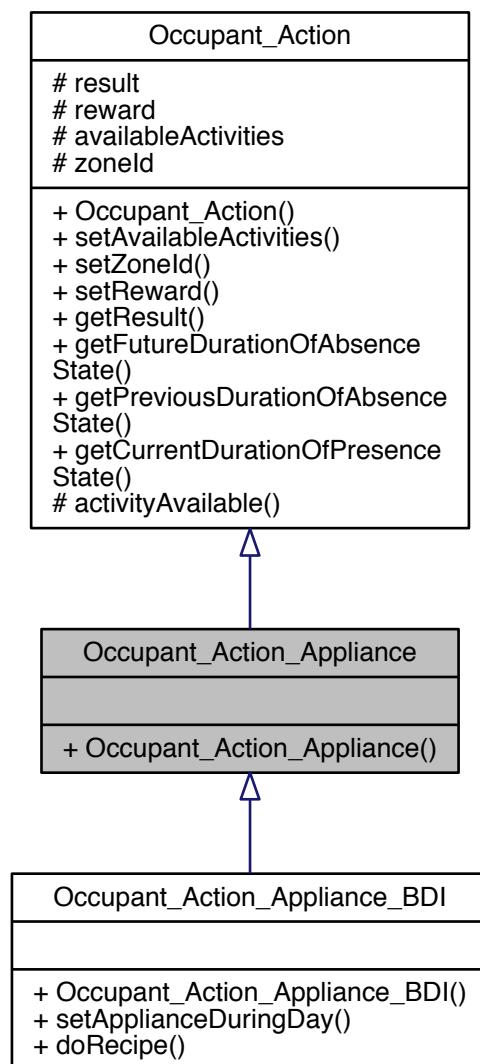
- Occupant_Action.hpp
- Occupant_Action.cpp

15.55 Occupant_Action_Appliance Class Reference

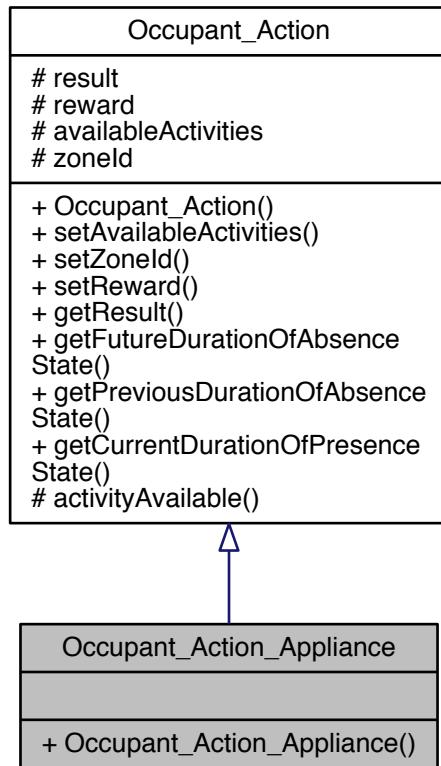
Occupant action on appliances.

```
#include <Occupant_Action_Appliance.hpp>
```

Inheritance diagram for Occupant_Action_Appliance:



Collaboration diagram for Occupant_Action_Appliance:



Public Member Functions

- [Occupant_Action_Appliance \(\)](#)

Additional Inherited Members

15.55.1 Detailed Description

Occupant action on appliances.

Occupant action on appliances

Definition at line 13 of file `Occupant_Action_Appliance.hpp`.

15.55.2 Constructor & Destructor Documentation

15.55.2.1 Occupant_Action_Appliance()

`Occupant_Action_Appliance::Occupant_Action_Appliance ()`

Definition at line 5 of file `Occupant_Action_Appliance.cpp`.

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

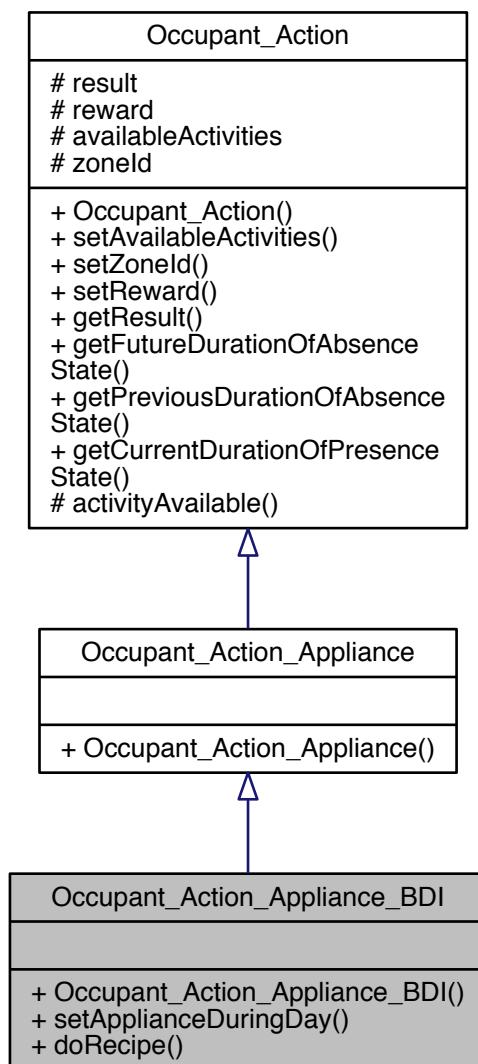
- `Occupant_Action_Appliance.hpp`
- `Occupant_Action_Appliance.cpp`

15.56 Occupant_Action_Appliance_BDI Class Reference

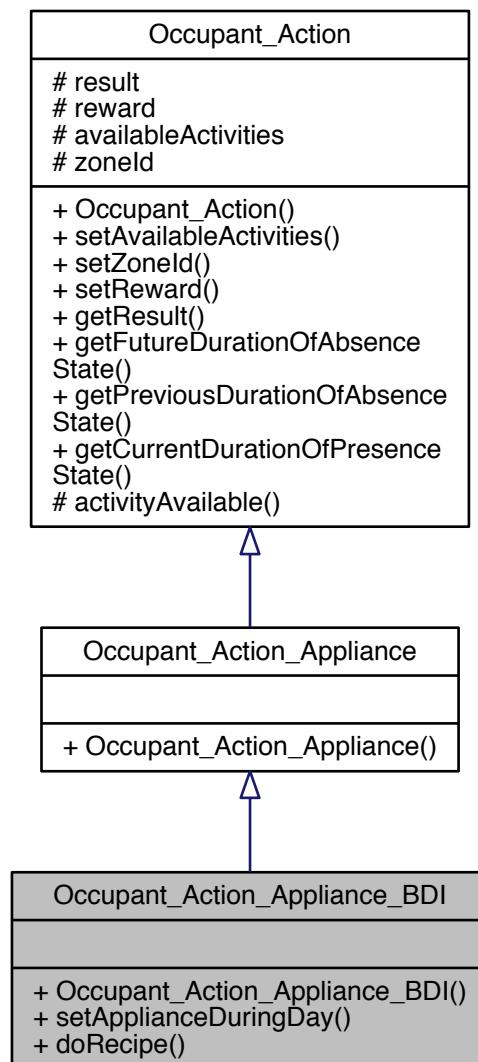
Occupant action on appliances using BDI.

```
#include <Occupant_Action_Appliance_BDI.hpp>
```

Inheritance diagram for `Occupant_Action_Appliance_BDI`:



Collaboration diagram for Occupant_Action_Appliance_BDI:



Public Member Functions

- [Occupant_Action_Appliance_BDI \(\)](#)
- void [setApplianceDuringDay](#) (double ApplianceDuringDay)
- bool [doRecipe](#) (const std::vector<double> &activities)

Additional Inherited Members

15.56.1 Detailed Description

Occupant action on appliances using BDI.

Occupant action on appliances using BDI adapted from Chapman, J., Siebers, P., & Robinson, D. (2017). Data Scarce Behavioural Modelling and the Representation of Social Interactions. Unpublished Manuscript, 1–48.

Definition at line 14 of file Occupant_Action_Appliance_BDI.hpp.

15.56.2 Constructor & Destructor Documentation

15.56.2.1 Occupant_Action_Appliance_BDI()

```
Occupant_Action_Appliance_BDI::Occupant_Action_Appliance_BDI ( )
```

Definition at line 7 of file Occupant_Action_Appliance_BDI.cpp.

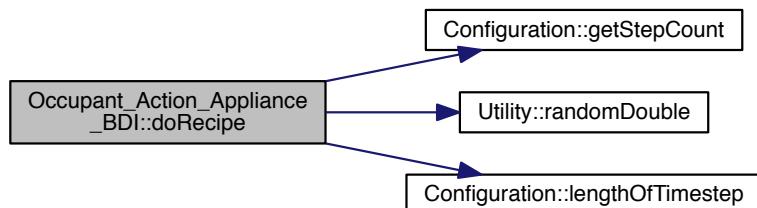
15.56.3 Member Function Documentation

15.56.3.1 doRecipe()

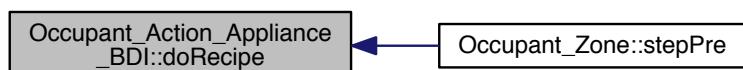
```
bool Occupant_Action_Appliance_BDI::doRecipe (
    const std::vector< double > & activities )
```

Definition at line 16 of file Occupant_Action_Appliance_BDI.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

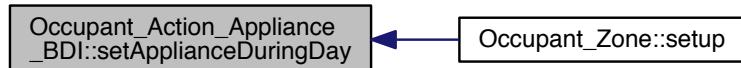


15.56.3.2 setApplianceDuringDay()

```
void Occupant_Action_Appliance_BDI::setApplianceDuringDay ( double ApplianceDuringDay )
```

Definition at line 11 of file Occupant_Action_Appliance_BDI.cpp.

Here is the caller graph for this function:



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

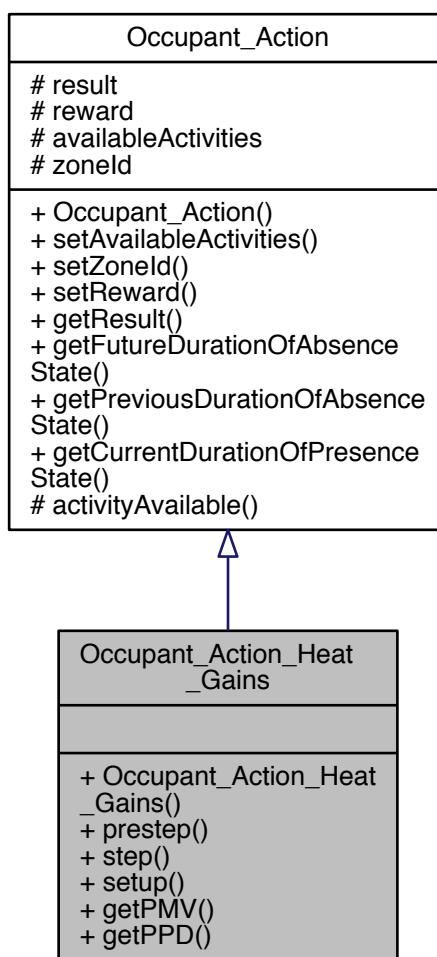
- Occupant_Action_Appliance_BDI.hpp
- Occupant_Action_Appliance_BDI.cpp

15.57 Occupant_Action_Heat_Gains Class Reference

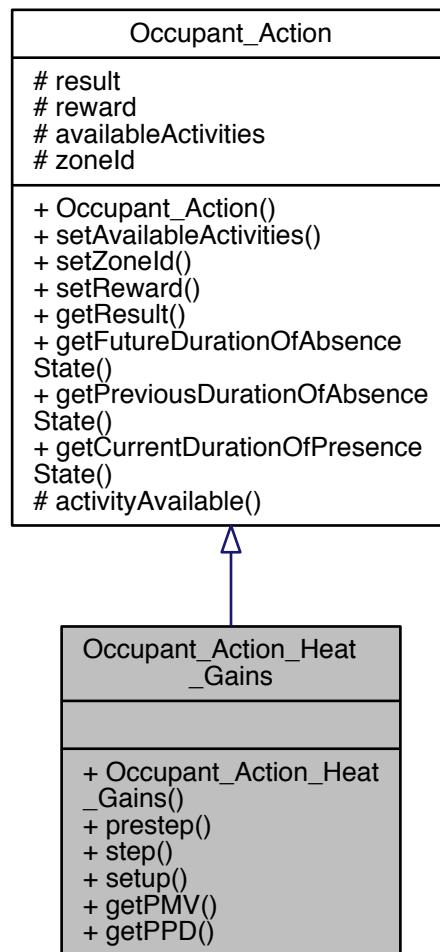
Occupant action of heat gains.

```
#include <Occupant_Action_Heat_Gains.hpp>
```

Inheritance diagram for Occupant_Action_Heat_Gains:



Collaboration diagram for Occupant_Action_Heat_Gains:



Public Member Functions

- `Occupant_Action_Heat_Gains ()`
- void `prestep` (double clo, double metabolicRate)
- void `step` (const `Building_Zone` &zone, const bool inZone)
- void `setup` (int buildingID, int agentid)
- double `getPMV () const`
- double `getPPD () const`

Additional Inherited Members

15.57.1 Detailed Description

Occupant action of heat gains.

Occupant action of heat gains

Definition at line 14 of file `Occupant_Action_Heat_Gains.hpp`.

15.57.2 Constructor & Destructor Documentation

15.57.2.1 Occupant_Action_Heat_Gains()

```
Occupant_Action_Heat_Gains::Occupant_Action_Heat_Gains ( )
```

Definition at line 10 of file Occupant_Action_Heat_Gains.cpp.

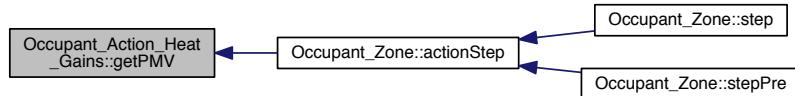
15.57.3 Member Function Documentation

15.57.3.1 getPMV()

```
double Occupant_Action_Heat_Gains::getPMV ( ) const
```

Definition at line 72 of file Occupant_Action_Heat_Gains.cpp.

Here is the caller graph for this function:

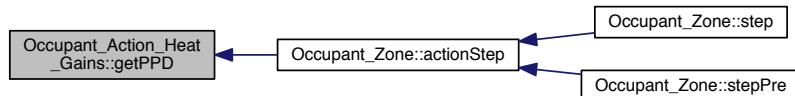


15.57.3.2 getPPD()

```
double Occupant_Action_Heat_Gains::getPPD ( ) const
```

Definition at line 76 of file Occupant_Action_Heat_Gains.cpp.

Here is the caller graph for this function:

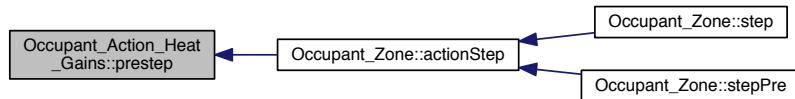


15.57.3.3 prestep()

```
void Occupant_Action_Heat_Gains::prestep (
    double clo,
    double metabolicRate )
```

Definition at line 30 of file Occupant_Action_Heat_Gains.cpp.

Here is the caller graph for this function:



15.57.3.4 setup()

```
void Occupant_Action_Heat_Gains::setup (
    int buildingID,
    int agentid )
```

Definition at line 12 of file Occupant_Action_Heat_Gains.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.57.3.5 step()

```
void Occupant_Action_Heat_Gains::step (
    const Building_Zone & zone,
    const bool inZone )
```

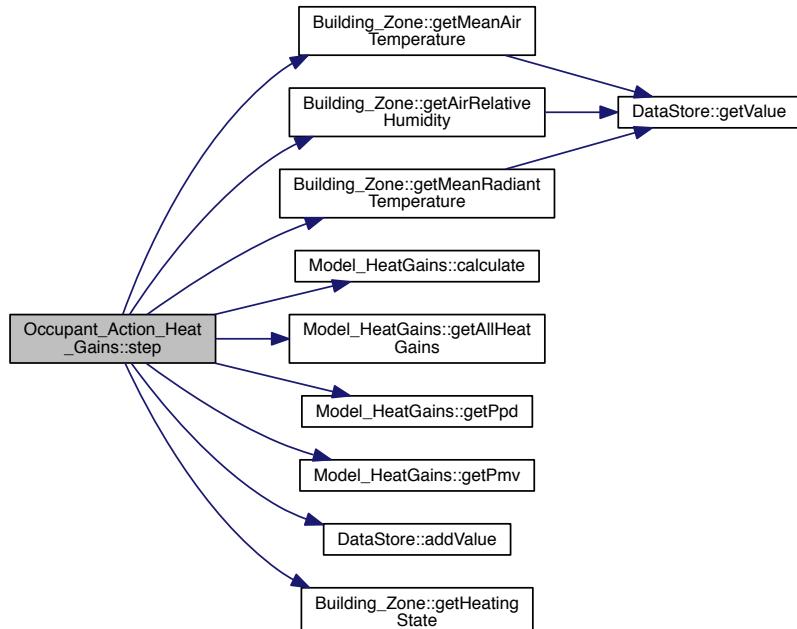
Calculates the Fanger pmv and sets the instance variables related to results.

Parameters

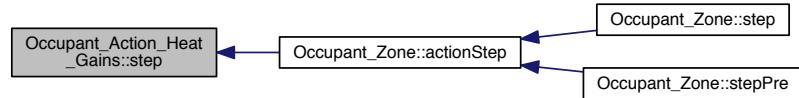
<i>metabolicRate</i>	Metabolic Rate
<i>partialWaterPressure</i>	partial water vapour kPa
<i>meanRadiantTemperature</i>	mean radiant temperature C
<i>externalWork</i>	external work
<i>ta</i>	air temperature
<i>clo</i>	Clothing value
<i>airVelocity</i>	Air velocity

Definition at line 35 of file Occupant_Action_Heat_Gains.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



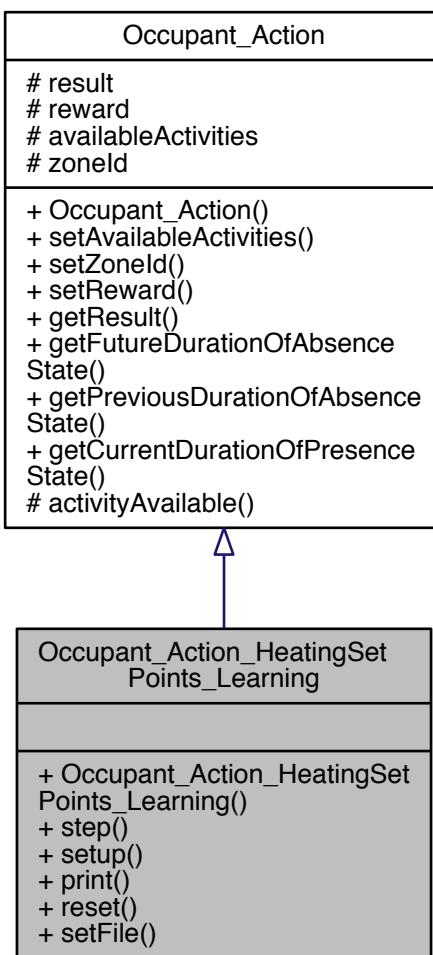
The documentation for this class was generated from the following files:

- Occupant_Action_Heat_Gains.hpp
- Occupant_Action_Heat_Gains.cpp

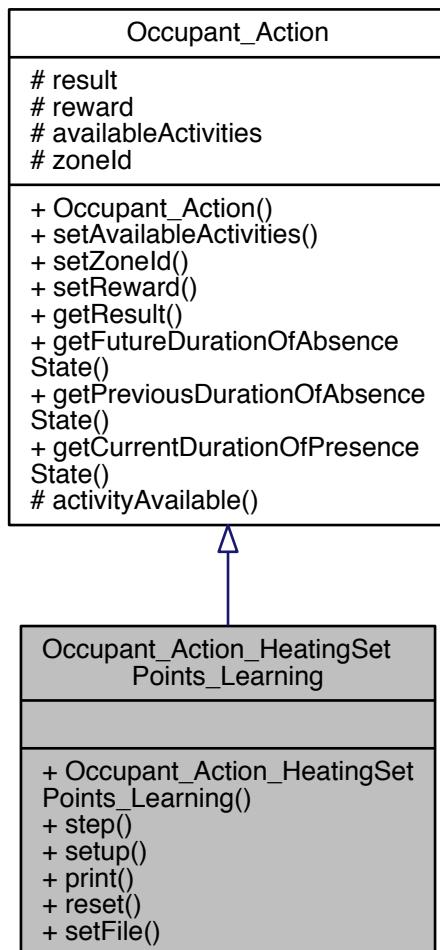
15.58 Occupant_Action_HeatingSetPoints_Learning Class Reference

```
#include <Occupant_Action_HeatingSetPoints_Learning.hpp>
```

Inheritance diagram for Occupant_Action_HeatingSetPoints_Learning:



Collaboration diagram for Occupant_Action_HeatingSetPoints_Learning:



Public Member Functions

- [`Occupant_Action_HeatingSetPoints_Learning \(\)`](#)
- `void step (const Building_Zone &zone, const bool inZone)`
- `void setup (const int id, const int learn)`
- `void print ()`
- `void reset ()`
- `void setFile (std::string file)`

Additional Inherited Members

15.58.1 Detailed Description

Definition at line 14 of file `Occupant_Action_HeatingSetPoints_Learning.hpp`.

15.58.2 Constructor & Destructor Documentation

15.58.2.1 Occupant_Action_HeatingSetPoints_Learning()

```
Occupant_Action_HeatingSetPoints_Learning::Occupant_Action_HeatingSetPoints_Learning ( )
```

Definition at line 16 of file Occupant_Action_HeatingSetPoints_Learning.cpp.

15.58.3 Member Function Documentation

15.58.3.1 print()

```
void Occupant_Action_HeatingSetPoints_Learning::print ( )
```

Definition at line 27 of file Occupant_Action_HeatingSetPoints_Learning.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.58.3.2 reset()

```
void Occupant_Action_HeatingSetPoints_Learning::reset ( )
```

Definition at line 31 of file Occupant_Action_HeatingSetPoints_Learning.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.58.3.3 setFile()

```
void Occupant_Action_HeatingSetPoints_Learning::setFile ( std::string file )
```

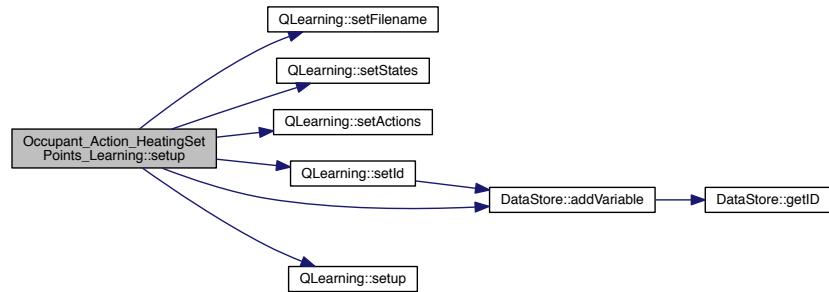
Definition at line 23 of file Occupant_Action_HeatingSetPoints_Learning.cpp.

15.58.3.4 setup()

```
void Occupant_Action_HeatingSetPoints_Learning::setup ( const int id, const int learn )
```

Definition at line 36 of file Occupant_Action_HeatingSetPoints_Learning.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.58.3.5 step()

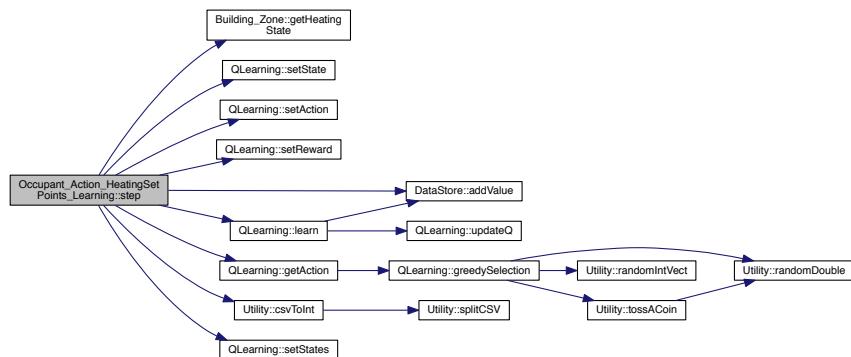
```

void Occupant_Action_HeatingSetPoints_Learning::step (
    const Building_Zone & zone,
    const bool inZone )

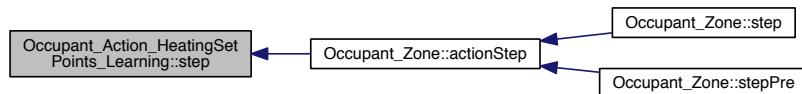
```

Definition at line 60 of file `Occupant_Action_HeatingSetPoints_Learning.cpp`.

Here is the call graph for this function:



Here is the caller graph for this function:



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

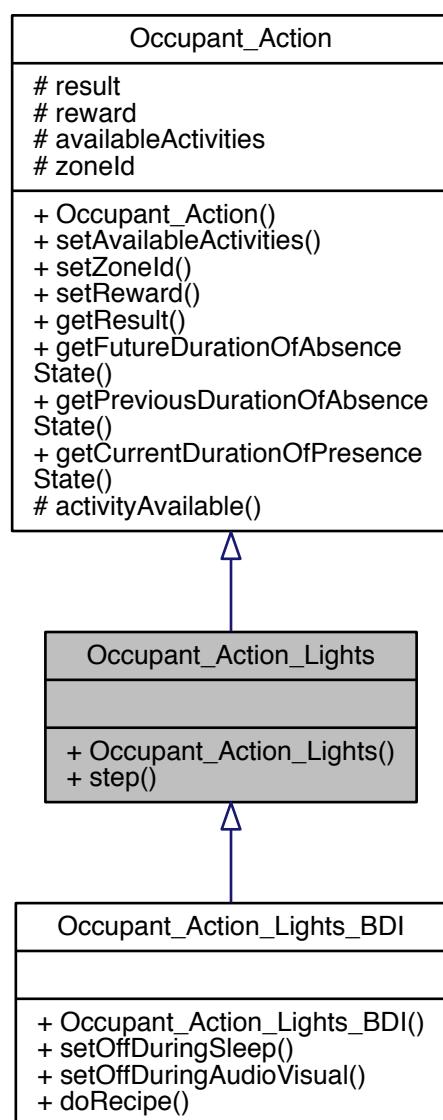
- `Occupant_Action_HeatingSetPoints_Learning.hpp`
- `Occupant_Action_HeatingSetPoints_Learning.cpp`

15.59 Occupant_Action_Lights Class Reference

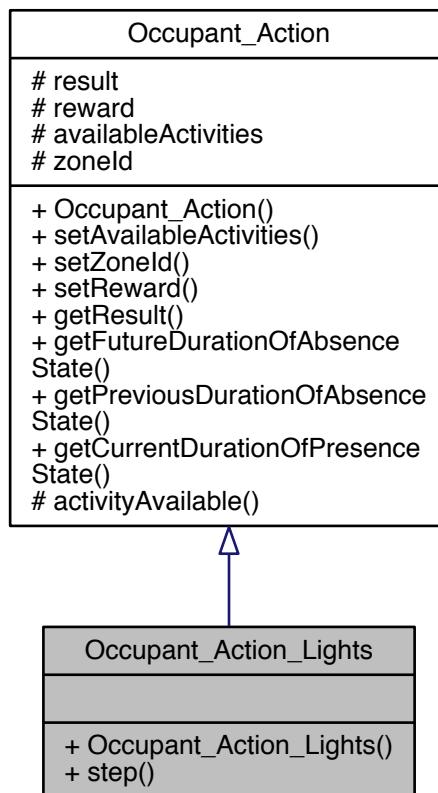
Occupant action on lights.

```
#include <Occupant_Action_Lights.hpp>
```

Inheritance diagram for Occupant_Action_Lights:



Collaboration diagram for Occupant_Action_Lights:



Public Member Functions

- [Occupant_Action_Lights \(\)](#)
- void [step](#) (const `Building_Zone` &zone, const bool inZone, const bool previouslyInZone, const std::vector<double > &activities)

Additional Inherited Members

15.59.1 Detailed Description

Occupant action on lights.

Occupant action on lights

Definition at line 13 of file `Occupant_Action_Lights.hpp`.

15.59.2 Constructor & Destructor Documentation

15.59.2.1 Occupant_Action_Lights()

```
Occupant_Action_Lights::Occupant_Action_Lights ( )
```

Definition at line 8 of file Occupant_Action_Lights.cpp.

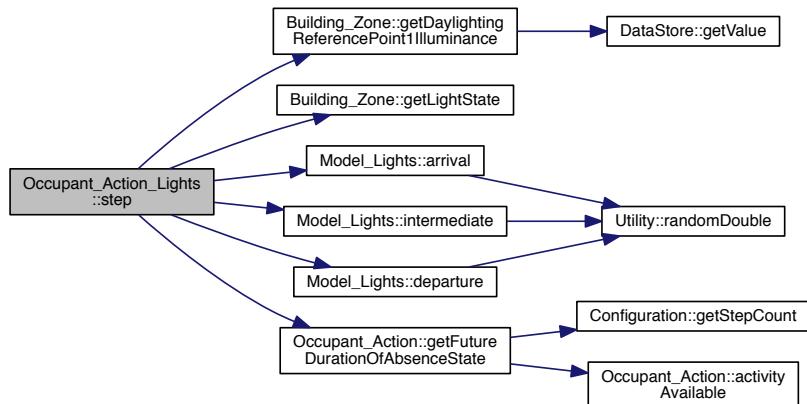
15.59.3 Member Function Documentation

15.59.3.1 step()

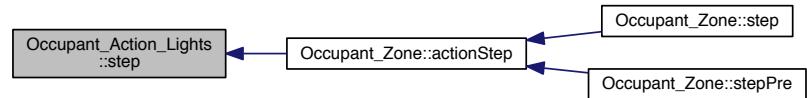
```
void Occupant_Action_Lights::step (
    const Building_Zone & zone,
    const bool inZone,
    const bool previouslyInZone,
    const std::vector< double > & activities )
```

Definition at line 10 of file Occupant_Action_Lights.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



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

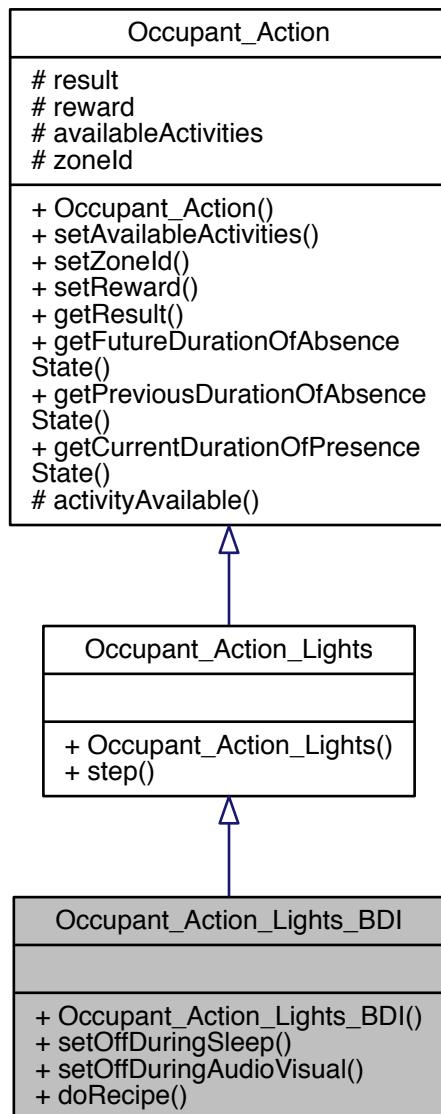
- `Occupant_Action_Lights.hpp`
- `Occupant_Action_Lights.cpp`

15.60 Occupant_Action_Lights_BDI Class Reference

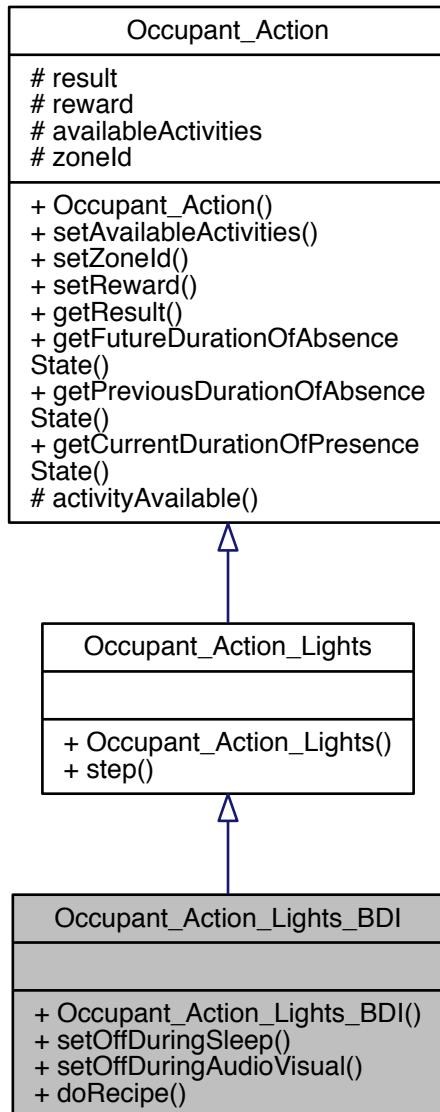
Occupant action on lights using BDI.

```
#include <Occupant_Action_Lights_BDI.hpp>
```

Inheritance diagram for Occupant_Action_Lights_BDI:



Collaboration diagram for Occupant_Action_Lights_BDI:



Public Member Functions

- [Occupant_Action_Lights_BDI \(\)](#)
- void [setOffDuringSleep](#) (double OffDuringSleep)
- void [setOffDuringAudioVisual](#) (double OffDuringAudioVisual)
- bool [doRecipe](#) (const std::vector<double> &activities)

Additional Inherited Members

15.60.1 Detailed Description

Occupant action on lights using BDI.

Occupant action on lights using BDI adapted from

Chapman, J., Siebers, P., & Robinson, D. (2017). Data Scarce Behavioural Modelling and the Representation of Social Interactions. Unpublished Manuscript, 1–48.

Definition at line 14 of file Occupant_Action_Lights_BDI.hpp.

15.60.2 Constructor & Destructor Documentation

15.60.2.1 Occupant_Action_Lights_BDI()

```
Occupant_Action_Lights_BDI::Occupant_Action_Lights_BDI ( )
```

Definition at line 7 of file Occupant_Action_Lights_BDI.cpp.

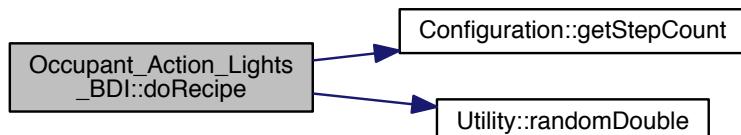
15.60.3 Member Function Documentation

15.60.3.1 doRecipe()

```
bool Occupant_Action_Lights_BDI::doRecipe (
    const std::vector< double > & activities )
```

Definition at line 21 of file Occupant_Action_Lights_BDI.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.60.3.2 setOffDuringAudioVisual()

```
void Occupant_Action_Lights_BDI::setOffDuringAudioVisual (
    double OffDuringAudioVisual )
```

Definition at line 12 of file Occupant_Action_Lights_BDI.cpp.

Here is the caller graph for this function:

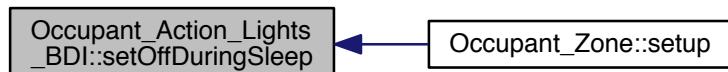


15.60.3.3 setOffDuringSleep()

```
void Occupant_Action_Lights_BDI::setOffDuringSleep (
    double OffDuringSleep )
```

Definition at line 17 of file Occupant_Action_Lights_BDI.cpp.

Here is the caller graph for this function:



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

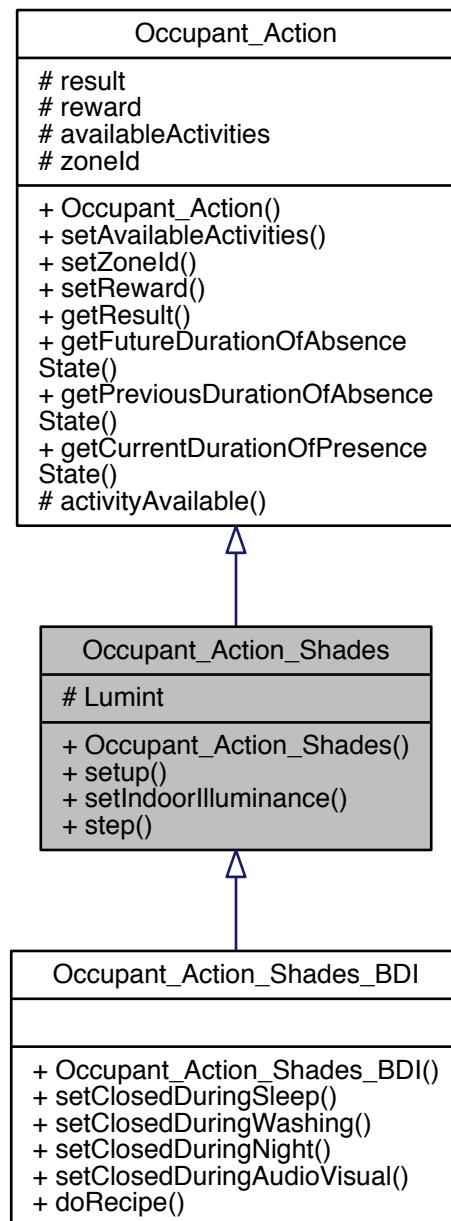
- Occupant_Action_Lights_BDI.hpp
- Occupant_Action_Lights_BDI.cpp

15.61 Occupant_Action_Shades Class Reference

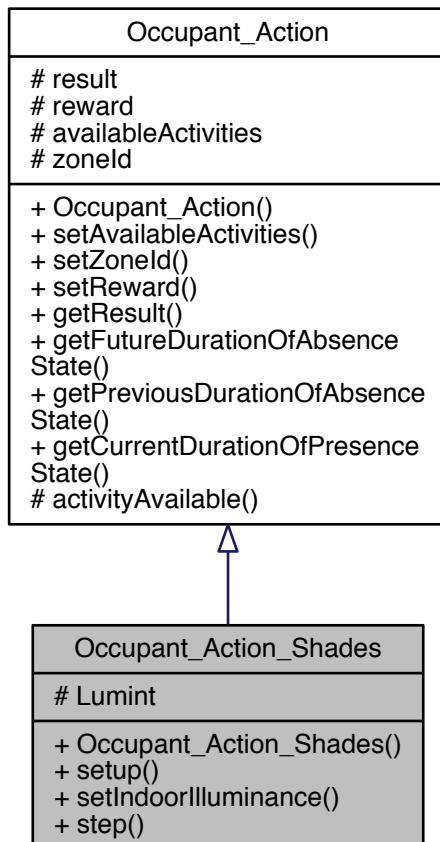
Occupant action on shades.

```
#include <Occupant_Action_Shades.hpp>
```

Inheritance diagram for Occupant_Action_Shades:



Collaboration diagram for Occupant_Action_Shades:



Public Member Functions

- `Occupant_Action_Shades ()`
- `void setup (int windowID)`
- `void setIndoorIlluminance (const float lumint)`
- `void step (const Building_Zone &zone, const bool inZone, const bool previouslyInZone)`

Protected Attributes

- `float Lumint = 0`

Additional Inherited Members

15.61.1 Detailed Description

Occupant action on shades.

Occupant action on shades

Definition at line 14 of file Occupant_Action_Shades.hpp.

15.61.2 Constructor & Destructor Documentation

15.61.2.1 Occupant_Action_Shades()

```
Occupant_Action_Shades::Occupant_Action_Shades ( )
```

Definition at line 11 of file Occupant_Action_Shades.cpp.

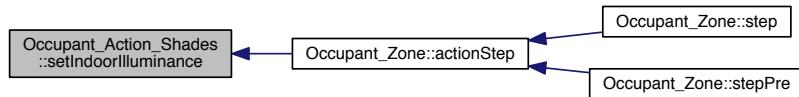
15.61.3 Member Function Documentation

15.61.3.1 setIndoorIlluminance()

```
void Occupant_Action_Shades::setIndoorIlluminance (
    const float lumint )
```

Definition at line 39 of file Occupant_Action_Shades.cpp.

Here is the caller graph for this function:

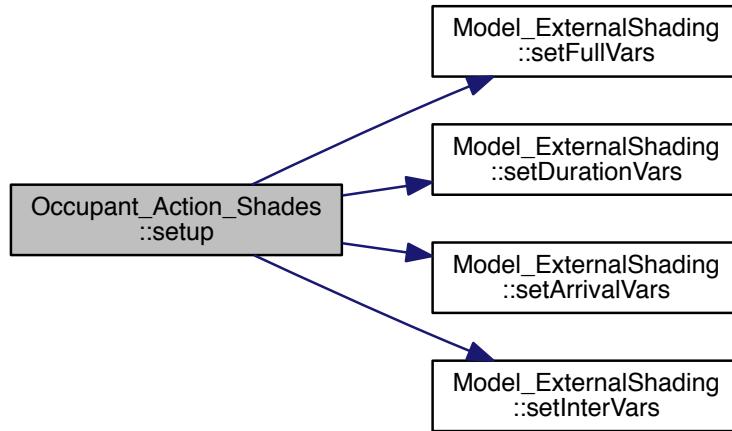


15.61.3.2 setup()

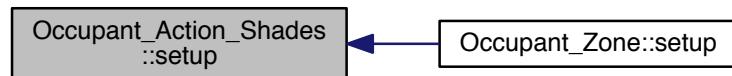
```
void Occupant_Action_Shades::setup (
    int windowID )
```

Definition at line 13 of file Occupant_Action_Shades.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

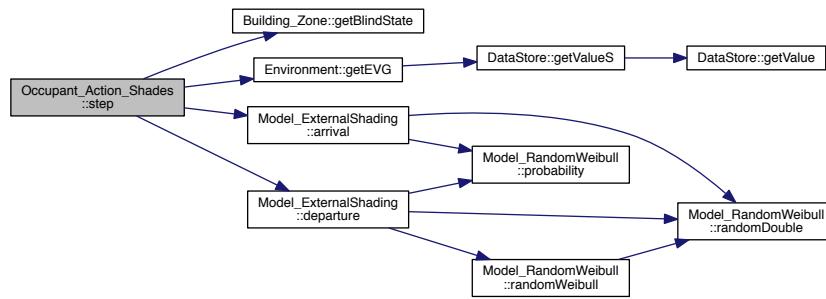


15.61.3.3 step()

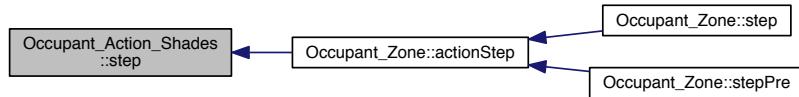
```
void Occupant_Action_Shades::step (
    const Building_Zone & zone,
    const bool inZone,
    const bool previouslyInZone )
```

Definition at line 24 of file `Occupant_Action_Shades.cpp`.

Here is the call graph for this function:



Here is the caller graph for this function:



15.61.4 Member Data Documentation

15.61.4.1 Lumint

```
float Occupant_Action_Shades::Lumint = 0 [protected]
```

Definition at line 23 of file Occupant_Action_Shades.hpp.

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

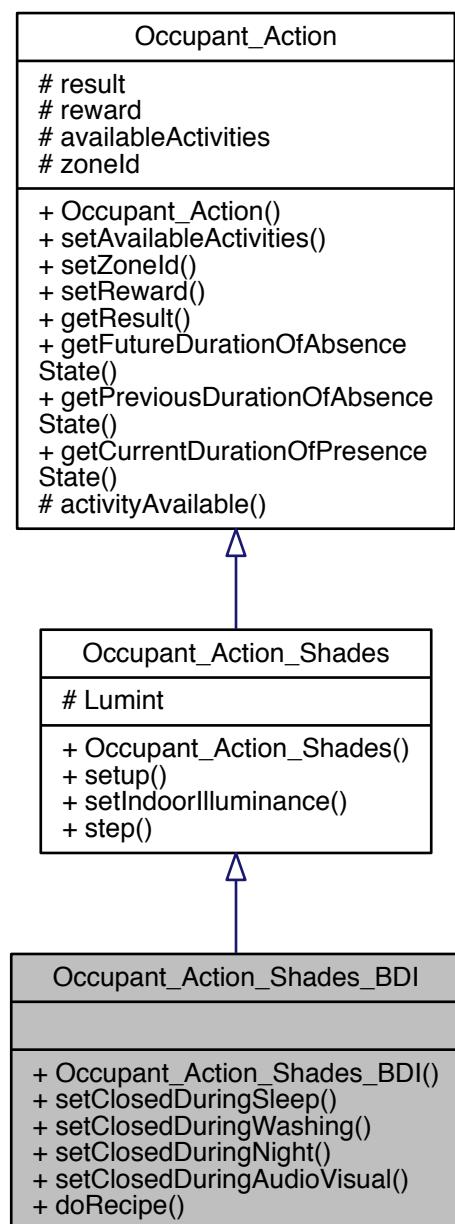
- Occupant_Action_Shades.hpp
- Occupant_Action_Shades.cpp

15.62 Occupant_Action_Shades_BDI Class Reference

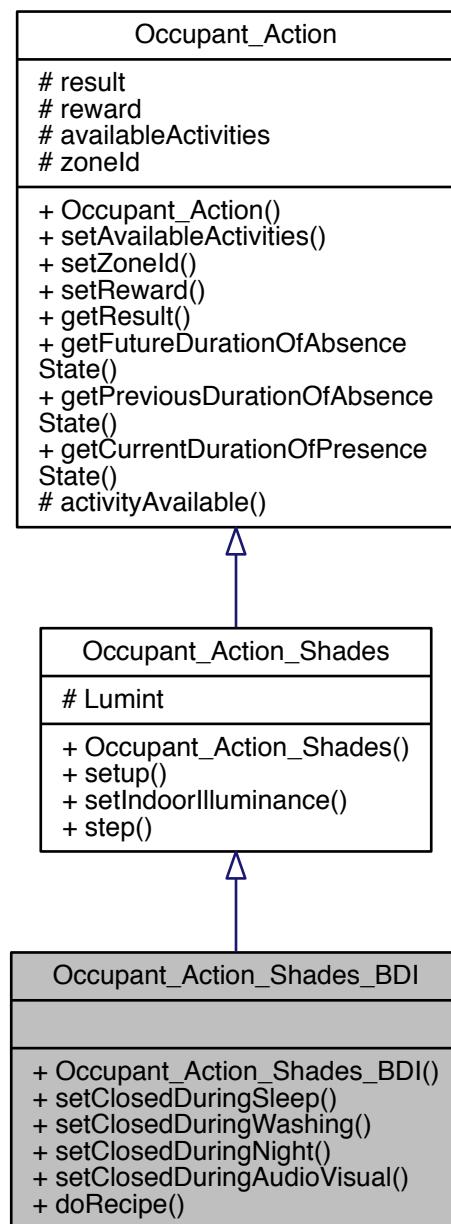
Occupant action on shades using BDI.

```
#include <Occupant_Action_Shades_BDI.hpp>
```

Inheritance diagram for Occupant_Action_Shades_BDI:



Collaboration diagram for Occupant_Action_Shades_BDI:



Public Member Functions

- [Occupant_Action_Shades_BDI \(\)](#)
- void [setClosedDuringSleep](#) (double ShadeClosedDuringSleep)
- void [setClosedDuringWashing](#) (double ShadeClosedDuringWashing)
- void [setClosedDuringNight](#) (double ShadeClosedDuringNight)
- void [setClosedDuringAudioVisual](#) (double ShadeClosedDuringAudioVisual)
- bool [doRecipe](#) (const std::vector< double > &activities)

Additional Inherited Members

15.62.1 Detailed Description

Occupant action on shades using BDI.

Occupant action on shades using BDI adapted from Chapman, J., Siebers, P., & Robinson, D. (2017). Data Scarce Behavioural Modelling and the Representation of Social Interactions. Unpublished Manuscript, 1–48.

Definition at line 14 of file Occupant_Action_Shades_BDI.hpp.

15.62.2 Constructor & Destructor Documentation

15.62.2.1 Occupant_Action_Shades_BDI()

```
Occupant_Action_Shades_BDI::Occupant_Action_Shades_BDI ( )
```

Definition at line 10 of file Occupant_Action_Shades_BDI.cpp.

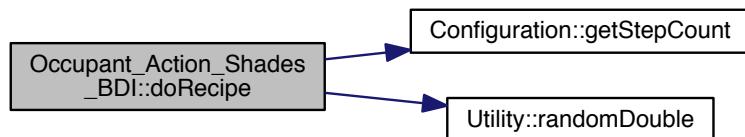
15.62.3 Member Function Documentation

15.62.3.1 doRecipe()

```
bool Occupant_Action_Shades_BDI::doRecipe (
    const std::vector< double > & activities )
```

Definition at line 36 of file Occupant_Action_Shades_BDI.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

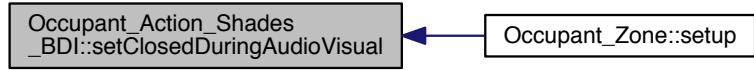


15.62.3.2 setClosedDuringAudioVisual()

```
void Occupant_Action_Shades_BDI::setClosedDuringAudioVisual ( double ShadeClosedDuringAudioVisual )
```

Definition at line 31 of file Occupant_Action_Shades_BDI.cpp.

Here is the caller graph for this function:

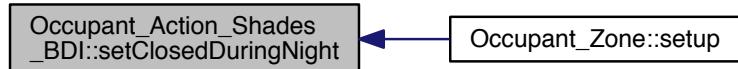


15.62.3.3 setClosedDuringNight()

```
void Occupant_Action_Shades_BDI::setClosedDuringNight ( double ShadeClosedDuringNight )
```

Definition at line 27 of file Occupant_Action_Shades_BDI.cpp.

Here is the caller graph for this function:

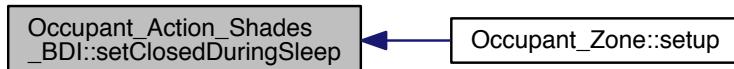


15.62.3.4 setClosedDuringSleep()

```
void Occupant_Action_Shades_BDI::setClosedDuringSleep (
    double ShadeClosedDuringSleep )
```

Definition at line 22 of file Occupant_Action_Shades_BDI.cpp.

Here is the caller graph for this function:



15.62.3.5 setClosedDuringWashing()

```
void Occupant_Action_Shades_BDI::setClosedDuringWashing (
    double ShadeClosedDuringWashing )
```

Definition at line 17 of file Occupant_Action_Shades_BDI.cpp.

Here is the caller graph for this function:



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

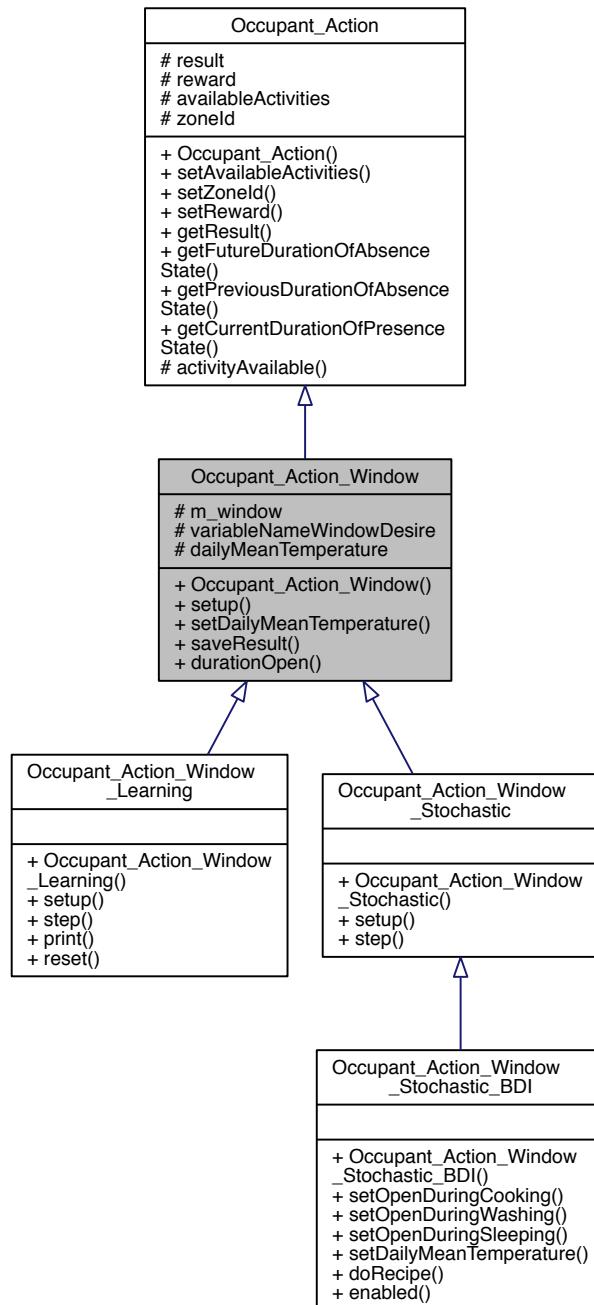
- Occupant_Action_Shades_BDI.hpp
- Occupant_Action_Shades_BDI.cpp

15.63 Occupant_Action_Window Class Reference

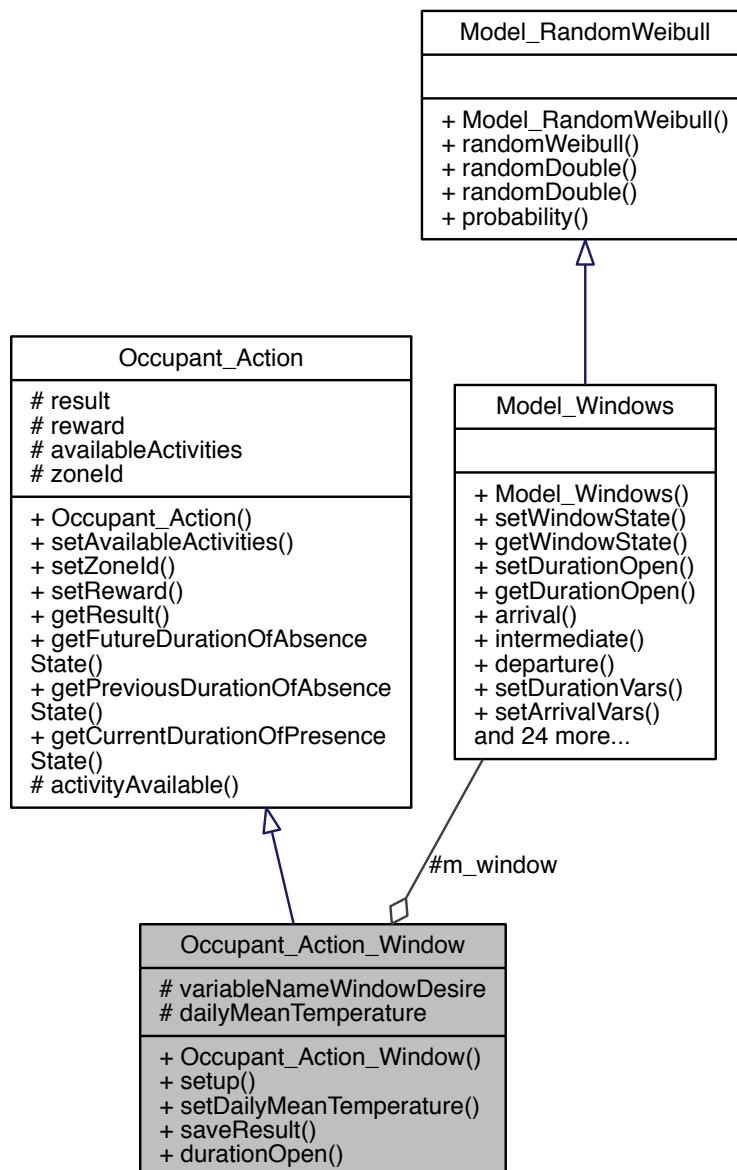
Occupant action on windows.

```
#include <Occupant_Action_Window.hpp>
```

Inheritance diagram for Occupant_Action_Window:



Collaboration diagram for Occupant_Action_Window:



Public Member Functions

- [Occupant_Action_Window \(\)](#)
- void [setup \(int windowID, int id\)](#)
- void [setDailyMeanTemperature \(double dailyMeanTemperature\)](#)
- void [saveResult \(\)](#)
- int [durationOpen \(\) const](#)

Protected Attributes

- Model_Windows m_window
- int variableNameWindowDesire
- double dailyMeanTemperature

Additional Inherited Members

15.63.1 Detailed Description

Occupant action on windows.

Occupant action on windows

Definition at line 15 of file Occupant_Action_Window.hpp.

15.63.2 Constructor & Destructor Documentation

15.63.2.1 Occupant_Action_Window()

```
Occupant_Action_Window::Occupant_Action_Window ( )
```

Definition at line 10 of file Occupant_Action_Window.cpp.

15.63.3 Member Function Documentation

15.63.3.1 durationOpen()

```
int Occupant_Action_Window::durationOpen ( ) const
```

Definition at line 28 of file Occupant_Action_Window.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

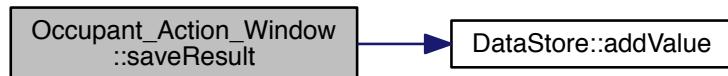


15.63.3.2 saveResult()

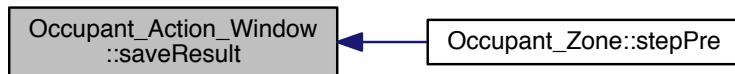
```
void Occupant_Action_Window::saveResult ( )
```

Definition at line 24 of file Occupant_Action_Window.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

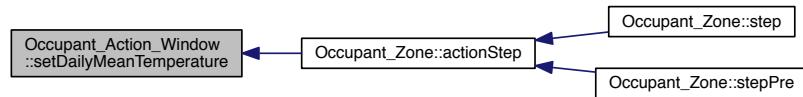


15.63.3.3 setDailyMeanTemperature()

```
void Occupant_Action_Window::setDailyMeanTemperature ( double dailyMeanTemperature )
```

Definition at line 13 of file Occupant_Action_Window.cpp.

Here is the caller graph for this function:



15.63.3.4 setup()

```
void Occupant_Action_Window::setup (
    int windowID,
    int id )
```

Definition at line 18 of file Occupant_Action_Window.cpp.

Here is the call graph for this function:



15.63.4 Member Data Documentation

15.63.4.1 dailyMeanTemperature

```
double Occupant_Action_Window::dailyMeanTemperature [protected]
```

Definition at line 26 of file Occupant_Action_Window.hpp.

15.63.4.2 m_window

```
Model_Windows Occupant_Action_Window::m_window [protected]
```

Definition at line 24 of file Occupant_Action_Window.hpp.

15.63.4.3 variableNameWindowDesire

```
int Occupant_Action_Window::variableNameWindowDesire [protected]
```

Definition at line 25 of file Occupant_Action_Window.hpp.

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

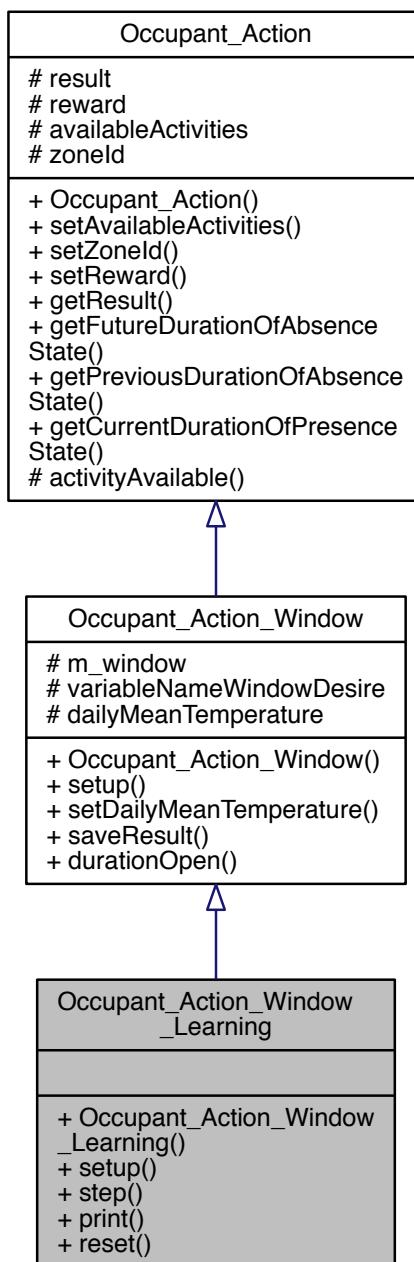
- Occupant_Action_Window.hpp
- Occupant_Action_Window.cpp

15.64 Occupant_Action_Window_Learning Class Reference

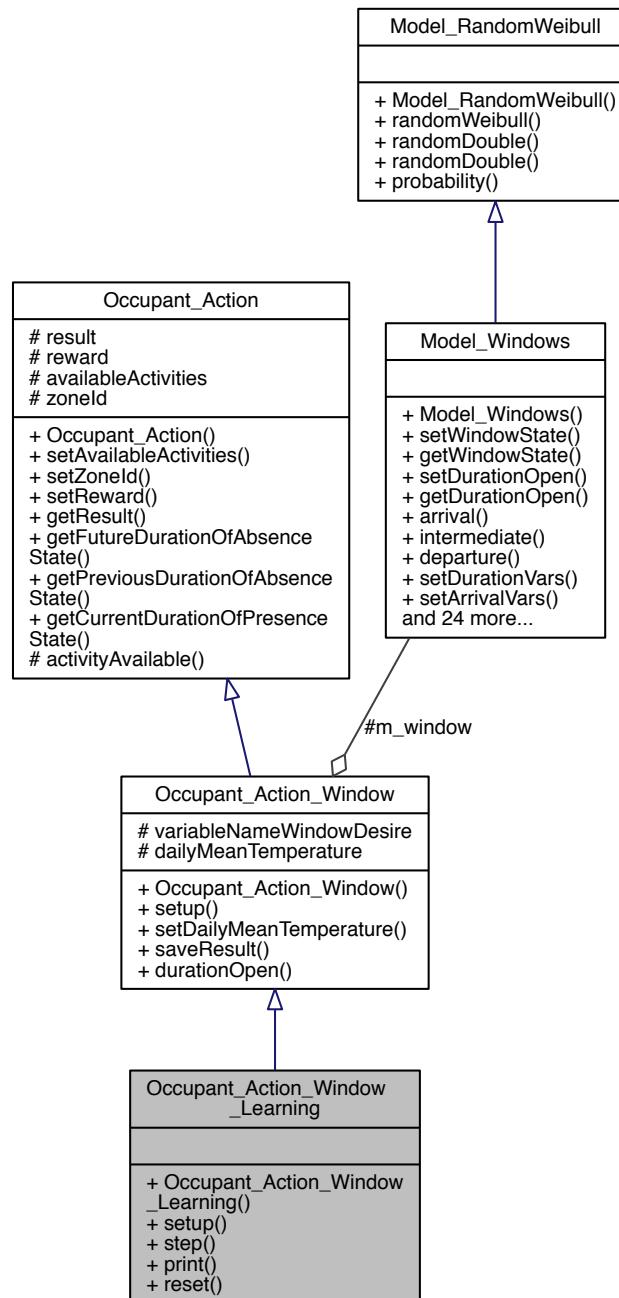
Occupant action on windows using Q-Learing model.

```
#include <Occupant_Action_Window_Learning.hpp>
```

Inheritance diagram for Occupant_Action_Window_Learning:



Collaboration diagram for Occupant_Action_Window_Learning:



Public Member Functions

- [Occupant_Action_Window_Learning \(\)](#)
- void [setup \(const int id\)](#)
- void [step \(const Building_Zone &zone, const bool inZone, const bool previouslyInZone\)](#)
- void [print \(\)](#)
- void [reset \(\)](#)

Additional Inherited Members

15.64.1 Detailed Description

Occupant action on windows using Q-Learning model.

Occupant action on windows using Q-Learning model

Definition at line 15 of file Occupant_Action_Window_Learning.hpp.

15.64.2 Constructor & Destructor Documentation

15.64.2.1 Occupant_Action_Window_Learning()

```
Occupant_Action_Window_Learning::Occupant_Action_Window_Learning ()
```

Definition at line 9 of file Occupant_Action_Window_Learning.cpp.

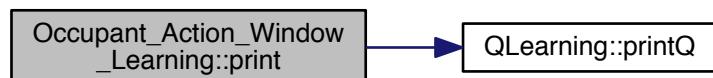
15.64.3 Member Function Documentation

15.64.3.1 print()

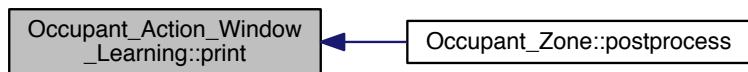
```
void Occupant_Action_Window_Learning::print ()
```

Definition at line 11 of file Occupant_Action_Window_Learning.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.64.3.2 reset()

```
void Occupant_Action_Window_Learning::reset ( )
```

Definition at line 14 of file Occupant_Action_Window_Learning.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

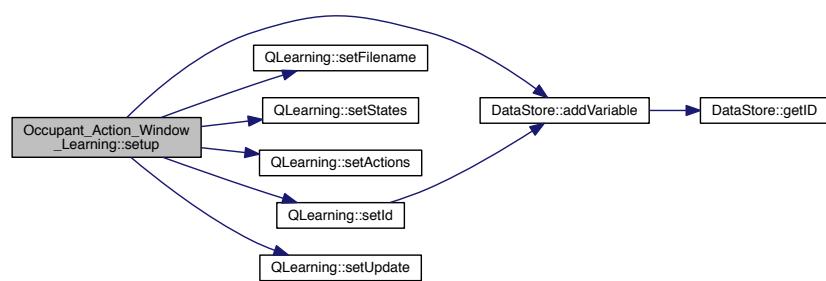


15.64.3.3 setup()

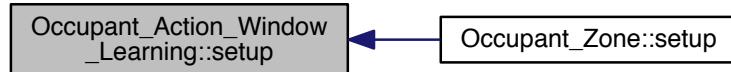
```
void Occupant_Action_Window_Learning::setup ( const int id )
```

Definition at line 18 of file Occupant_Action_Window_Learning.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



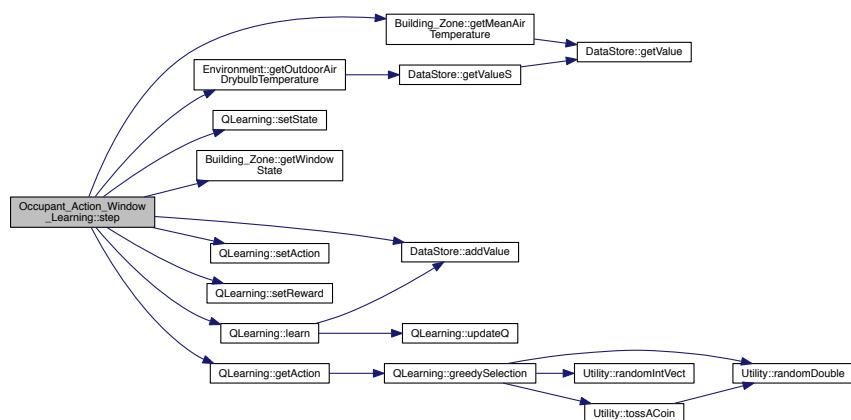
15.64.3.4 step()

```

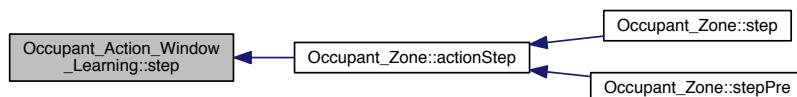
void Occupant_Action_Window_Learning::step (
    const Building_Zone & zone,
    const bool inZone,
    const bool previouslyInZone )
  
```

Definition at line 35 of file Occupant_Action_Window_Learning.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



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

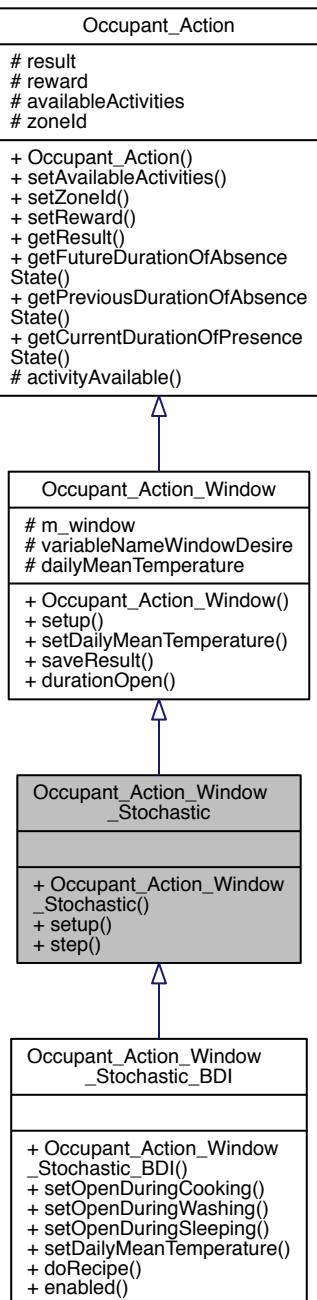
- Occupant_Action_Window_Learning.hpp
- Occupant_Action_Window_Learning.cpp

15.65 Occupant_Action_Window_Stochastic Class Reference

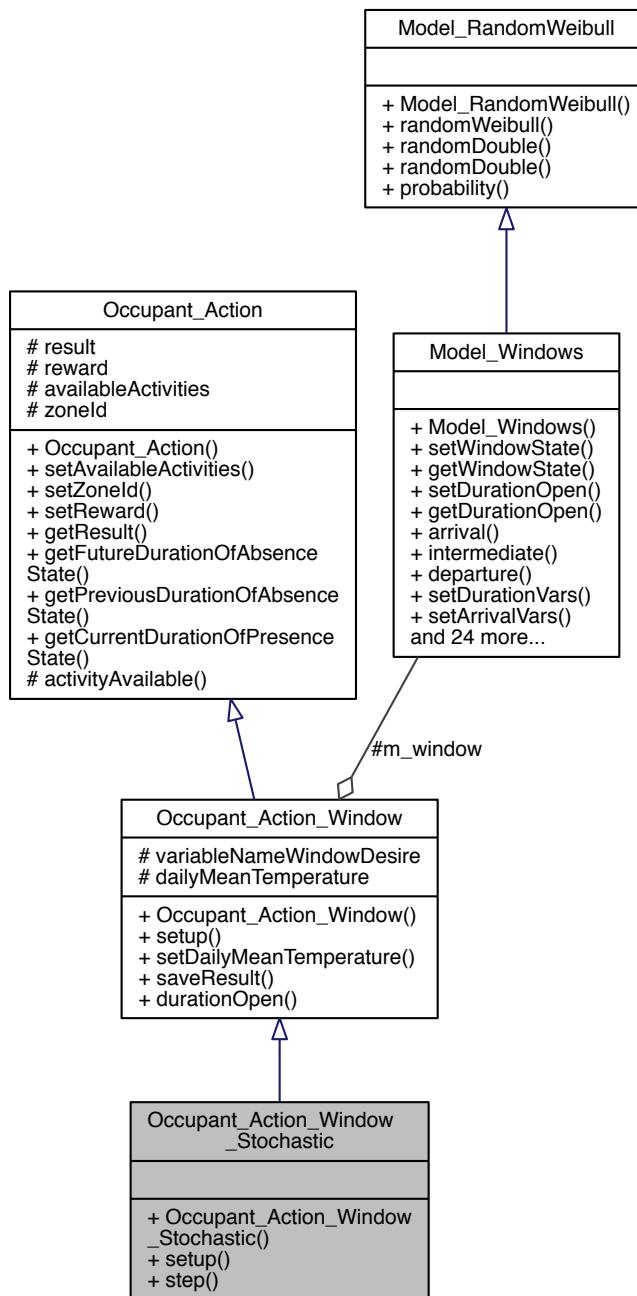
Occupant action on windows using stochastic model.

```
#include <Occupant_Action_Window_Stochastic.hpp>
```

Inheritance diagram for Occupant_Action_Window_Stochastic:



Collaboration diagram for Occupant_Action_Window_Stochastic:



Public Member Functions

- [Occupant_Action_Window_Stochastic \(\)](#)
- void [setup](#) (int windowID, int id)
- void [step](#) (const [Building_Zone](#) &zone, const bool inZone, const bool previouslyInZone, const std::vector<double > &activities)

Additional Inherited Members

15.65.1 Detailed Description

Occupant action on windows using stochastic model.

Occupant action on windows using stochastic model

Definition at line 13 of file Occupant_Action_Window_Stochastic.hpp.

15.65.2 Constructor & Destructor Documentation

15.65.2.1 Occupant_Action_Window_Stochastic()

```
Occupant_Action_Window_Stochastic::Occupant_Action_Window_Stochastic ( )
```

Definition at line 11 of file Occupant_Action_Window_Stochastic.cpp.

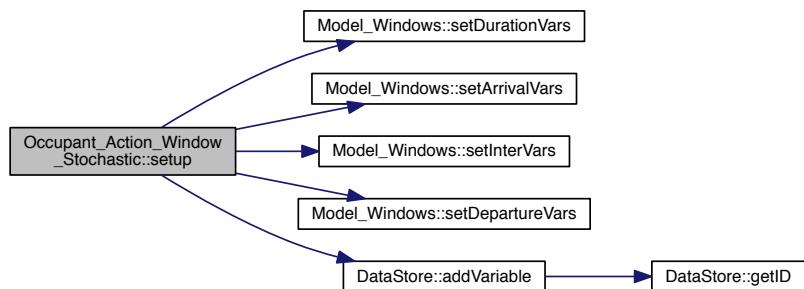
15.65.3 Member Function Documentation

15.65.3.1 setup()

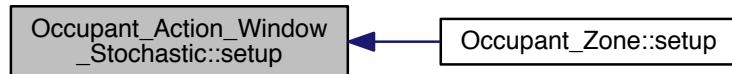
```
void Occupant_Action_Window_Stochastic::setup (
    int windowID,
    int id )
```

Definition at line 14 of file Occupant_Action_Window_Stochastic.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



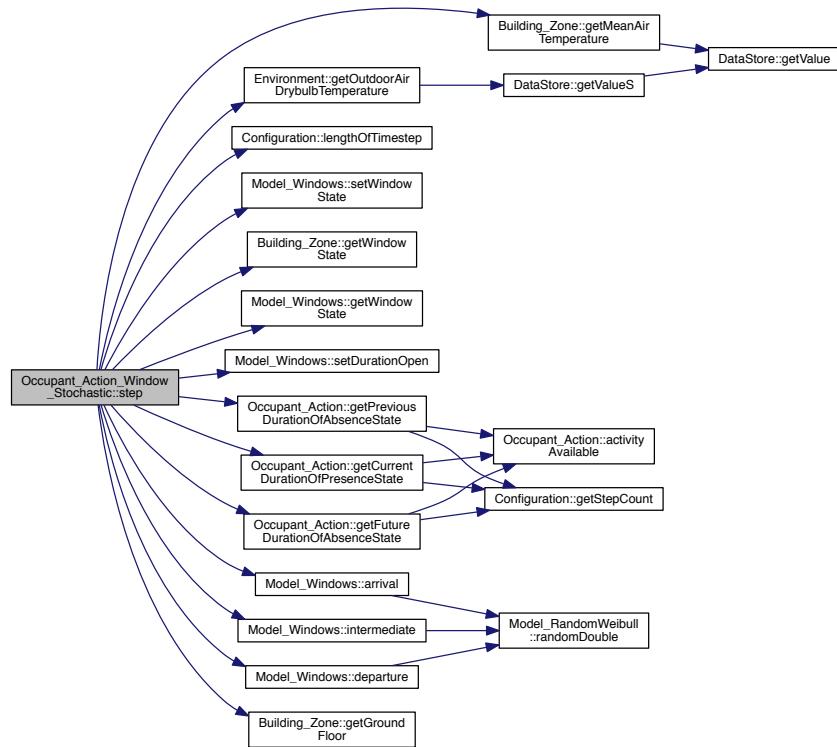
15.65.3.2 step()

```

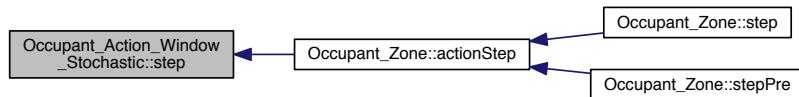
void Occupant_Action_Window_Stochastic::step (
    const Building_Zone & zone,
    const bool inZone,
    const bool previouslyInZone,
    const std::vector< double > & activities )
  
```

Definition at line 30 of file Occupant_Action_Window_Stochastic.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



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

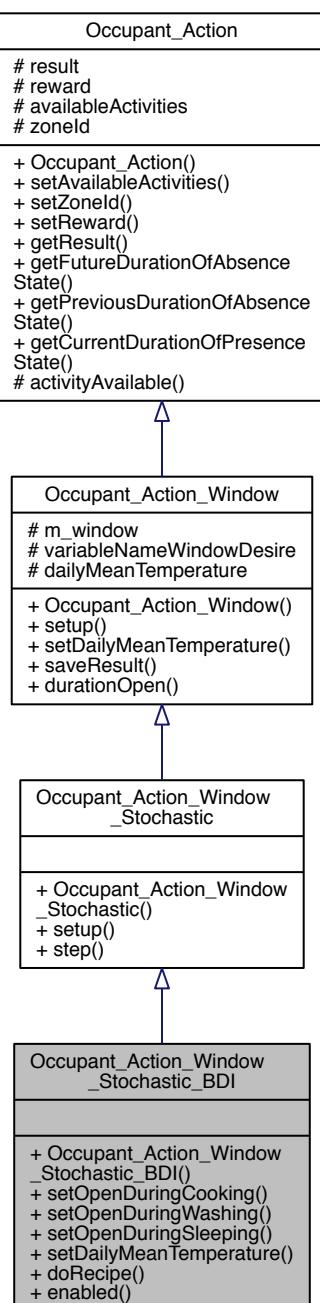
- `Occupant_Action_Window_Stochastic.hpp`
- `Occupant_Action_Window_Stochastic.cpp`

15.66 Occupant_Action_Window_Stochastic_BDI Class Reference

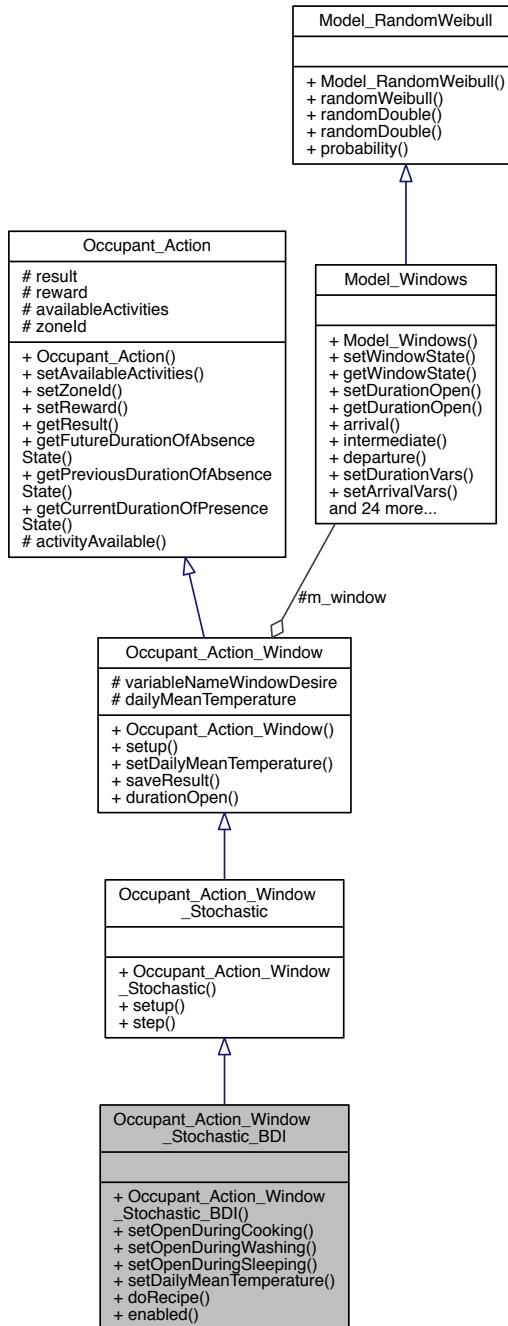
Occupant action on windows using BDI.

```
#include <Occupant_Action_Window_Stochastic_BDI.hpp>
```

Inheritance diagram for Occupant_Action_Window_Stochastic_BDI:



Collaboration diagram for Occupant_Action_Window_Stochastic_BDI:



Public Member Functions

- `Occupant_Action_Window_Stochastic_BDI ()`
- void `setOpenDuringCooking` (double `OpenDuringCooking`)
- void `setOpenDuringWashing` (double `OpenDuringWashing`)
- void `setOpenDuringSleeping` (double `OpenDuringSleeping`)
- void `setDailyMeanTemperature` (double `dailyMeanTemperature`)

- bool `doRecipe` (const std::vector< double > &activities)
- bool `enabled` () const

Additional Inherited Members

15.66.1 Detailed Description

Occupant action on windows using BDI.

Occupant action on windows using BDI adapted from

Chapman, J., Siebers, P., & Robinson, D. (2017). Data Scarce Behavioural Modelling and the Representation of Social Interactions. Unpublished Manuscript, 1–48.

Definition at line 12 of file Occupant_Action_Window_Stochastic_BDI.hpp.

15.66.2 Constructor & Destructor Documentation

15.66.2.1 Occupant_Action_Window_Stochastic_BDI()

`Occupant_Action_Window_Stochastic_BDI::Occupant_Action_Window_Stochastic_BDI ()`

Definition at line 10 of file Occupant_Action_Window_Stochastic_BDI.cpp.

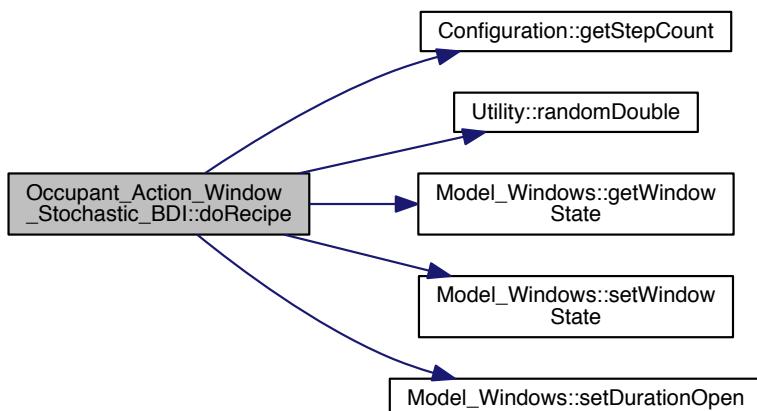
15.66.3 Member Function Documentation

15.66.3.1 doRecipe()

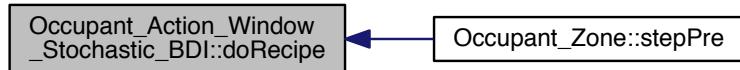
```
bool Occupant_Action_Window_Stochastic_BDI::doRecipe (
    const std::vector< double > & activities )
```

Definition at line 36 of file Occupant_Action_Window_Stochastic_BDI.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.66.3.2 enabled()

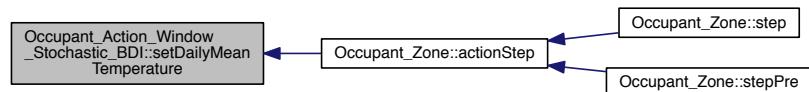
```
bool Occupant_Action_Window_Stochastic_BDI::enabled () const
```

15.66.3.3 setDailyMeanTemperature()

```
void Occupant_Action_Window_Stochastic_BDI::setDailyMeanTemperature (
    double dailyMeanTemperature )
```

Definition at line 31 of file Occupant_Action_Window_Stochastic_BDI.cpp.

Here is the caller graph for this function:

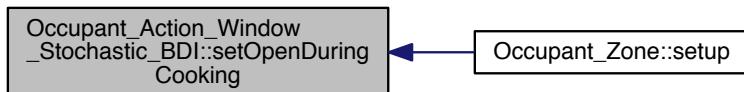


15.66.3.4 setOpenDuringCooking()

```
void Occupant_Action_Window_Stochastic_BDI::setOpenDuringCooking (
    double OpenDuringCooking )
```

Definition at line 21 of file Occupant_Action_Window_Stochastic_BDI.cpp.

Here is the caller graph for this function:

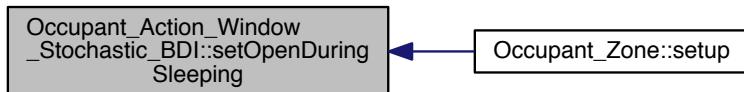


15.66.3.5 setOpenDuringSleeping()

```
void Occupant_Action_Window_Stochastic_BDI::setOpenDuringSleeping ( double OpenDuringSleeping )
```

Definition at line 26 of file Occupant_Action_Window_Stochastic_BDI.cpp.

Here is the caller graph for this function:

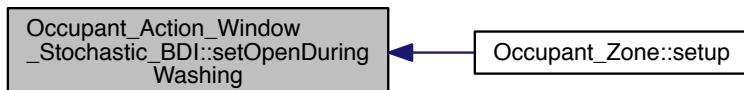


15.66.3.6 setOpenDuringWashing()

```
void Occupant_Action_Window_Stochastic_BDI::setOpenDuringWashing ( double OpenDuringWashing )
```

Definition at line 16 of file Occupant_Action_Window_Stochastic_BDI.cpp.

Here is the caller graph for this function:



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

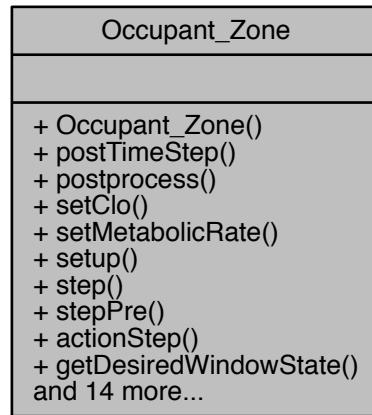
- Occupant_Action_Window_Stochastic_BDI.hpp
- Occupant_Action_Window_Stochastic_BDI.cpp

15.67 Occupant_Zone Class Reference

The occupants understanding of a zone.

```
#include <Occupant_Zone.hpp>
```

Collaboration diagram for Occupant_Zone:



Public Member Functions

- [Occupant_Zone \(\)](#)
- void [postTimeStep \(\)](#)
- void [postprocess \(\)](#)
- void [setClo \(double clo\)](#)
- void [setMetabolicRate \(double metabolicRate\)](#)
- void [setup \(int buildingID, const Building_Zone &buldingZone, int agentid, const ConfigStructAgent &agent\)](#)

Initialises the occupants understanding of a zone.
- void [step \(const Building_Zone &zone, const Building_Zone &zonePrevious, const std::vector< double > &activities\)](#)
- void [stepPre \(const Building_Zone &zone, const Building_Zone &zonePrevious, const std::vector< double > &activities\)](#)
- void [actionStep \(int action, const Building_Zone &zone, bool inZone, bool preZone, const std::vector< double > &activities\)](#)
- bool [getDesiredWindowState \(\) const](#)
- bool [getDesiredLightState \(\) const](#)
- bool [isActionWindow \(\) const](#)

is there currently a window action
- bool [isActionLights \(\) const](#)

is there currently a Light action
- bool [isActionShades \(\) const](#)

is there currently a shade action
- bool [isActionHeatGains \(\) const](#)

- *is there currently a heat gain action*
 - bool `isActionLearning () const`
is there currently a learning action
 - bool `isActionAppliance () const`
 - int `getld () const`
 - int `getDesiredWindowDuration () const`
 - double `getDesiredShadeState () const`
 - double `getDesiredAppliance () const`
 - double `getDesiredHeatingSetPoint () const`
 - double `getPMV () const`
 - double `getHeatgains () const`

15.67.1 Detailed Description

The occupants understanding of a zone.

Contains all information about the occupants and their associated interactions with a zone

Definition at line 32 of file Occupant_Zone.hpp.

15.67.2 Constructor & Destructor Documentation

15.67.2.1 Occupant_Zone()

```
Occupant_Zone::Occupant_Zone ( )
```

Definition at line 12 of file Occupant_Zone.cpp.

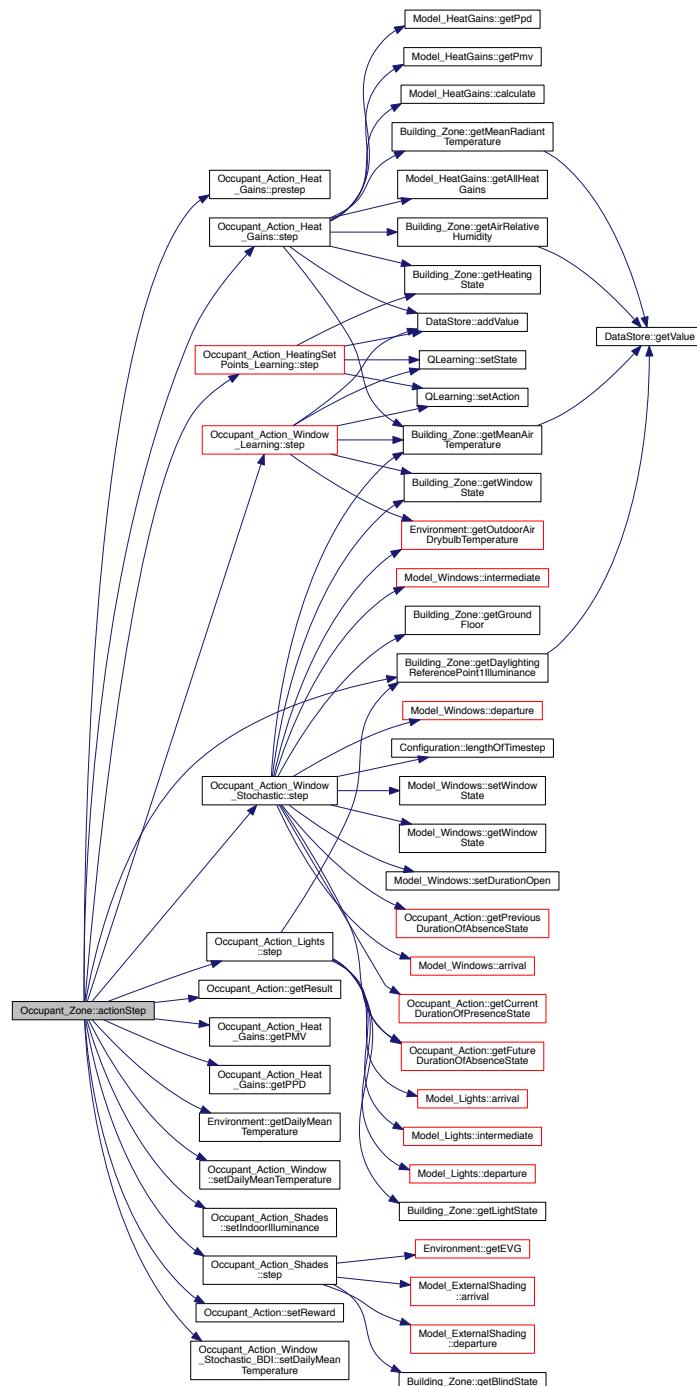
15.67.3 Member Function Documentation

15.67.3.1 actionStep()

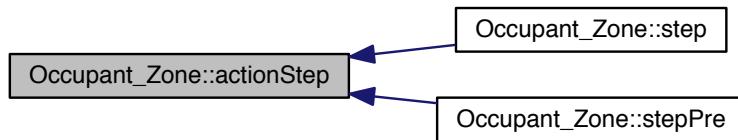
```
void Occupant_Zone::actionStep (
    int action,
    const Building_Zone & zone,
    bool inZone,
    bool preZone,
    const std::vector< double > & activities )
```

Definition at line 236 of file Occupant_Zone.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.67.3.2 `getDesiredAppliance()`

```
double Occupant_Zone::getDesiredAppliance ( ) const
```

Definition at line 376 of file Occupant_Zone.cpp.

15.67.3.3 `getDesiredHeatingSetPoint()`

```
double Occupant_Zone::getDesiredHeatingSetPoint ( ) const
```

Definition at line 333 of file Occupant_Zone.cpp.

15.67.3.4 `getDesiredLightState()`

```
bool Occupant_Zone::getDesiredLightState ( ) const
```

Definition at line 321 of file Occupant_Zone.cpp.

15.67.3.5 `getDesiredShadeState()`

```
double Occupant_Zone::getDesiredShadeState ( ) const
```

Definition at line 337 of file Occupant_Zone.cpp.

15.67.3.6 getDesiredWindowDuration()

```
int Occupant_Zone::getDesiredWindowDuration ( ) const
```

Definition at line 372 of file Occupant_Zone.cpp.

Here is the call graph for this function:



15.67.3.7 getDesiredWindowState()

```
bool Occupant_Zone::getDesiredWindowState ( ) const
```

Definition at line 317 of file Occupant_Zone.cpp.

15.67.3.8 getHeatgains()

```
double Occupant_Zone::getHeatgains ( ) const
```

Definition at line 329 of file Occupant_Zone.cpp.

15.67.3.9 getId()

```
int Occupant_Zone::getId ( ) const
```

Definition at line 313 of file Occupant_Zone.cpp.

15.67.3.10 getPMV()

```
double Occupant_Zone::getPMV ( ) const
```

Definition at line 325 of file Occupant_Zone.cpp.

15.67.3.11 isActionAppliance()

```
bool Occupant_Zone::isActionAppliance ( ) const
```

Definition at line 380 of file Occupant_Zone.cpp.

15.67.3.12 isActionHeatGains()

```
bool Occupant_Zone::isActionHeatGains ( ) const
```

is there currently a heat gain action

Returns

true if there is a action taking place

Definition at line 42 of file Occupant_Zone.cpp.

15.67.3.13 isActionLearning()

```
bool Occupant_Zone::isActionLearning ( ) const
```

is there currently a learning action

Returns

true if there is a action taking place

Definition at line 50 of file Occupant_Zone.cpp.

15.67.3.14 isActionLights()

```
bool Occupant_Zone::isActionLights ( ) const
```

is there currently a Light action

Returns

true if there is a action taking place

Definition at line 26 of file Occupant_Zone.cpp.

15.67.3.15 isActionShades()

```
bool Occupant_Zone::isActionShades ( ) const
```

is there currently a shade action

Returns

true if there is a action taking place

Definition at line 34 of file Occupant_Zone.cpp.

15.67.3.16 isActionWindow()

```
bool Occupant_Zone::isActionWindow ( ) const
```

is there currently a window action

Returns

true if there is a action taking place

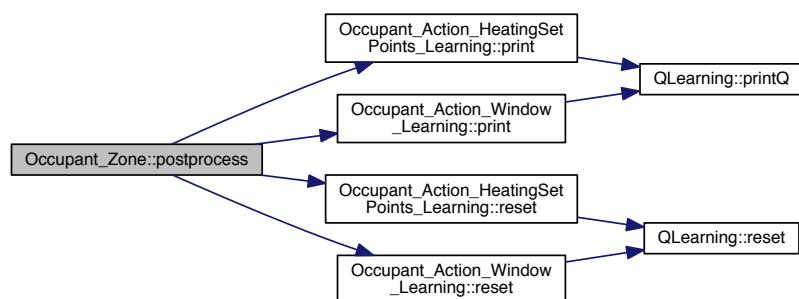
Definition at line 18 of file Occupant_Zone.cpp.

15.67.3.17 postprocess()

```
void Occupant_Zone::postprocess ( )
```

Definition at line 349 of file Occupant_Zone.cpp.

Here is the call graph for this function:



15.67.3.18 postTimeStep()

```
void Occupant_Zone::postTimeStep ( )
```

Definition at line 360 of file Occupant_Zone.cpp.

15.67.3.19 setClo()

```
void Occupant_Zone::setClo (
    double clo )
```

Definition at line 341 of file Occupant_Zone.cpp.

15.67.3.20 setMetabolicRate()

```
void Occupant_Zone::setMetabolicRate (
    double metabolicRate )
```

Definition at line 345 of file Occupant_Zone.cpp.

15.67.3.21 setup()

```
void Occupant_Zone::setup (
    int buildingID,
    const Building_Zone & buldingZone,
    int agentid,
    const ConfigStructAgent & agent )
```

Initialises the occupants understanding of a zone.

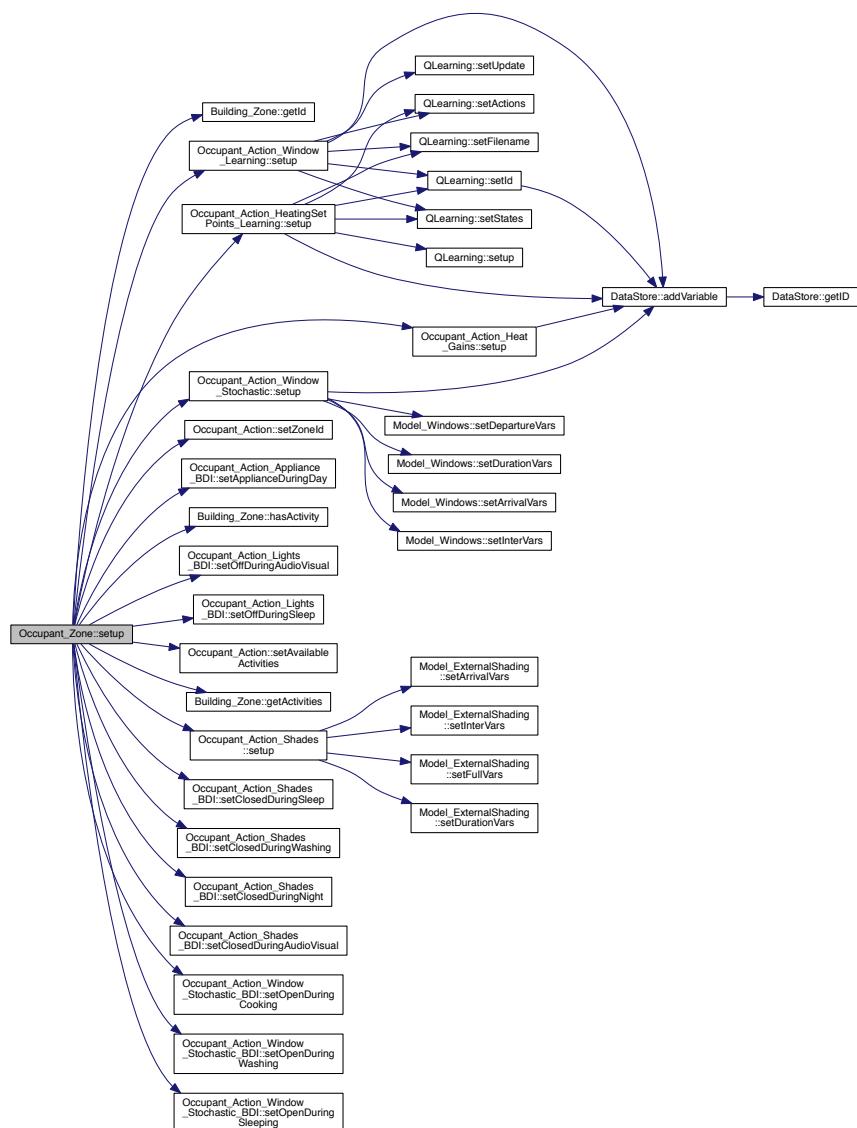
enables the different interactions that can take place in a zone

Parameters

<i>buildingID</i>	The id of the building the zone is in
<i>buldingZone</i>	The actual zone
<i>agentid</i>	The agents id
<i>agent</i>	The configuration struct for the agent

Definition at line 63 of file Occupant_Zone.cpp.

Here is the call graph for this function:



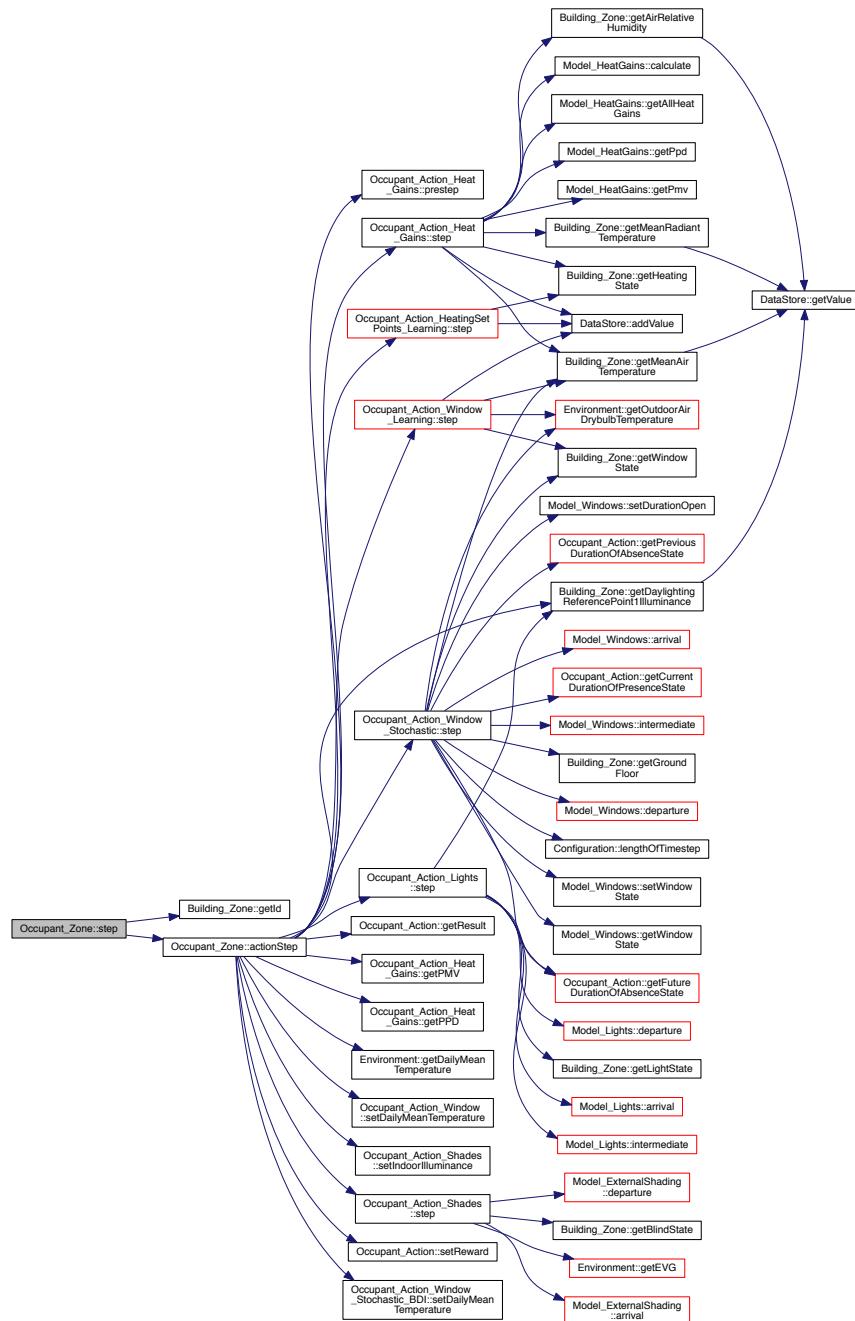
15.67.3.22 step()

```

void Occupant_Zone::step (
    const Building_Zone & zone,
    const Building_Zone & zonePrevious,
    const std::vector< double > & activities )
  
```

Definition at line 175 of file Occupant_Zone.cpp.

Here is the call graph for this function:

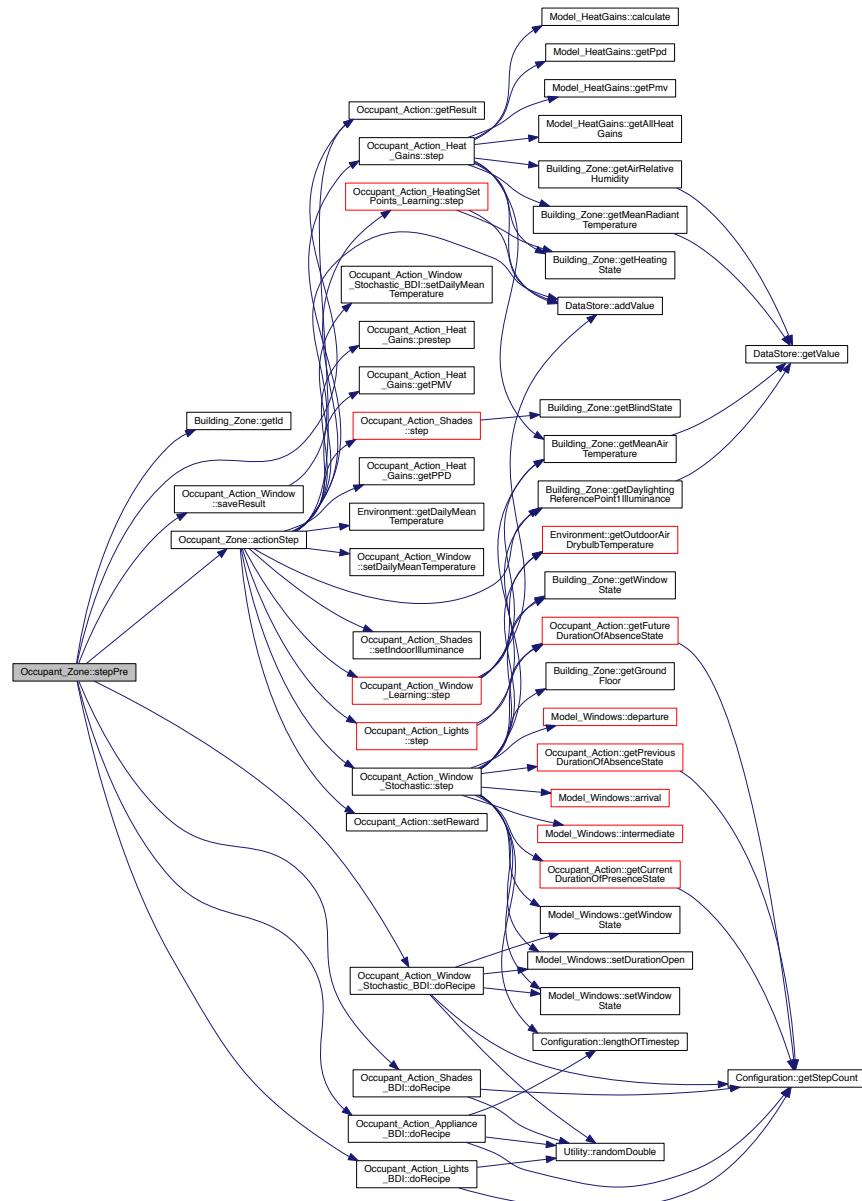


15.67.3.23 stepPre()

```
void Occupant_Zone::stepPre (
    const Building_Zone & zone,
    const Building_Zone & zonePrevious,
    const std::vector< double > & activities )
```

Definition at line 194 of file Occupant_Zone.cpp.

Here is the call graph for this function:



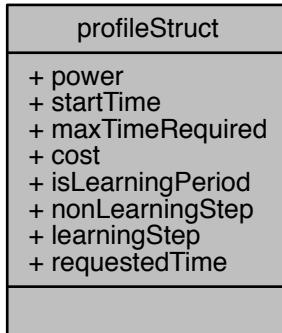
The documentation for this class was generated from the following files:

- Occupant_Zone.hpp
 - Occupant_Zone.cpp

15.68 profileStruct Struct Reference

```
#include <Appliance_Large_Learning.hpp>
```

Collaboration diagram for profileStruct:



Public Attributes

- std::vector< double > **power**
- int **startTime** = -1
- double **maxTimeRequired** = 0
- double **cost** = 0
- bool **isLearningPeriod** = false
- int **nonLearningStep** = 0
- unsigned int **learningStep** = 0
- int **requestedTime**

15.68.1 Detailed Description

Definition at line 12 of file Appliance_Large_Learning.hpp.

15.68.2 Member Data Documentation

15.68.2.1 cost

```
double profileStruct::cost = 0
```

Definition at line 16 of file Appliance_Large_Learning.hpp.

15.68.2.2 isLearningPeriod

```
bool profileStruct::isLearningPeriod = false
```

Definition at line 17 of file Appliance_Large_Learning.hpp.

15.68.2.3 learningStep

```
unsigned int profileStruct::learningStep = 0
```

Definition at line 19 of file Appliance_Large_Learning.hpp.

15.68.2.4 maxTimeRequired

```
double profileStruct::maxTimeRequired = 0
```

Definition at line 15 of file Appliance_Large_Learning.hpp.

15.68.2.5 nonLearningStep

```
int profileStruct::nonLearningStep = 0
```

Definition at line 18 of file Appliance_Large_Learning.hpp.

15.68.2.6 power

```
std::vector<double> profileStruct::power
```

Definition at line 13 of file Appliance_Large_Learning.hpp.

15.68.2.7 requestedTime

```
int profileStruct::requestedTime
```

Definition at line 20 of file Appliance_Large_Learning.hpp.

15.68.2.8 startTime

```
int profileStruct::startTime = -1
```

Definition at line 14 of file Appliance_Large_Learning.hpp.

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

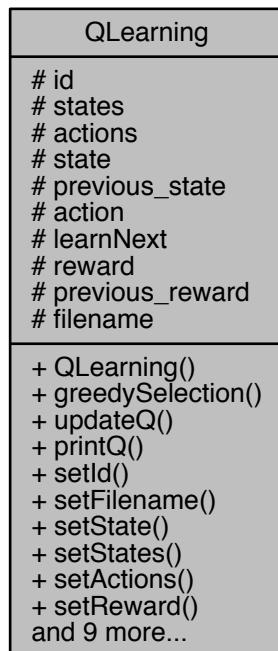
- Appliance_Large_Learning.hpp

15.69 QLearning Class Reference

The QLearning algorithm.

```
#include <QLearning.hpp>
```

Collaboration diagram for QLearning:



Public Member Functions

- `QLearning ()`
- `int greedySelection (const int s) const`
- `void updateQ (const int s, const int a, const double r, const int sp)`
- `void printQ () const`
- `virtual void setId (const int id)`
- `void setFilename (const std::string &filename)`
- `void setState (const int state)`
- `void setStates (const int states)`
- `void setActions (const int actions)`
- `void setReward (const double reward)`
- `void setEpsilon (const double epsilon)`
- `void setAlpha (double alpha)`
- `void setGamma (double gamma)`
- `void setUpdate (bool update)`
- `void setup ()`
- `void setAction (const double action)`
- `double getAction ()`
- `void learn ()`
- `virtual void reset ()`

Protected Attributes

- `int id`
- `int states = 0`
- `int actions = 0`
- `int state`
- `int previous_state = 0`
- `int action = 0`
- `bool learnNext = false`
- `double reward`
- `double previous_reward`
- `std::string filename`

15.69.1 Detailed Description

The QLearning algorithm.

C++ implementation of the QLearning algorithm

Definition at line 14 of file QLearning.hpp.

15.69.2 Constructor & Destructor Documentation

15.69.2.1 QLearning()

```
QLearning::QLearning ( )
```

Definition at line 16 of file QLearning.cpp.

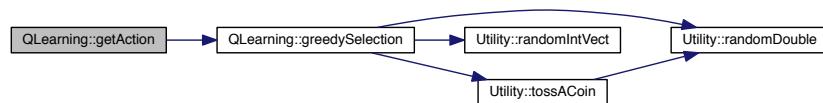
15.69.3 Member Function Documentation

15.69.3.1 getAction()

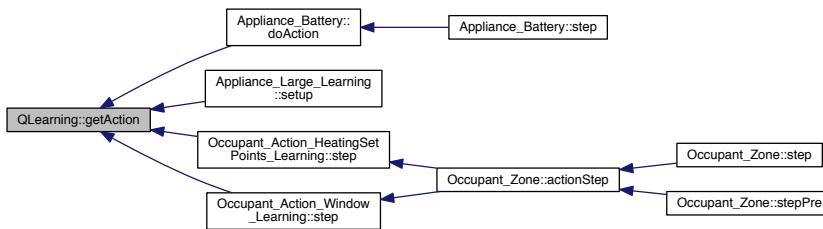
```
double QLearning::getAction ( )
```

Definition at line 125 of file QLearning.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

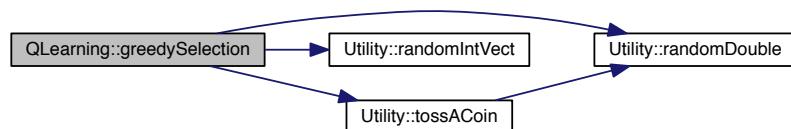


15.69.3.2 greedySelection()

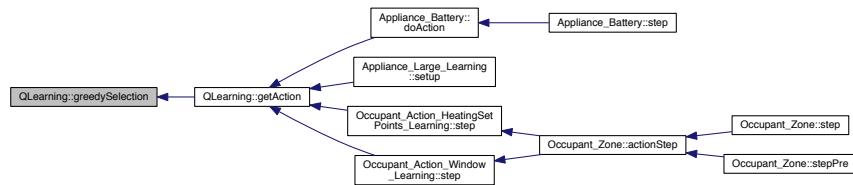
```
int QLearning::greedySelection (
    const int s ) const
```

Definition at line 46 of file QLearning.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

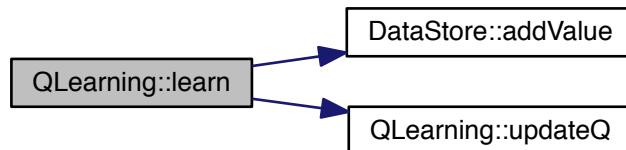


15.69.3.3 learn()

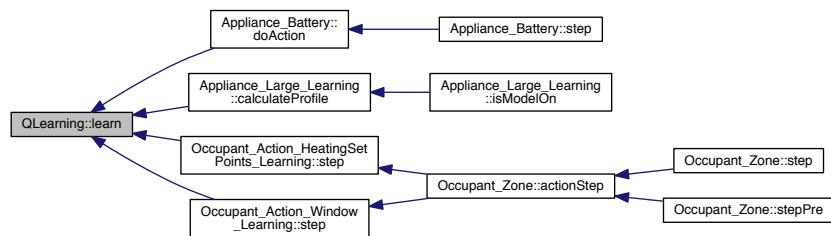
```
void QLearning::learn( )
```

Definition at line 117 of file QLearning.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

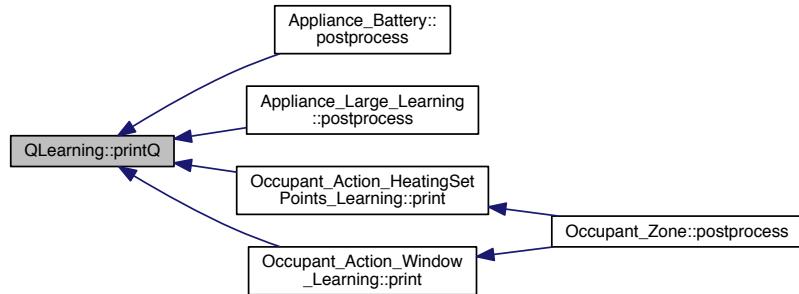


15.69.3.4 printQ()

```
void QLearning::printQ ( ) const
```

Definition at line 79 of file QLearning.cpp.

Here is the caller graph for this function:

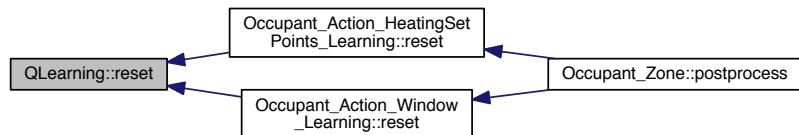


15.69.3.5 reset()

```
void QLearning::reset ( ) [virtual]
```

Definition at line 131 of file QLearning.cpp.

Here is the caller graph for this function:

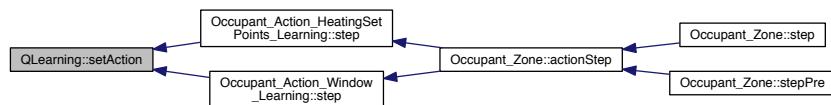


15.69.3.6 setAction()

```
void QLearning::setAction (
    const double action )
```

Definition at line 133 of file QLearning.cpp.

Here is the caller graph for this function:

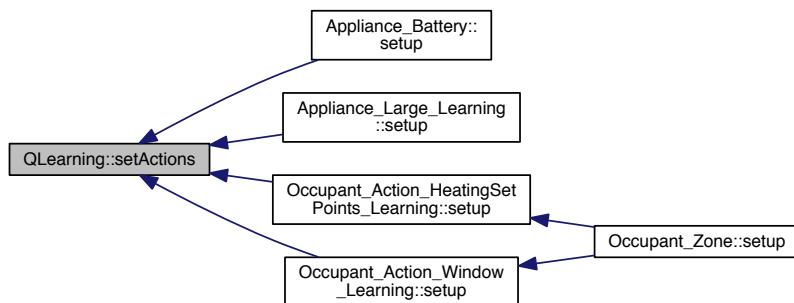


15.69.3.7 setActions()

```
void QLearning::setActions (
    const int actions )
```

Definition at line 109 of file QLearning.cpp.

Here is the caller graph for this function:

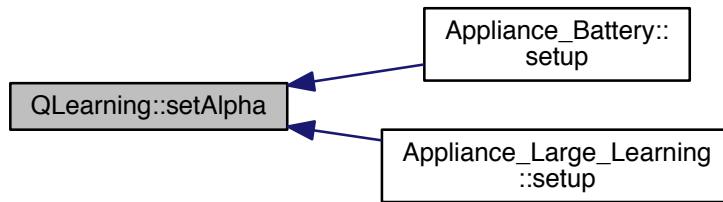


15.69.3.8 setAlpha()

```
void QLearning::setAlpha (
    double alpha )
```

Definition at line 145 of file QLearning.cpp.

Here is the caller graph for this function:

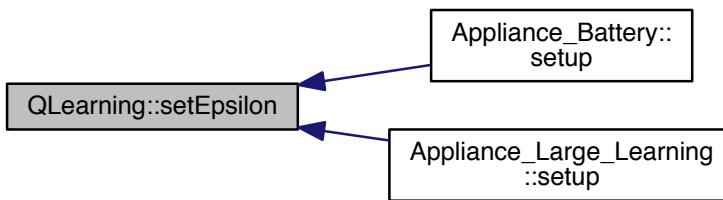


15.69.3.9 setEpsilon()

```
void QLearning::setEpsilon (
    const double epsilon )
```

Definition at line 137 of file QLearning.cpp.

Here is the caller graph for this function:

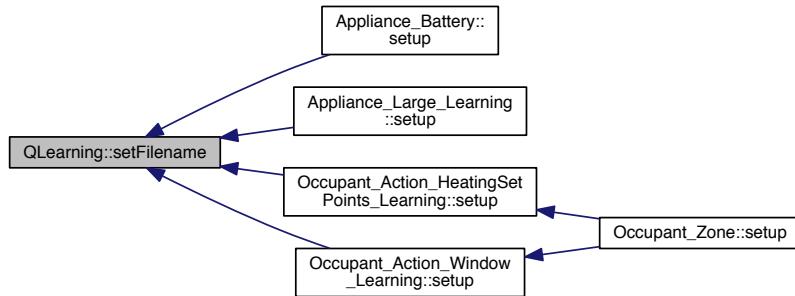


15.69.3.10 setFilename()

```
void QLearning::setFilename (
    const std::string & filename )
```

Definition at line 141 of file QLearning.cpp.

Here is the caller graph for this function:

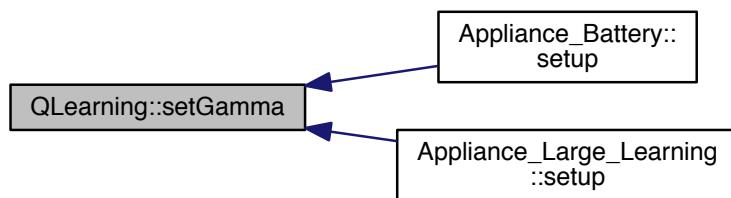


15.69.3.11 setGamma()

```
void QLearning::setGamma (
    double gamma )
```

Definition at line 149 of file QLearning.cpp.

Here is the caller graph for this function:

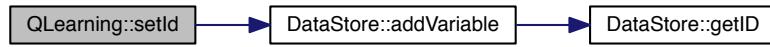


15.69.3.12 setId()

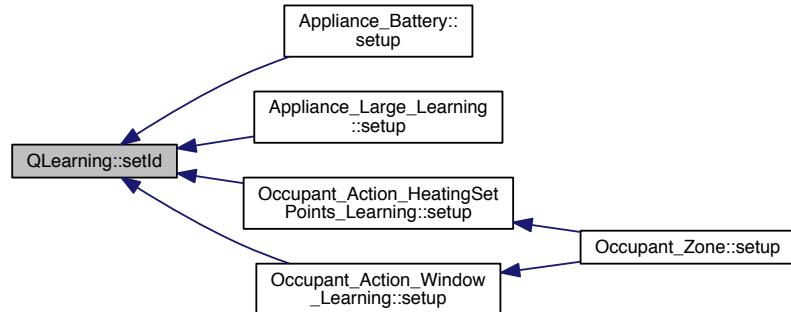
```
void QLearning::setId (
    const int id ) [virtual]
```

Definition at line 92 of file QLearning.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

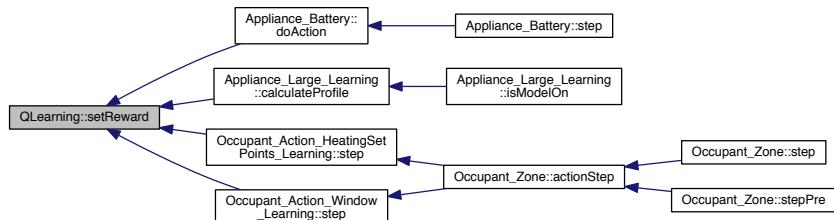


15.69.3.13 setReward()

```
void QLearning::setReward (
    const double reward )
```

Definition at line 113 of file QLearning.cpp.

Here is the caller graph for this function:

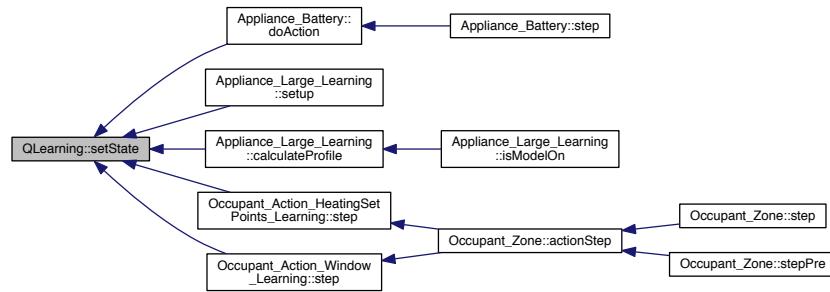


15.69.3.14 setState()

```
void QLearning::setState (
    const int state )
```

Definition at line 101 of file QLearning.cpp.

Here is the caller graph for this function:

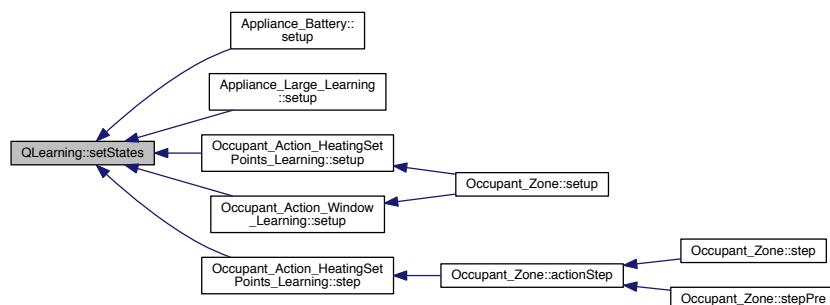


15.69.3.15 setStates()

```
void QLearning::setStates (
    const int states )
```

Definition at line 105 of file QLearning.cpp.

Here is the caller graph for this function:

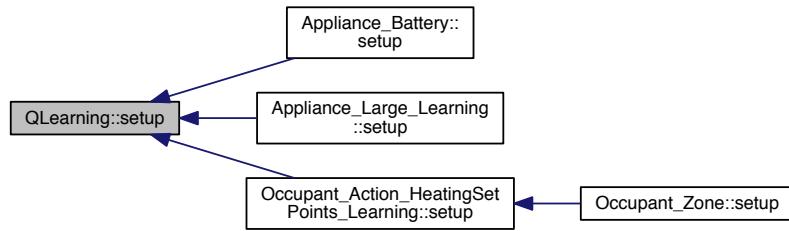


15.69.3.16 setup()

```
void QLearning::setup ( )
```

Definition at line 21 of file QLearning.cpp.

Here is the caller graph for this function:

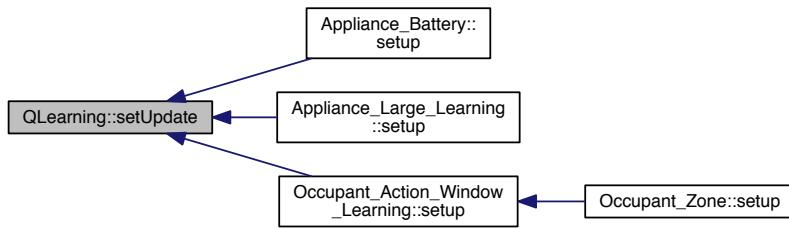


15.69.3.17 setUpdate()

```
void QLearning::setUpdate ( bool update )
```

Definition at line 153 of file QLearning.cpp.

Here is the caller graph for this function:

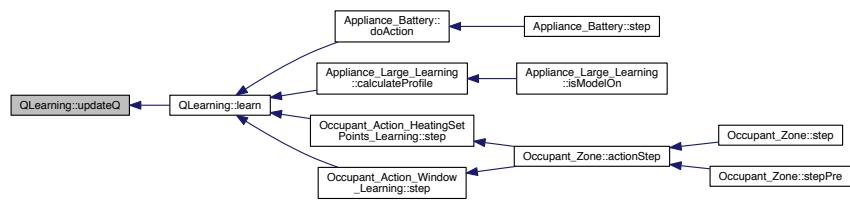


15.69.3.18 updateQ()

```
void QLearning::updateQ (
    const int s,
    const int a,
    const double r,
    const int sp )
```

Definition at line 71 of file QLearning.cpp.

Here is the caller graph for this function:



15.69.4 Member Data Documentation

15.69.4.1 action

```
int QLearning::action = 0 [protected]
```

Definition at line 42 of file QLearning.hpp.

15.69.4.2 actions

```
int QLearning::actions = 0 [protected]
```

Definition at line 39 of file QLearning.hpp.

15.69.4.3 filename

```
std::string QLearning::filename [protected]
```

Definition at line 46 of file QLearning.hpp.

15.69.4.4 id

```
int QLearning::id [protected]
```

Definition at line 37 of file QLearning.hpp.

15.69.4.5 learnNext

```
bool QLearning::learnNext = false [protected]
```

Definition at line 43 of file QLearning.hpp.

15.69.4.6 previous_reward

```
double QLearning::previous_reward [protected]
```

Definition at line 45 of file QLearning.hpp.

15.69.4.7 previous_state

```
int QLearning::previous_state = 0 [protected]
```

Definition at line 41 of file QLearning.hpp.

15.69.4.8 reward

```
double QLearning::reward [protected]
```

Definition at line 44 of file QLearning.hpp.

15.69.4.9 state

```
int QLearning::state [protected]
```

Definition at line 40 of file QLearning.hpp.

15.69.4.10 states

```
int QLearning::states = 0 [protected]
```

Definition at line 38 of file QLearning.hpp.

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

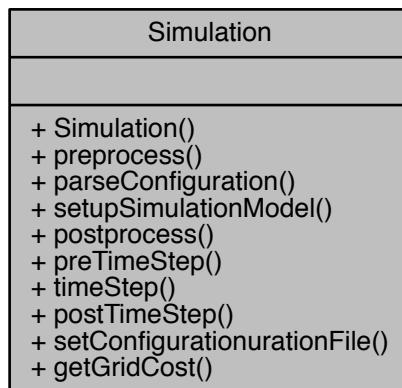
- QLearning.hpp
- QLearning.cpp

15.70 Simulation Class Reference

Main NoMASS simulation manager.

```
#include <Simulation.hpp>
```

Collaboration diagram for Simulation:



Public Member Functions

- [Simulation \(\)](#)
- [void preprocess \(\)](#)

Calls the simulation preprocess.
- [void parseConfiguration \(const std::string &file\)](#)
- [void setupSimulationModel \(\)](#)
- [void postprocess \(\)](#)

Calls the simulation postprocess.
- [void preTimeStep \(\)](#)

processes before timestep
- [void timeStep \(\)](#)

Increments the timestep for the simulation.
- [void postTimeStep \(\)](#)

processes After timestep
- [void setConfigurationFile \(const std::string &filename\)](#)

Static Public Member Functions

- static double [getGridCost \(\)](#)

15.70.1 Detailed Description

Main NoMASS simulation manager.

Called through the FMU interface, manages the simulation of the NoMass platform

Definition at line 16 of file Simulation.hpp.

15.70.2 Constructor & Destructor Documentation

15.70.2.1 Simulation()

```
Simulation::Simulation ( )
```

Definition at line 22 of file Simulation.cpp.

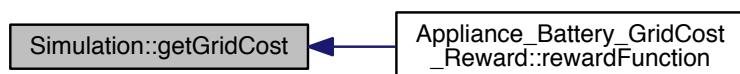
15.70.3 Member Function Documentation

15.70.3.1 getGridCost()

```
double Simulation::getGridCost ( ) [static]
```

Definition at line 18 of file Simulation.cpp.

Here is the caller graph for this function:

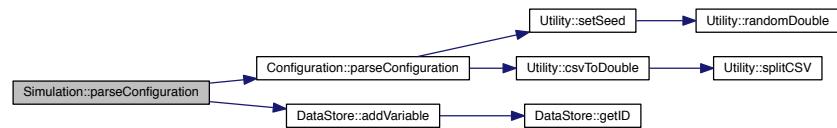


15.70.3.2 parseConfiguration()

```
void Simulation::parseConfiguration (
    const std::string & file )
```

Definition at line 43 of file Simulation.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



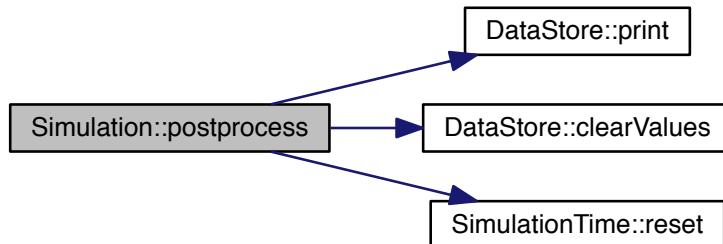
15.70.3.3 postprocess()

```
void Simulation::postprocess ( )
```

Calls the simulation postprocess.

Definition at line 63 of file Simulation.cpp.

Here is the call graph for this function:



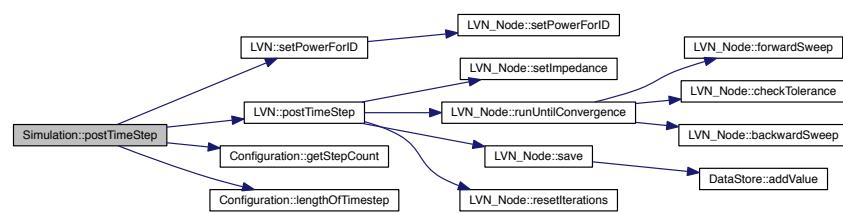
15.70.3.4 postTimeStep()

```
void Simulation::postTimeStep ( )
```

processes After timestep

Definition at line 142 of file Simulation.cpp.

Here is the call graph for this function:



15.70.3.5 preprocess()

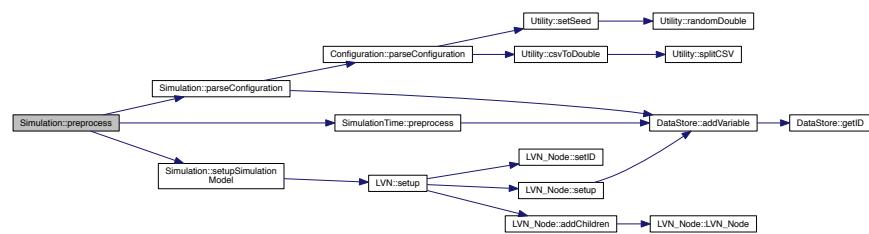
```
void Simulation::preprocess ( )
```

Calls the simulation preprocess.

Reads in the configuration file and sends to parser. Sets up the EnergyPlus processor, the AgentModel and the ZoneManager.

Definition at line 34 of file Simulation.cpp.

Here is the call graph for this function:

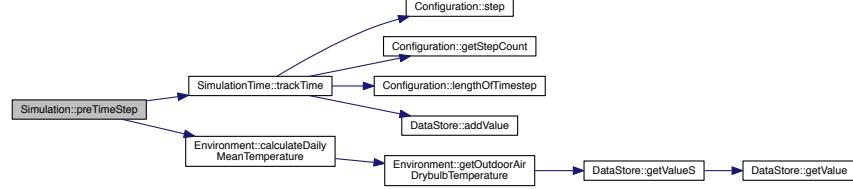


15.70.3.6 preTimeStep()

```
void Simulation::preTimeStep ( )
processes before timestep
```

Definition at line 75 of file Simulation.cpp.

Here is the call graph for this function:



15.70.3.7 setConfigurationurationFile()

```
void Simulation::setConfigurationurationFile (
    const std::string & filename )
```

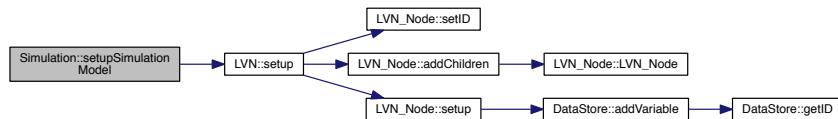
Definition at line 26 of file Simulation.cpp.

15.70.3.8 setupSimulationModel()

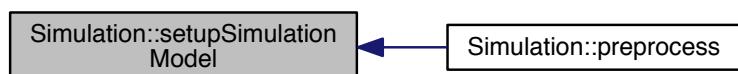
```
void Simulation::setupSimulationModel ( )
```

Definition at line 50 of file Simulation.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.70.3.9 timeStep()

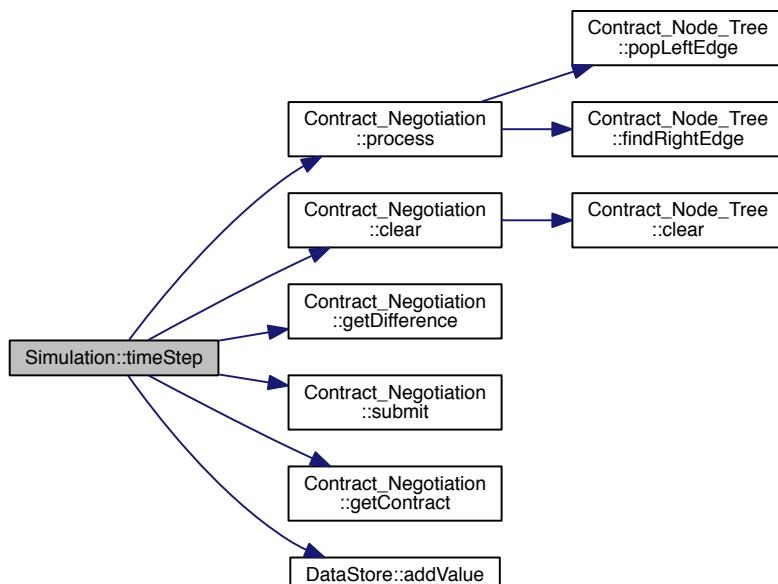
```
void Simulation::timeStep( )
```

Increments the timestep for the simulation.

Increments the timestep for the EnergyPlus processor, the AgentModel and the ZoneManager. Also we send any effects the agent have to the zones they are located in.

Definition at line 85 of file Simulation.cpp.

Here is the call graph for this function:



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

- `Simulation.hpp`
- `Simulation.cpp`

15.71 SimulationTime Class Reference

Keeps track of time.

```
#include <SimulationTime.hpp>
```

Collaboration diagram for SimulationTime:

SimulationTime
+ stepCount + databaseIdStepCount + minute + databaseIdMinute + hour + databaseIdHour + day + databaseIdDay + month + databaseIdMonth + hourOfDay + databaseIdHourOfDay + minuteOfDay + databaseIdMinuteOfDay
+ preprocess() + trackTime() + reset()

Static Public Member Functions

- static void [preprocess \(\)](#)
Calls the SimulationTime preprocess.
- static void [trackTime \(\)](#)
processes before timestep
- static void [reset \(\)](#)

Static Public Attributes

- static int [stepCount](#) = 0
- static int [databaseIdStepCount](#)
- static int [minute](#) = 0
- static int [databaseIdMinute](#)
- static int [hour](#) = 0
- static int [databaseIdHour](#)
- static int [day](#) = 0
- static int [databaseIdDay](#)
- static int [month](#) = 0
- static int [databaseIdMonth](#)
- static int [hourOfDay](#) = 0
- static int [databaseIdHourOfDay](#)
- static int [minuteOfDay](#) = 0
- static int [databaseIdMinuteOfDay](#)

15.71.1 Detailed Description

Keeps track of time.

Keeps track of time

Definition at line 13 of file SimulationTime.hpp.

15.71.2 Member Function Documentation

15.71.2.1 preprocess()

```
void SimulationTime::preprocess ( ) [static]
```

Calls the SimulationTime preprocess.

Reads in the configuration file and sends to parser. Sets up the EnergyPlus processor, the AgentModel and the ZoneManager.

Definition at line 41 of file SimulationTime.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.71.2.2 reset()

```
void SimulationTime::reset ( ) [static]
```

Definition at line 26 of file SimulationTime.cpp.

Here is the caller graph for this function:



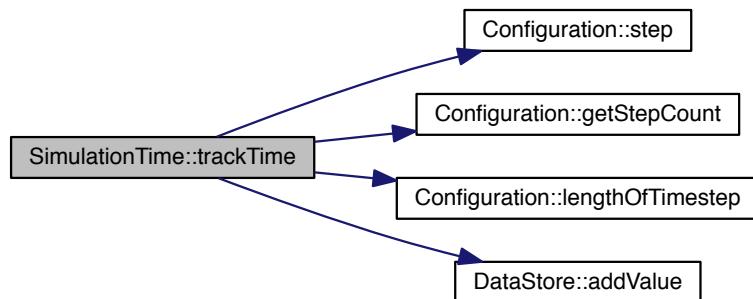
15.71.2.3 trackTime()

```
void SimulationTime::trackTime ( ) [static]
```

processes before timestep

Definition at line 65 of file SimulationTime.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.71.3 Member Data Documentation

15.71.3.1 databaseIdDay

```
int SimulationTime::databaseIdDay [static]
```

Definition at line 26 of file SimulationTime.hpp.

15.71.3.2 databaseIdHour

```
int SimulationTime::databaseIdHour [static]
```

Definition at line 24 of file SimulationTime.hpp.

15.71.3.3 databaseIdHourOfDay

```
int SimulationTime::databaseIdHourOfDay [static]
```

Definition at line 30 of file SimulationTime.hpp.

15.71.3.4 databaseIdMinute

```
int SimulationTime::databaseIdMinute [static]
```

Definition at line 22 of file SimulationTime.hpp.

15.71.3.5 databaseIdMinuteOfDay

```
int SimulationTime::databaseIdMinuteOfDay [static]
```

Definition at line 32 of file SimulationTime.hpp.

15.71.3.6 databaseIdMonth

```
int SimulationTime::databaseIdMonth [static]
```

Definition at line 28 of file SimulationTime.hpp.

15.71.3.7 databaseIdStepCount

```
int SimulationTime::databaseIdStepCount [static]
```

Definition at line 20 of file SimulationTime.hpp.

15.71.3.8 day

```
int SimulationTime::day = 0 [static]
```

Definition at line 25 of file SimulationTime.hpp.

15.71.3.9 hour

```
int SimulationTime::hour = 0 [static]
```

Definition at line 23 of file SimulationTime.hpp.

15.71.3.10 hourOfDay

```
int SimulationTime::hourOfDay = 0 [static]
```

Definition at line 29 of file SimulationTime.hpp.

15.71.3.11 minute

```
int SimulationTime::minute = 0 [static]
```

Definition at line 21 of file SimulationTime.hpp.

15.71.3.12 minuteOfDay

```
int SimulationTime::minuteOfDay = 0 [static]
```

Definition at line 31 of file SimulationTime.hpp.

15.71.3.13 month

```
int SimulationTime::month = 0 [static]
```

Definition at line 27 of file SimulationTime.hpp.

15.71.3.14 stepCount

```
int SimulationTime::stepCount = 0 [static]
```

Definition at line 19 of file SimulationTime.hpp.

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

- SimulationTime.hpp
- SimulationTime.cpp

15.72 State Class Reference

The state of an occupant.

```
#include <State.hpp>
```

Collaboration diagram for State:

State
id # metabolicRate # clo # activity # states # zone
+ State() + State() + ~State() + hasState() + getState() + addState() + setZonePtr() + getId() + setId() + setActivity() and 7 more...

Public Member Functions

- `State ()`
- `State (int id, double metabolicRate, double clo, const std::string &activity)`
- `virtual ~State ()`
- `virtual bool hasState (const int stateID) const`
- `virtual State getState (const int stateID) const`
- `void addState (State s)`
- `void setZonePtr (std::shared_ptr< Building_Zone > zoneptr)`
- `int getId () const`
- `void setId (int id)`
- `void setActivity (const std::string &activity)`
- `void setMetabolicRate (double metabolicRate)`
- `void setClo (double clo)`
- `unsigned int getNumberOfSubStates () const`
- `double getMetabolicRate () const`
- `double getClo () const`
- `bool isInActivity (const std::string &activity) const`
- `std::shared_ptr< Building_Zone > getZonePtr () const`

Protected Attributes

- `int id`
- `double metabolicRate`
- `double clo`
- `std::string activity`
- `std::vector< State > states`
- `std::shared_ptr< Building_Zone > zone`

15.72.1 Detailed Description

The state of an occupant.

The state of an occupant based on the activity model

Definition at line 15 of file State.hpp.

15.72.2 Constructor & Destructor Documentation

15.72.2.1 State() [1/2]

`State::State ()`

Definition at line 8 of file State.cpp.

15.72.2.2 State() [2/2]

```
State::State (
    int id,
    double metabolicRate,
    double clo,
    const std::string & activity )
```

Definition at line 11 of file State.cpp.

15.72.2.3 ~State()

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

Definition at line 9 of file State.cpp.

15.72.3 Member Function Documentation**15.72.3.1 addState()**

```
void State::addState (
    State s )
```

Definition at line 53 of file State.cpp.

Here is the caller graph for this function:



15.72.3.2 getClo()

```
double State::getClo ( ) const
```

Definition at line 45 of file State.cpp.

Here is the caller graph for this function:

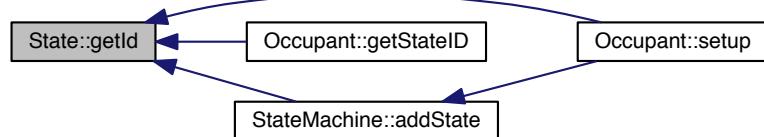


15.72.3.3 getId()

```
int State::getId ( ) const
```

Definition at line 37 of file State.cpp.

Here is the caller graph for this function:



15.72.3.4 getMetabolicRate()

```
double State::getMetabolicRate ( ) const
```

Definition at line 41 of file State.cpp.

Here is the caller graph for this function:



15.72.3.5 getState()

```
State State::getState (
    const int stateID ) const [virtual]
```

Definition at line 68 of file State.cpp.

15.72.3.6 getZonePtr()

```
std::shared_ptr< Building_Zone > State::getZonePtr ( ) const
```

Definition at line 82 of file State.cpp.

Here is the caller graph for this function:



15.72.3.7 hasState()

```
bool State::hasState (
    const int stateID ) const [virtual]
```

Definition at line 57 of file State.cpp.

15.72.3.8 `isInActivity()`

```
bool State::isInActivity (
    const std::string & activity ) const
```

Definition at line 49 of file State.cpp.

15.72.3.9 `numberOfSubStates()`

```
unsigned int State::numberOfSubStates ( ) const
```

Definition at line 17 of file State.cpp.

15.72.3.10 `setActivity()`

```
void State::setActivity (
    const std::string & activity )
```

Definition at line 33 of file State.cpp.

15.72.3.11 `setClo()`

```
void State::setClo (
    double clo )
```

Definition at line 29 of file State.cpp.

15.72.3.12 `setId()`

```
void State::setId (
    int id )
```

Definition at line 21 of file State.cpp.

15.72.3.13 `setMetabolicRate()`

```
void State::setMetabolicRate (
    double metabolicRate )
```

Definition at line 25 of file State.cpp.

15.72.3.14 setZonePtr()

```
void State::setZonePtr ( std::shared_ptr< Building_Zone > zoneptr )
```

Definition at line 85 of file State.cpp.

Here is the caller graph for this function:



15.72.4 Member Data Documentation

15.72.4.1 activity

```
std::string State::activity [protected]
```

Definition at line 40 of file State.hpp.

15.72.4.2 clo

```
double State::clo [protected]
```

Definition at line 39 of file State.hpp.

15.72.4.3 id

```
int State::id [protected]
```

Definition at line 37 of file State.hpp.

15.72.4.4 metabolicRate

```
double State::metabolicRate [protected]
```

Definition at line 38 of file State.hpp.

15.72.4.5 states

```
std::vector<State> State::states [protected]
```

Definition at line 41 of file State.hpp.

15.72.4.6 zone

```
std::shared_ptr<Building_Zone> State::zone [protected]
```

Definition at line 42 of file State.hpp.

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

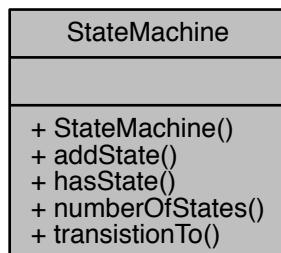
- State.hpp
- State.cpp

15.73 StateMachine Class Reference

Moves an agent into a different state.

```
#include <StateMachine.hpp>
```

Collaboration diagram for StateMachine:



Public Member Functions

- `StateMachine ()`
- `void addState (const State &s)`
- `bool hasState (const int stateID) const`
- `unsigned int numberOfStates () const`
- `State transitionTo (const int stateID) const`

15.73.1 Detailed Description

Moves an agent into a different state.

Moves an agent into a different state

Definition at line 12 of file StateMachine.hpp.

15.73.2 Constructor & Destructor Documentation

15.73.2.1 StateMachine()

```
StateMachine::StateMachine ( )
```

Definition at line 6 of file StateMachine.cpp.

15.73.3 Member Function Documentation

15.73.3.1 addState()

```
void StateMachine::addState (
    const State & s )
```

Definition at line 12 of file StateMachine.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

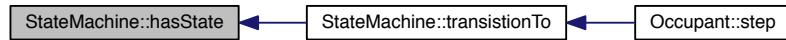


15.73.3.2 hasState()

```
bool StateMachine::hasState (
    const int stateID ) const
```

Definition at line 16 of file StateMachine.cpp.

Here is the caller graph for this function:



15.73.3.3 numberOfStates()

```
unsigned int StateMachine::numberOfStates ( ) const
```

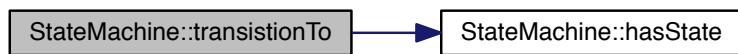
Definition at line 8 of file StateMachine.cpp.

15.73.3.4 `transistionTo()`

```
State StateMachine::transistionTo (
    const int stateID ) const
```

Definition at line 29 of file StateMachine.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



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

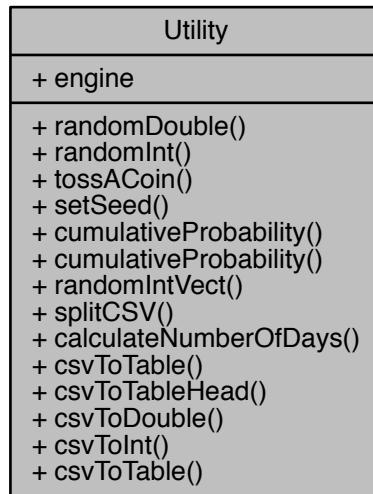
- StateMachine.hpp
- StateMachine.cpp

15.74 Utility Class Reference

Utility functions.

```
#include <Utility.hpp>
```

Collaboration diagram for Utility:



Public Types

- template<typename T >
using `uTable` = std::vector< std::vector< T > >

Static Public Member Functions

- static double `randomDouble` (double min, double max)
- static int `randomInt` (int min, int max)
- static bool `tossACoin` ()
- static void `setSeed` (int seed)
- static int `cumulativeProbability` (const std::vector< double > &v)
- static int `cumulativeProbability` (const double *v, const int len)
- static std::vector< int > `randomIntVect` (int number)
- static void `splitCSV` (const std::string &typeString, std::vector< std::string > *types)
- static int `calculateNumberOfDays` (int startDay, int startMonth, int endDay, int endMonth)
- template<typename T >
static `uTable`< T > `csvToTable` (const std::string &filename, bool header)
- static std::vector< std::string > `csvToTableHead` (const std::string &filename)
- static std::vector< double > `csv.ToDouble` (const std::string &s)
- static std::vector< int > `csvToInt` (const std::string &s)
- template<typename T >
static `uTable`< T > `csvToTable` (const std::string &filename)

Static Public Attributes

- static std::mt19937_64 `engine`

15.74.1 Detailed Description

Utility functions.

Utility functions

Definition at line 19 of file Utility.hpp.

15.74.2 Member Typedef Documentation

15.74.2.1 uTable

```
template<typename T >
using Utility::uTable = std::vector<std::vector<T> >
```

Definition at line 22 of file Utility.hpp.

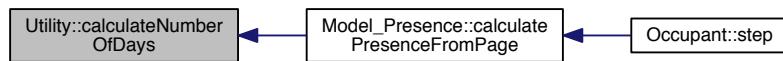
15.74.3 Member Function Documentation

15.74.3.1 calculateNumberOfDays()

```
int Utility::calculateNumberOfDays (
    int startDay,
    int startMonth,
    int endDay,
    int endMonth ) [static]
```

Definition at line 59 of file Utility.cpp.

Here is the caller graph for this function:

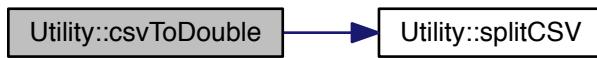


15.74.3.2 csvToDouble()

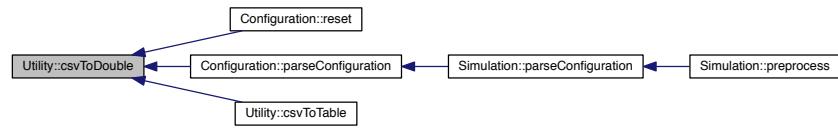
```
std::vector< double > Utility::csvToDouble (
    const std::string & s ) [static]
```

Definition at line 103 of file Utility.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.74.3.3 csvToInt()

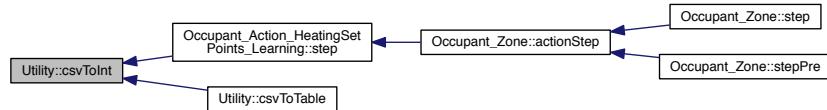
```
std::vector< int > Utility::csvToInt (
    const std::string & s ) [static]
```

Definition at line 113 of file Utility.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

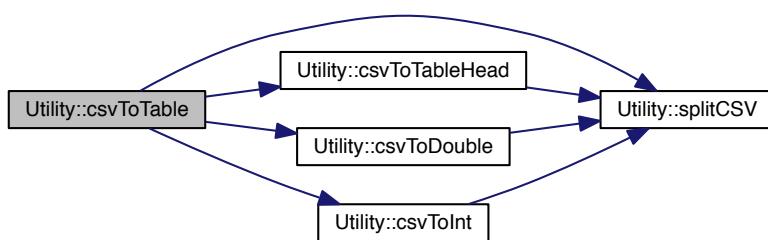


15.74.3.4 csvToTable() [1/2]

```
template<typename T >
static uTable<T> Utility::csvToTable (
    const std::string & filename,
    bool header ) [inline], [static]
```

Definition at line 35 of file Utility.hpp.

Here is the call graph for this function:



15.74.3.5 csvToTable() [2/2]

```
template<typename T >
static uTable<T> Utility::csvToTable (
    const std::string & filename ) [inline], [static]
```

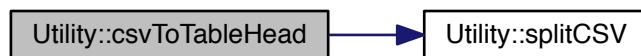
Definition at line 63 of file Utility.hpp.

15.74.3.6 csvToTableHead()

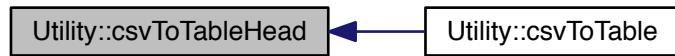
```
std::vector< std::string > Utility::csvToTableHead (
    const std::string & filename )  [static]
```

Definition at line 91 of file Utility.cpp.

Here is the call graph for this function:



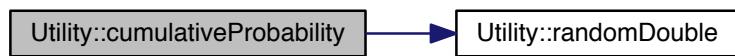
Here is the caller graph for this function:

**15.74.3.7 cumulativeProbability() [1/2]**

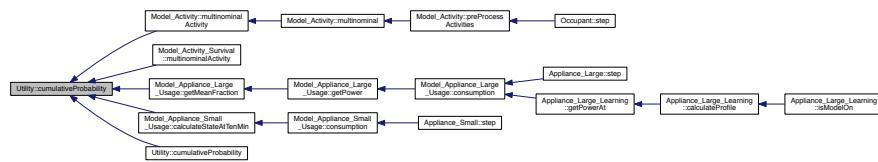
```
int Utility::cumulativeProbability (
    const std::vector< double > & v )  [static]
```

Definition at line 77 of file Utility.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

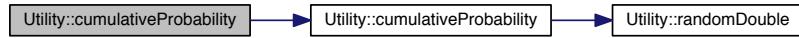


15.74.3.8 cumulativeProbability() [2/2]

```
int Utility::cumulativeProbability (
```

Definition at line 73 of file Utility.cpp.

Here is the call graph for this function:



15.74.3.9 randomDouble()

```
double Utility::randomDouble (
```



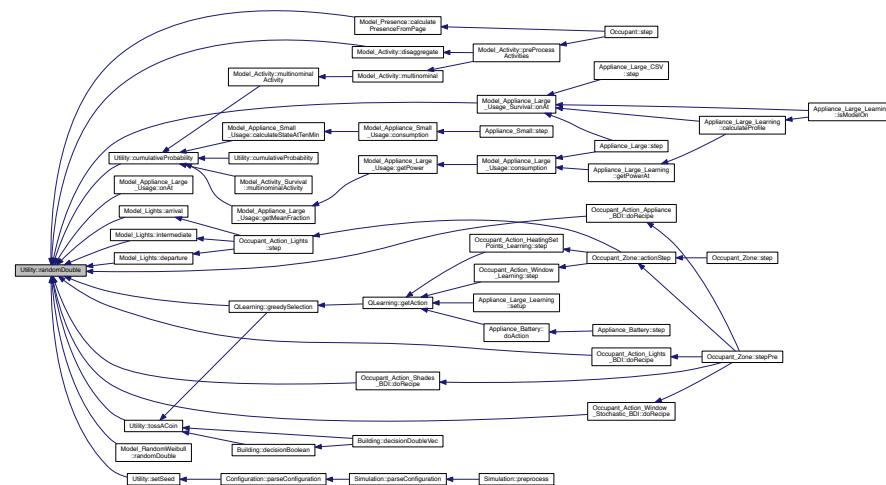
```
    double min,
```



```
    double max ) [static]
```

Definition at line 25 of file Utility.cpp.

Here is the caller graph for this function:

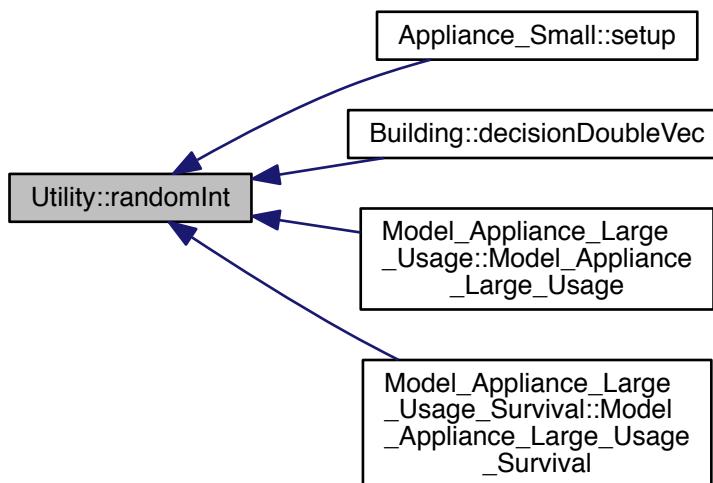


15.74.3.10 randomInt()

```
int Utility::randomInt (
    int min,
    int max ) [static]
```

Definition at line 34 of file Utility.cpp.

Here is the caller graph for this function:

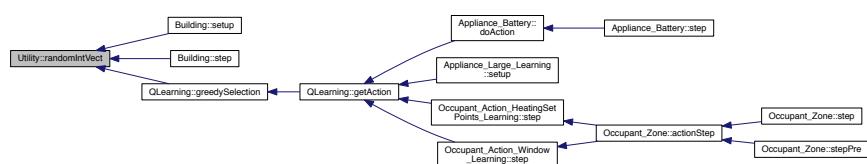


15.74.3.11 randomIntVect()

```
std::vector< int > Utility::randomIntVect (
    int number ) [static]
```

Definition at line 39 of file Utility.cpp.

Here is the caller graph for this function:

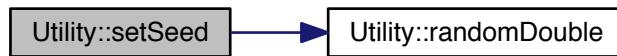


15.74.3.12 setSeed()

```
void Utility::setSeed (
    int seed ) [static]
```

Definition at line 20 of file Utility.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

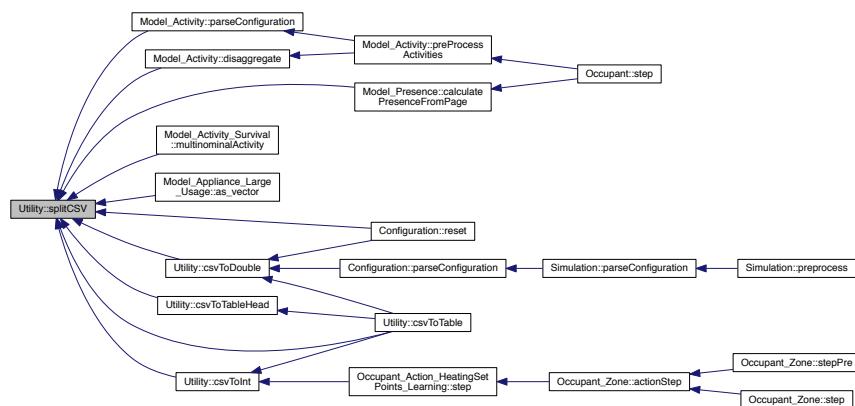


15.74.3.13 splitCSV()

```
void Utility::splitCSV (
    const std::string & typeString,
    std::vector< std::string > * types ) [static]
```

Definition at line 46 of file Utility.cpp.

Here is the caller graph for this function:



15.74.3.14 tossACoin()

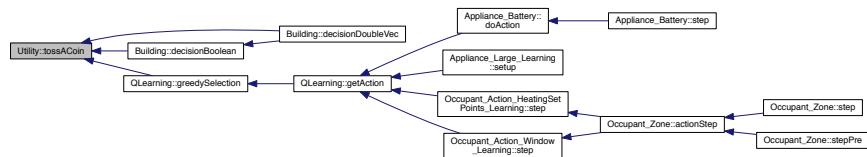
```
bool Utility::tossACoin ( ) [static]
```

Definition at line 30 of file Utility.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



15.74.4 Member Data Documentation

15.74.4.1 engine

```
std::mt19937_64 Utility::engine [static]
```

Definition at line 66 of file Utility.hpp.

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

- Utility.hpp
- Utility.cpp

Index

~State
 State, [476](#)

a01arr
 ConfigStructShade, [206](#)
 ConfigStructWindow, [215](#)

a01dep
 ConfigStructWindow, [215](#)

a01int
 ConfigStructShade, [206](#)
 ConfigStructWindow, [215](#)

a10arr
 ConfigStructShade, [206](#)

a10dep
 ConfigStructWindow, [216](#)

a10int
 ConfigStructShade, [206](#)

aSFlower
 ConfigStructShade, [207](#)

action
 Appliance_Battery, [91](#)
 QLearning, [461](#)

actionStep
 Occupant_Zone, [438](#)

actions
 QLearning, [461](#)

active
 ConfigStructZone, [220](#)

Activities
 Appliance, [71](#)

activities
 ConfigStructAppliance, [196](#)
 ConfigStructZone, [220](#)

activity
 State, [480](#)

activityAvailable
 Occupant_Action, [377](#)

addChildren
 LVN_Node, [273](#)

addContactsTo
 Building, [159](#)
 Building_Appliances, [168](#)

AddCost
 Appliance_Battery, [80](#)

addCurrentStates
 Building_Appliances, [169](#)

addGlobalContactsTo
 Appliance_Group, [110](#)

addNode
 LVN_Node, [274](#)

addState
 State, [476](#)
 StateMachine, [482](#)

addToCost
 Appliance_Large_Learning, [142](#)

addValue
 DataStore, [252](#)

addValueS
 DataStore, [253](#)

addVariable
 DataStore, [253](#)

afulllower
 ConfigStructShade, [206](#)

afullraise
 ConfigStructShade, [206](#)

age
 ConfigStructAgent, [190](#)
 Model_Activity, [292](#)

Agent, [45](#)
 Agent, [48](#)
 buildingID, [50](#)
 getBuildingID, [48](#)
 getID, [48](#)
 id, [50](#)
 idString, [50](#)
 postTimeStep, [48](#)
 postprocess, [48](#)
 preprocess, [48](#)
 setBuildingID, [49](#)
 setIDString, [49](#)
 setID, [49](#)
 setup, [50](#)
 step, [50](#)

agentHeatGains
 ConfigStructSimulation, [210](#)

agents
 ConfigStructBuilding, [201](#)

allocateMemory
 fmiCallbackFunctions, [262](#)

alpha
 ConfigStructAppliance, [196](#)

aop
 ConfigStructWindow, [216](#)

Appliance, [51](#)
 Activities, [71](#)
 Appliance, [55](#)
 beforeClear, [56](#)
 calculateHourOfDay, [56](#)
 clear, [56](#)

datastoreGridIDCost, 72
 datastoreGridIDReceived, 72
 datastoreGridIDRequested, 72
 datastoreGridIDSupplied, 72
 datastoreGridIDSuppliedCost, 72
 datastoreIDCost, 72
 datastoreIDReceived, 73
 datastoreIDRequested, 73
 datastoreIDSupplied, 73
 datastoreIDSuppliedCost, 73
 datastoreLocalIDCost, 73
 datastoreLocalIDReceived, 73
 datastoreLocalIDRequested, 74
 datastoreLocalIDSupplied, 74
 datastoreLocalIDSuppliedCost, 74
 datastoreNeighbourhoodIDCost, 74
 datastoreNeighbourhoodIDReceived, 74
 datastoreNeighbourhoodIDRequested, 74
 datastoreNeighbourhoodIDSupplied, 75
 datastoreNeighbourhoodIDSuppliedCost, 75
 getGridPower, 57
 getGridReceived, 57
 getGridReceivedCost, 57
 getGridSupply, 57
 getGridSupplyLeft, 57
 getLocalPower, 58
 getLocalReceived, 58
 getLocalReceivedCost, 58
 getLocalSupply, 58
 getLocalSupplyLeft, 58
 getNeighbourhoodPower, 58
 getNeighbourhoodReceived, 59
 getNeighbourhoodReceivedCost, 59
 getNeighbourhoodSupply, 59
 getNeighbourhoodSupplyLeft, 59
 getPower, 59
 getPriority, 60
 getReceived, 60
 getReceivedCost, 61
 getSupply, 61
 getSupplyCost, 62
 getSupplyLeft, 62
 global, 75
 hasActivities, 62
 hourlyCost, 75
 hourlyPriority, 75
 isGlobal, 62
 isLocal, 63
 local, 75
 match, 76
 parameters, 76
 parametersGrid, 76
 parametersLocal, 76
 parametersNeighbourhood, 76
 save, 63
 saveGlobal, 63
 saveGlobalCalculate, 63
 saveLocal, 64
 saveLocalCalculate, 64
 saveNeighbourhood, 65
 saveNeighbourhoodCalculate, 65
 setActivities, 65
 setGlobal, 66
 setHourlyPriority, 66
 setHourlyCost, 67
 setLocal, 67
 setPower, 68
 setReceived, 68
 setReceivedCost, 69
 setSupply, 69
 setSupplyCost, 70
 setSupplyLeft, 70
 setup, 71
 setupSave, 71
Appliance_Battery, 77
 action, 91
 AddCost, 80
Appliance_Battery, 80
 BatteryDeltaT, 91
 batteryNeighbourhoodCharge, 91
 batteryNeighbourhoodDischarge, 91
 calculateDeltaE, 80
 calculateStateOfCharge, 80
 calculateSupply, 81
 capacity, 92
 chargeRate, 92
 clear, 82
 cost, 92
 datastoreIDstateOfCharge, 92
 dischargeRate, 92
 doAction, 82
 efficiency, 92
 energy_calc, 83
 get_charge_delta, 84
 get_new_SOC_charge, 84
 get_new_SOC_discharge, 85
 getStateOfCharge, 86
 mostShortage, 93
 postprocess, 86
 powerAvailable, 87
 powerShortage, 93
 previousHourOfDay, 93
 qLearning, 93
 rewardFunction, 87
 saveGlobalCalculate, 87
 saveNeighbourhoodCalculate, 88
 setBatteryNeighbourhoodCharge, 88
 setBatteryNeighbourhoodDischarge, 88
 setPowerShortage, 88
 setup, 89
 setupModel, 89
 stateOfCharge, 93
 step, 90
 stepNeighbourhood, 90
 sumShort, 93
 sumSupply, 94

Appliance_Battery_GridCost_Reward, 94
Appliance_Battery_GridCost_Reward, 97
rewardFunction, 97

Appliance_FMI, 98
Appliance_FMI, 100
setFMIVariableName, 100
setup, 100
step, 101

Appliance_Generic_CSV, 102
Appliance_Generic_CSV, 104
setFileDemand, 104
setFileSupply, 105
setup, 105
step, 106

Appliance_Group
addGlobalContactsTo, 110
Appliance_Group, 110
appliances, 121
clear, 111
datastoreIDCost, 121
datastoreIDReceived, 121
datastoreIDRequested, 121
datastoreIDSupplied, 122
datastoreIDSuppliedCost, 122
getApplianceAt, 111
getPower, 111
getReceived, 112
getReceivedCost, 112
getSupply, 113
getSupplyCost, 113
getSupplyLeft, 113
globalContracts, 122
globalNegotiation, 114
hasActivities, 114
idString, 122
localNegotiation, 114
negotiationApp, 115
negotiationAppGlobal, 115
neighbourhoodNegotiation, 116
postprocess, 116
sendCondition, 117
sendContractGlobal, 117
sendContractLocal, 118
setIdString, 118
setup, 119
setupSave, 119
shuffleAppliances, 120
step, 120
stepApp, 120

Appliance_Group< T >, 107

Appliance_Group_Battery
Appliance_Group_Battery, 125
getPowerShortage, 125
globalCost, 125
negotiationApp, 126
negotiationAppGlobal, 126
neighbourhoodNegotiationBattery, 126
sendCondition, 127

setPowerShortage, 127
stepApp, 127

Appliance_Group_Battery< T >, 123

Appliance_Large, 128
Appliance_Large, 131
file, 134
isOn, 131
model, 134
profileCSV, 135
setFile, 132
setup, 132
setupModel, 133
step, 134

Appliance_Large_CSV, 135
Appliance_Large_CSV, 138
count, 139
running, 139
setupModel, 138
step, 138

Appliance_Large_Learning, 140
addToCost, 142
Appliance_Large_Learning, 142
calculateProfile, 143
getPowerAt, 143
getRequiredTime, 144
isModelOn, 144
postprocess, 145
powerProfile, 147
setHourlyTimeRequired, 145
setup, 146
step, 147

Appliance_Large_Learning_CSV, 148
Appliance_Large_Learning_CSV, 150
calculateProfile, 150
setupModel, 150

Appliance_Small, 151
Appliance_Small, 154
setup, 154
step, 155

ApplianceDuringDay
ConfigStructAgent, 190

ApplianceParameters, 156
power, 156
received, 157
receivedCost, 157
suppliedLeft, 157
supply, 157
supplyCost, 157

appliances
Appliance_Group, 121

AppliancesBattery
ConfigStructBuilding, 201

AppliancesBatteryGrid
ConfigStructBuilding, 201

AppliancesCSV
ConfigStructBuilding, 201

AppliancesFMI
ConfigStructBuilding, 201

AppliancesGrid
 ConfigStructBuilding, 201
 AppliancesLarge
 ConfigStructBuilding, 202
 AppliancesLargeCSV
 ConfigStructBuilding, 202
 AppliancesLargeLearning
 ConfigStructBuilding, 202
 AppliancesLargeLearningCSV
 ConfigStructBuilding, 202
 AppliancesSmall
 ConfigStructBuilding, 202
 arrival
 Model_ExternalShading, 329
 Model_Lights, 338
 Model_Windows, 349
 as_vector
 Model_Appliance_Large_Usage, 301
 as_vector_vector
 Model_Appliance_Large_Usage, 301
 at
 Model_Presence, 341
 availableActivities
 Occupant_Action, 381

 b
 ModellInstance, 359
 b01absdep
 ConfigStructWindow, 216
 b01absprevar
 ConfigStructWindow, 216
 b01gddep
 ConfigStructWindow, 216
 b01inarr
 ConfigStructShade, 207
 ConfigStructWindow, 216
 b01init
 ConfigStructShade, 207
 ConfigStructWindow, 217
 b01outarr
 ConfigStructWindow, 217
 b01outdep
 ConfigStructWindow, 217
 b01outint
 ConfigStructWindow, 217
 b01presint
 ConfigStructWindow, 217
 b01rnarr
 ConfigStructWindow, 217
 b01rnint
 ConfigStructWindow, 218
 b01sarr
 ConfigStructShade, 207
 b01sint
 ConfigStructShade, 207
 b10absdep
 ConfigStructWindow, 218
 b10gddep
 ConfigStructWindow, 218

 b10inarr
 ConfigStructShade, 207
 b10indep
 ConfigStructWindow, 218
 b10inint
 ConfigStructShade, 208
 b10outdep
 ConfigStructWindow, 218
 b10sarr
 ConfigStructShade, 208
 b10sint
 ConfigStructShade, 208
 bSFlower
 ConfigStructShade, 208
 backwardSweep
 LVN_Node, 274
 BatteryDeltaT
 Appliance_Battery, 91
 batteryNeighbourhoodCharge
 Appliance_Battery, 91
 ConfigStructAppliance, 196
 batteryNeighbourhoodDischarge
 Appliance_Battery, 91
 ConfigStructAppliance, 196
 bedroom
 ConfigStructAgent, 190
 beforeClear
 Appliance, 56
 bopout
 ConfigStructWindow, 218
 boutfulllower
 ConfigStructShade, 208
 boutfullraise
 ConfigStructShade, 208
 bsfulllower
 ConfigStructShade, 209
 bsfullraise
 ConfigStructShade, 209
 Building, 158
 addContactsTo, 159
 Building, 159
 decisionBoolean, 160
 decisionDoubleVec, 161
 getID, 161
 getPower, 162
 hasZone, 162
 postTimeStep, 163
 postprocess, 163
 preprocess, 163
 setup, 164
 step, 164
 stepAppliancesNegotiation, 165
 stepAppliancesNegotiationNeighbourhood, 165
 stepAppliancesUse, 166
 stepAppliancesUseBatteries, 166
 Building_Appliances, 167
 addContactsTo, 168
 addCurrentStates, 169

Building_Appliances, 168
getTotalPower, 169
postTimeStep, 170
postprocess, 170
preprocess, 171
setup, 171
stepAppliancesUseBatteries, 172
stepGlobalNegotiation, 173
stepLocal, 173
stepLocalNegotiation, 174
stepNeighbourhoodNegotiation, 175
Building_Zone, 176
Building_Zone, 177
getActivities, 177
getAirRelativeHumidity, 178
getAirSystemSensibleHeatingRate, 178
getBlindState, 179
getCurrentOccupantCount, 179
getCurrentOccupantGains, 179
getDaylightingReferencePoint1Illuminance, 179
getGroundFloor, 180
getHeatingState, 180
getId, 181
getLightState, 182
getMeanAirTemperature, 183
getMeanRadiantTemperature, 183
getNumberOfActivities, 184
getOccupantFraction, 184
getWindowDurationOpen, 184
getWindowState, 184
hasActivity, 184
isActive, 185
isNamed, 185
setActive, 185
setAppFraction, 185
setBlindState, 186
setCurrentOccupantGains, 186
setGroundFloor, 186
setHeatingState, 186
setIDString, 187
setLightState, 187
setName, 187
setOccupantFraction, 187
setWindowDurationOpen, 188
setWindowState, 188
setup, 187
step, 188
buildingID
 Agent, 50
 Contract, 229
buildings
 Configuration, 226
calculate
 Model_HeatGains, 334
calculateDailyMeanTemperature
 Environment, 258
calculateDeltaE
 Appliance_Battery, 80
calculateHourOfDay
 Appliance, 56
calculateNumberOfDays
 Utility, 486
calculatePresenceFromPage
 Model_Presence, 341
calculateProfile
 Appliance_Large_Learning, 143
 Appliance_Large_Learning_CSV, 150
calculateStateAtTenMin
 Model_Appliance_Small_Usage, 321
calculateStateOfCharge
 Appliance_Battery, 80
calculateSupply
 Appliance_Battery, 81
capacity
 Appliance_Battery, 92
caseOrder
 ConfigStructSimulation, 210
chargeRate
 Appliance_Battery, 92
checkTolerance
 LVN_Node, 275
children
 ConfigStructLVNNNode, 204
civstat
 ConfigStructAgent, 191
 Model_Activity, 292
clear
 Appliance, 56
 Appliance_Battery, 82
 Appliance_Group, 111
 Contract_Negotiation, 232
 Contract_Node_Tree, 246
 DataStore, 254
clearValues
 DataStore, 254
clo
 State, 480
compare
 Contract_Node_Priority, 239
 Contract_Node_Supply, 243
 Contract_Node_Tree, 246
computer
 ConfigStructAgent, 191
 Model_Activity, 293
ConfigStructAgent, 189
 age, 190
 ApplianceDuringDay, 190
 bedroom, 190
 civstat, 191
 computer, 191
 edtry, 191
 famstat, 191
 LightOffDuringAudioVisual, 191
 LightOffDuringSleep, 191
 name, 192
 office, 192

power, 192
 profile, 192
 retired, 192
 sex, 192
 ShadeClosedDuringSleep, 193
 ShadeClosedDuringWashing, 193
 ShadeDuringAudioVisual, 193
 ShadeDuringNight, 193
 shadeld, 193
 unemp, 193
 windowId, 194
 WindowOpenDuringCooking, 194
 WindowOpenDuringSleeping, 194
 WindowOpenDuringWashing, 194
ConfigStructAppliance, 195
 activities, 196
 alpha, 196
 batteryNeighbourhoodCharge, 196
 batteryNeighbourhoodDischarge, 196
 cost, 196
 costVector, 196
 epsilon, 197
 fileDemand, 197
 fileProfile, 197
 fileSupply, 197
 Fractions, 197
 gamma, 197
 id, 198
 name, 198
 priority, 198
 StateProbabilities, 198
 SumRatedPowers, 198
 timeRequired, 198
 update, 199
 variableName, 199
 WeibullParameters, 199
ConfigStructBuilding, 200
 agents, 201
 AppliancesBattery, 201
 AppliancesBatteryGrid, 201
 AppliancesCSV, 201
 AppliancesFMI, 201
 AppliancesGrid, 201
 AppliancesLarge, 202
 AppliancesLargeCSV, 202
 AppliancesLargeLearning, 202
 AppliancesLargeLearningCSV, 202
 AppliancesSmall, 202
 id, 202
 name, 203
 zones, 203
ConfigStructLVNNode, 203
 children, 204
 id, 204
 impedance, 204
 parent, 204
ConfigStructShade, 205
 a01arr, 206
 a01int, 206
 a10arr, 206
 a10int, 206
 aSFlower, 207
 afulllower, 206
 afullraise, 206
 b01inarr, 207
 b01inint, 207
 b01sarr, 207
 b01sint, 207
 b10inarr, 207
 b10inint, 208
 b10sarr, 208
 b10sint, 208
 bSFlower, 208
 boutfulllower, 208
 boutfullraise, 208
 bsfulllower, 209
 bsfullraise, 209
 shapelower, 209
ConfigStructSimulation, 209
 agentHeatGains, 210
 caseOrder, 210
 endDay, 210
 endMonth, 211
 GridCost, 211
 heating, 211
 learn, 211
 learnep, 211
 learnupdate, 211
 lights, 212
 precision, 212
 presencePage, 212
 save, 212
 ShadeClosedDuringNight, 212
 shading, 212
 startDay, 213
 startDayOfWeek, 213
 startMonth, 213
 timeSteps, 213
 timeStepsPerHour, 213
 windows, 213
 windowsLearn, 214
ConfigStructWindow, 214
 a01arr, 215
 a01dep, 215
 a01int, 215
 a10dep, 216
 aop, 216
 b01absdep, 216
 b01absprevarr, 216
 b01gddep, 216
 b01inarr, 216
 b01inint, 217
 b01outarr, 217
 b01outdep, 217
 b01outint, 217
 b01presint, 217

b01rnarr, 217
b01rnint, 218
b10absdep, 218
b10gddep, 218
b10indep, 218
b10outdep, 218
bopout, 218
shapeop, 219
ConfigStructZone, 219
active, 220
activities, 220
groundFloor, 220
id, 220
name, 220
windowCount, 220
Configuration, 221
buildings, 226
FileActivity, 226
FileLargeAppliance, 226
FolderSmallAppliance, 226
getStepCount, 222
getZone, 223
info, 227
isZoneGroundFloor, 223
lengthOfTimestep, 224
lvn, 227
outputRegexs, 227
parseConfiguration, 224
reset, 225
RunLocation, 227
setStepCount, 225
shades, 227
step, 226
windows, 227
consumption
 Model_Appliance_Large_Usage, 301
 Model_Appliance_Small_Usage, 321
Contract, 228
 buildingID, 229
 id, 229
 priority, 229
 received, 229
 receivedCost, 229
 requested, 230
 supplied, 230
 suppliedCost, 230
 suppliedLeft, 230
Contract_Negotiation, 231
 clear, 232
 Contract_Negotiation, 231
 getContract, 232
 getCostOfPowerForContract, 233
 getDifference, 233
 getReceivedPowerForContract, 234
 process, 234
 submit, 235
Contract_Node_Priority, 236
 compare, 239
 Contract_Node_Priority, 239
 isNodeRemoveable, 239
 isRemoveable, 239
 makeLeft, 239
 makeRight, 240
 Contract_Node_Supply, 240
 compare, 243
 Contract_Node_Supply, 243
 isNodeRemoveable, 243
 isRemoveable, 243
 makeLeft, 243
 makeRight, 244
 Contract_Node_Tree
 clear, 246
 compare, 246
 Contract_Node_Tree, 246
 findLeftEdge, 247
 findRightEdge, 247
 getNodeObject, 247
 insert, 247
 isLeftNull, 248
 isNodeRemoveable, 248
 isRemoveable, 248
 isRightNull, 249
 makeLeft, 249
 makeRight, 249
 nodeObject, 250
 pLeft, 250
 pRight, 251
 popLeftEdge, 250
Contract_Node_Tree< T >, 244
cooker
 Model_Appliance_Ownership, 315
cost
 Appliance_Battery, 92
 ConfigStructAppliance, 196
 profileStruct, 448
costVector
 ConfigStructAppliance, 196
count
 Appliance_Large_CSV, 139
country
 Model_Appliance_Large_Usage, 306
csv.ToDouble
 Utility, 486
csvToInt
 Utility, 487
csv.ToTable
 Utility, 488
csv.ToTableHead
 Utility, 488
cumulativeProbability
 Utility, 489, 490
currentlyInZone
 Occupant, 364
dailyMeanTemperature
 Environment, 261
 Occupant_Action_Window, 421

DataStore, 251
 addValue, 252
 addValueS, 253
 addVariable, 253
 clear, 254
 clearValues, 254
 getID, 255
 getValue, 255
 getValueForZone, 256
 getValueS, 256
 print, 257
 databaseIdDay
 SimulationTime, 472
 databaseIdHour
 SimulationTime, 472
 databaseIdHourOfDay
 SimulationTime, 472
 databaseIdMinute
 SimulationTime, 472
 databaseIdMinuteOfDay
 SimulationTime, 472
 databaseIdMonth
 SimulationTime, 472
 databaseIdStepCount
 SimulationTime, 473
 datastoreGridIDCost
 Appliance, 72
 datastoreGridIDReceived
 Appliance, 72
 datastoreGridIDRequested
 Appliance, 72
 datastoreGridIDSupplied
 Appliance, 72
 datastoreGridIDSuppliedCost
 Appliance, 72
 datastoreIDCost
 Appliance, 72
 Appliance_Group, 121
 datastoreIDReceived
 Appliance, 73
 Appliance_Group, 121
 datastoreIDRequested
 Appliance, 73
 Appliance_Group, 121
 datastoreIDSupplied
 Appliance, 73
 Appliance_Group, 122
 datastoreIDSuppliedCost
 Appliance, 73
 Appliance_Group, 122
 datastoreIDstateOfCharge
 Appliance_Battery, 92
 datastoreLocalIDCost
 Appliance, 73
 datastoreLocalIDReceived
 Appliance, 73
 datastoreLocalIDRequested
 Appliance, 74
 datastoreLocalIDSupplied
 Appliance, 74
 datastoreLocalIDSuppliedCost
 Appliance, 74
 datastoreNeighbourhoodIDCost
 Appliance, 74
 datastoreNeighbourhoodIDReceived
 Appliance, 74
 datastoreNeighbourhoodIDRequested
 Appliance, 74
 datastoreNeighbourhoodIDSupplied
 Appliance, 75
 datastoreNeighbourhoodIDSuppliedCost
 Appliance, 75
 day
 SimulationTime, 473
 decisionBoolean
 Building, 160
 decisionDoubleVec
 Building, 161
 decreaseDuration
 Model_Appliance_Large_Usage_Survival, 311
 departure
 Model_ExternalShading, 329
 Model_Lights, 338
 Model_Windows, 350
 dictionary
 Model_Activity, 293
 disaggregate
 Model_Activity, 284
 dischargeRate
 Appliance_Battery, 92
 dishwasher
 Model_Appliance_Ownership, 315
 doAction
 Appliance_Battery, 82
 doRecipe
 Occupant_Action_Appliance_BDI, 386
 Occupant_Action_Lights_BDI, 405
 Occupant_Action_Shades_BDI, 414
 Occupant_Action_Window_Stochastic_BDI, 434
 durationAtState
 Model_Appliance_Small_Usage, 322
 durationOpen
 Occupant_Action_Window, 419
 edtry
 ConfigStructAgent, 191
 Model_Activity, 293
 efficiency
 Appliance_Battery, 92
 enabled
 Occupant_Action_Window_Stochastic_BDI, 435
 endDay
 ConfigStructSimulation, 210
 endMonth
 ConfigStructSimulation, 211
 energy_calc
 Appliance_Battery, 83

engine
 Utility, 493

Environment, 258
 calculateDailyMeanTemperature, 258
 dailyMeanTemperature, 261
 getDailyMeanTemperature, 259
 getEVG, 259
 getOutdoorAirDrybulbTemperature, 260

epsilon
 ConfigStructAppliance, 197

error
 Log, 266

eventInfo
 ModellInstance, 359

famstat
 ConfigStructAgent, 191
 Model_Activity, 293

file
 Appliance_Large, 134

FileActivity
 Configuration, 226

fileDemand
 ConfigStructAppliance, 197

FileLargeAppliance
 Configuration, 226

fileProfile
 ConfigStructAppliance, 197

fileSupply
 ConfigStructAppliance, 197

filename
 QLearning, 461

findLeftEdge
 Contract_Node_Tree, 247

findRightEdge
 Contract_Node_Tree, 247

fmiCallbackFunctions, 261
 allocateMemory, 262
 freeMemory, 262
 logger, 262
 stepFinished, 262

fmiEventInfo, 263
 iterationConverged, 263
 nextEventTime, 263
 stateValueReferencesChanged, 264
 stateValuesChanged, 264
 terminateSimulation, 264
 upcomingTimeEvent, 264

FolderSmallAppliance
 Configuration, 226

forwardSweep
 LVN_Node, 275

Fractions
 ConfigStructAppliance, 197

freeMemory
 fmiCallbackFunctions, 262

fridge
 Model_Appliance_Ownership, 315

functions
 ModellInstance, 359

GUID
 ModellInstance, 359

gamma
 ConfigStructAppliance, 197

get_charge_delta
 Appliance_Battery, 84

get_new_SOC_charge
 Appliance_Battery, 84

get_new_SOC_discharge
 Appliance_Battery, 85

getA01arr
 Model_Windows, 351

getA01dep
 Model_Windows, 351

getA01int
 Model_Windows, 351

getA10dep
 Model_Windows, 351

getAction
 QLearning, 452

getActivities
 Building_Zone, 177

getAirRelativeHumidity
 Building_Zone, 178

getAirSystemSensibleHeatingRate
 Building_Zone, 178

getAllHeatGains
 Model_HeatGains, 334

getAop
 Model_Windows, 351

getApplianceAt
 Appliance_Group, 111

getB01absdep
 Model_Windows, 351

getB01absprevarr
 Model_Windows, 352

getB01gddep
 Model_Windows, 352

getB01inarr
 Model_Windows, 352

getB01init
 Model_Windows, 352

getB01outarr
 Model_Windows, 352

getB01outdep
 Model_Windows, 352

getB01outint
 Model_Windows, 353

getB01presint
 Model_Windows, 353

getB01rnarr
 Model_Windows, 353

getB01rnint
 Model_Windows, 353

getB10absdep
 Model_Windows, 353

getB10gddep
 Model_Windows, 353

Model_Windows, 353
 getB10indep
 Model_Windows, 354
 getB10outdep
 Model_Windows, 354
 getBlindState
 Building_Zone, 179
 getBopout
 Model_Windows, 354
 getBuildingID
 Agent, 48
 getClo
 State, 476
 getContract
 Contract_Negotiation, 232
 getConvectiveHeatGains
 Model_HeatGains, 335
 getCostOfPowerForContract
 Contract_Negotiation, 233
 getCountry
 Model_Appliance_Large_Usage, 302
 getCurrentDurationOfPresenceState
 Occupant_Action, 377
 getCurrentOccupantCount
 Building_Zone, 179
 getCurrentOccupantGains
 Building_Zone, 179
 getCurrentRadientGains
 Occupant, 365
 getDailyMeanTemperature
 Environment, 259
 getDay
 Model_Activity, 284
 getDaylightingReferencePoint1Illuminance
 Building_Zone, 179
 getDesiredAppliance
 Occupant, 366
 Occupant_Zone, 440
 getDesiredHeatState
 Occupant, 366
 getDesiredHeatingSetPoint
 Occupant_Zone, 440
 getDesiredLightState
 Occupant, 366
 Occupant_Zone, 440
 getDesiredShadeState
 Occupant, 367
 Occupant_Zone, 440
 getDesiredWindowDuration
 Occupant_Zone, 440
 getDesiredWindowState
 Occupant, 367
 Occupant_Zone, 441
 getDifference
 Contract_Negotiation, 233
 getDryRespiration
 Model_HeatGains, 335
 getDurationOpen
 Model_Windows, 354
 getEVG
 Environment, 259
 getError
 Log, 266
 getFractionalPowerAtState
 Model_Appliance_Small_Usage, 323
 getFutureDurationOfAbsenceState
 Occupant_Action, 378
 getGridCost
 Simulation, 464
 getGridPower
 Appliance, 57
 getGridReceived
 Appliance, 57
 getGridReceivedCost
 Appliance, 57
 getGridSupply
 Appliance, 57
 getGridSupplyLeft
 Appliance, 57
 getGroundFloor
 Building_Zone, 180
 getHeatgains
 Occupant_Zone, 441
 getHeatingState
 Building_Zone, 180
 getID
 Agent, 48
 Building, 161
 DataStore, 255
 LVN_Node, 276
 getId
 Building_Zone, 181
 Occupant_Zone, 441
 State, 477
 getLatentRespirationHeatGains
 Model_HeatGains, 335
 getLightState
 Building_Zone, 182
 getLocalPower
 Appliance, 58
 getLocalReceived
 Appliance, 58
 getLocalReceivedCost
 Appliance, 58
 getLocalSupply
 Appliance, 58
 getLocalSupplyLeft
 Appliance, 58
 getMeanAirTemperature
 Building_Zone, 183
 getMeanFraction
 Model_Appliance_Large_Usage, 302
 getMeanRadiantTemperature
 Building_Zone, 183
 getMetabolicRate
 State, 477

getNeighbourhoodPower
 Appliance, 58
getNeighbourhoodReceived
 Appliance, 59
getNeighbourhoodReceivedCost
 Appliance, 59
getNeighbourhoodSupply
 Appliance, 59
getNeighbourhoodSupplyLeft
 Appliance, 59
getNodeObject
 Contract_Node_Tree, 247
getNumberOfActivities
 Building_Zone, 184
getOccupantFraction
 Building_Zone, 184
getOutdoorAirDrybulbTemperature
 Environment, 260
getPMV
 Occupant_Action_Heat_Gains, 390
 Occupant_Zone, 441
getPPD
 Occupant_Action_Heat_Gains, 390
getPmv
 Model_HeatGains, 335
getPower
 Appliance, 59
 Appliance_Group, 111
 Building, 162
 Model_Appliance_Large_Usage, 302
 Occupant, 368
getPowerAt
 Appliance_Large_Learning, 143
getPowerShortage
 Appliance_Group_Battery, 125
getPpd
 Model_HeatGains, 336
getPreviousDurationOfAbsenceState
 Occupant_Action, 378
getPriority
 Appliance, 60
getRadianHeatGains
 Model_HeatGains, 336
getReceived
 Appliance, 60
 Appliance_Group, 112
getReceivedCost
 Appliance, 61
 Appliance_Group, 112
getReceivedPowerForContract
 Contract_Negotiation, 234
getRequiredTime
 Appliance_Large_Learning, 144
getResult
 Occupant_Action, 379
getSeasonInt
 Model_Activity, 285
getSeasonString

 Model_Activity, 285
getState
 State, 478
getStateID
 Occupant, 368
getStateOfCharge
 Appliance_Battery, 86
getStateProbabilities
 Model_Appliance_Small_Usage, 323
getStepCount
 Configuration, 222
getSupply
 Appliance, 61
 Appliance_Group, 113
getSupplyCost
 Appliance, 62
 Appliance_Group, 113
getSupplyLeft
 Appliance, 62
 Appliance_Group, 113
getSweatEvaporation
 Model_HeatGains, 336
getTotalPower
 Building_Applications, 169
getValue
 DataStore, 255
getValueForZone
 DataStore, 256
getValueS
 DataStore, 256
getWindowDurationOpen
 Building_Zone, 184
getWindowState
 Building_Zone, 184
 Model_Windows, 355
getZone
 Configuration, 223
getZonePtr
 State, 478
getshapeop
 Model_Windows, 354
global
 Appliance, 75
globalContracts
 Appliance_Group, 122
globalCost
 Appliance_Group_Battery, 125
globalNegotiation
 Appliance_Group, 114
greedySelection
 QLearning, 452
GridCost
 ConfigStructSimulation, 211
groundFloor
 ConfigStructZone, 220
hasActivities
 Appliance, 62
 Appliance_Group, 114

hasActivity
 Building_Zone, 184
 hasState
 State, 478
 StateMachine, 483
 hasZone
 Building, 162
 heating
 ConfigStructSimulation, 211
 hour
 SimulationTime, 473
 hourOfDay
 SimulationTime, 473
 hourlyCost
 Appliance, 75
 hourlyPriority
 Appliance, 75
 i
 ModellInstance, 360
 id
 Agent, 50
 ConfigStructAppliance, 198
 ConfigStructBuilding, 202
 ConfigStructLVNNode, 204
 ConfigStructZone, 220
 Contract, 229
 Model_Appliance_Large_Usage, 306
 QLearning, 461
 State, 480
 idString
 Agent, 50
 Appliance_Group, 122
 impedance
 ConfigStructLVNNode, 204
 info
 Configuration, 227
 insert
 Contract_Node_Tree, 247
 instanceName
 ModellInstance, 360
 InteractionOnZone
 Occupant, 368
 intermediate
 Model_ExternalShading, 330
 Model_Lights, 339
 Model_Windows, 355
 isActionAppliance
 Occupant, 369
 Occupant_Zone, 441
 isActionHeatGains
 Occupant, 369
 Occupant_Zone, 442
 isActionLearning
 Occupant, 370
 Occupant_Zone, 442
 isActionLights
 Occupant, 370
 Occupant_Zone, 442
 isActionShades
 Occupant, 370
 Occupant_Zone, 442
 isActionWindow
 Occupant, 371
 Occupant_Zone, 443
 isActive
 Building_Zone, 185
 isGlobal
 Appliance, 62
 isInActivity
 State, 478
 isLearningPeriod
 profileStruct, 448
 isLeftNull
 Contract_Node_Tree, 248
 isLocal
 Appliance, 63
 isModelOn
 Appliance_Large_Learning, 144
 isNamed
 Building_Zone, 185
 isNodeRemoveable
 Contract_Node_Priority, 239
 Contract_Node_Supply, 243
 Contract_Node_Tree, 248
 isOn
 Appliance_Large, 131
 Model_Appliance_Large_Usage, 303
 isPositive
 ModellInstance, 360
 isRemoveable
 Contract_Node_Priority, 239
 Contract_Node_Supply, 243
 Contract_Node_Tree, 248
 isRightNull
 Contract_Node_Tree, 249
 isZoneGroundFloor
 Configuration, 223
 iterationConverged
 fmiEventInfo, 263
 LVN_Node, 272
 addChild, 273
 addNode, 274
 backwardSweep, 274
 checkTolerance, 275
 forwardSweep, 275
 getID, 276
 LVN_Node, 273
 resetIterations, 276
 runUntilConvergence, 276
 save, 277
 setID, 278
 setImpedance, 278
 setNodeLoad, 279
 setNominalVoltage, 279
 setPowerForID, 279
 setup, 280

LVN, 269
LVN, 269
postTimeStep, 270
setPowerForID, 270
setup, 271
learn
 ConfigStructSimulation, 211
 QLearning, 453
learnNext
 QLearning, 462
learnp
 ConfigStructSimulation, 211
learningStep
 profileStruct, 449
learnupdate
 ConfigStructSimulation, 211
lengthOfTimestep
 Configuration, 224
LightOffDuringAudioVisual
 ConfigStructAgent, 191
LightOffDuringSleep
 ConfigStructAgent, 191
lights
 ConfigStructSimulation, 212
local
 Appliance, 75
localNegotiation
 Appliance_Group, 114
Log, 265
 error, 266
 getError, 266
 Log, 265
 operator<<, 267
 printLog, 267
 reset, 268
logger
 fmiCallbackFunctions, 262
loggingOn
 ModellInstance, 360
Lumint
 Occupant_Action_Shades, 411
lvn
 Configuration, 227
m_window
 Occupant_Action_Window, 421
makeLeft
 Contract_Node_Priority, 239
 Contract_Node_Supply, 243
 Contract_Node_Tree, 249
makeRight
 Contract_Node_Priority, 240
 Contract_Node_Supply, 244
 Contract_Node_Tree, 249
match
 Appliance, 76
maxPower
 Model_Appliance_Large_Usage, 306
maxTimeRequired
 profileStruct, 449
meanF
 Model_Appliance_Large_Usage, 307
metabolicRate
 State, 480
microware
 Model_Appliance_Ownership, 316
minute
 SimulationTime, 473
minuteOfDay
 SimulationTime, 473
model
 Appliance_Large, 134
Model_Activity, 281
 age, 292
 civstat, 292
 computer, 293
 dictionary, 293
 disaggregate, 284
 edtry, 293
 famstat, 293
 getDay, 284
 getSeasonInt, 285
 getSeasonString, 285
 Model_Activity, 283
 multinomial, 285
 multinomialActivity, 286
 multinomialP, 286
 parseConfiguration, 287
 parseOther, 288
 preProcessActivities, 288
 probMap, 293
 retired, 293
 setAge, 289
 setCivstat, 289
 setComputer, 289
 setEdtry, 290
 setFamstat, 290
 setProbMap, 291
 setRetired, 291
 setSex, 291
 setUnemp, 292
 sex, 294
 unemp, 294
Model_Activity_Survival, 294
 Model_Activity_Survival, 297
 multinomialActivity, 297
Model_Appliance_Large_Usage, 298
 as_vector, 301
 as_vector_vector, 301
 consumption, 301
 country, 306
 getCountry, 302
 getMeanFraction, 302
 getPower, 302
 id, 306
 isOn, 303
 maxPower, 306

meanF, 307
 Model_Appliance_Large_Usage, 300
 name, 307
 on, 307
 onAt, 303
 onProbabilities, 307
 onProbabilities10, 307
 parseConfiguration, 304
 parseShapeScale, 304
 probOn, 305
 setCountry, 305
 setId, 306
 state, 307
 transitions, 308
 Model_Appliance_Large_Usage_Survival, 308
 decreaseDuration, 311
 Model_Appliance_Large_Usage_Survival, 311
 onAt, 312
 setDuration, 313
 setScale, 313
 setShape, 313
 Model_Appliance_Ownership, 314
 cooker, 315
 dishwasher, 315
 fridge, 315
 microware, 316
 Model_Appliance_Ownership, 315
 tvless21, 316
 tvmore21, 316
 washingMachine, 316
 Model_Appliance_Power_CSV, 317
 Model_Appliance_Power_CSV, 318
 parseConfiguration, 318
 power, 318
 Model_Appliance_Small_Usage, 319
 calculateStateAtTenMin, 321
 consumption, 321
 durationAtState, 322
 getFractionalPowerAtState, 323
 getStateProbabilities, 323
 Model_Appliance_Small_Usage, 323
 readFractions, 323
 readStateProbabilities, 323
 readSumRatedPowers, 324
 readWeibullParameters, 324
 setFolderLocation, 325
 setRatedPowerAt, 325
 weibullInvCdf, 325
 Model_ExternalShading, 326
 arrival, 329
 departure, 329
 intermediate, 330
 Model_ExternalShading, 329
 setArrivalVars, 330
 setDurationVars, 331
 setFullVars, 331
 setInterVars, 332
 Model_HeatGains, 333
 calculate, 334
 getAllHeatGains, 334
 getConvectiveHeatGains, 335
 getDryRespiration, 335
 getLatentRespirationHeatGains, 335
 getPmv, 335
 getPpd, 336
 getRadiantHeatGains, 336
 getSweatEvaporation, 336
 Model_HeatGains, 334
 Model_Lights, 337
 arrival, 338
 departure, 338
 intermediate, 339
 Model_Lights, 337
 Model_Presence, 340
 at, 341
 calculatePresenceFromPage, 341
 Model_Presence, 340
 presentForFutureSteps, 342
 setProbMap, 342
 size, 342
 Model_RandomWeibull, 343
 Model_RandomWeibull, 344
 probability, 345
 randomDouble, 345
 randomWeibull, 346
 Model_Windows, 347
 arrival, 349
 departure, 350
 getA01arr, 351
 getA01dep, 351
 getA01int, 351
 getA10dep, 351
 getAop, 351
 getB01absdep, 351
 getB01absprevarr, 352
 getB01gddep, 352
 getB01inarr, 352
 getB01int, 352
 getB01outarr, 352
 getB01outdep, 352
 getB01outint, 353
 getB01presint, 353
 getB01rnarr, 353
 getB01rnint, 353
 getB10absdep, 353
 getB10gddep, 353
 getB10indep, 354
 getB10outdep, 354
 getBopout, 354
 getDurationOpen, 354
 getWindowState, 355
 getshapeop, 354
 intermediate, 355
 Model_Windows, 349
 setArrivalVars, 355
 setDepartureVars, 356

setDurationOpen, 356
setDurationVars, 357
setInterVars, 357
setWindowState, 357
ModellInstance, 358
 b, 359
 eventInfo, 359
 functions, 359
 GUID, 359
 i, 360
 instanceName, 360
 isPositive, 360
 loggingOn, 360
 r, 360
 s, 360
 sim, 361
 state, 361
 time, 361
month
 SimulationTime, 474
mostShortage
 Appliance_Battery, 93
multinomial
 Model_Activity, 285
multinomialActivity
 Model_Activity, 286
 Model_Activity_Survival, 297
multinomialP
 Model_Activity, 286
name
 ConfigStructAgent, 192
 ConfigStructAppliance, 198
 ConfigStructBuilding, 203
 ConfigStructZone, 220
 Model_Appliance_Large_Usage, 307
negotiationApp
 Appliance_Group, 115
 Appliance_Group_Battery, 126
negotiationAppGlobal
 Appliance_Group, 115
 Appliance_Group_Battery, 126
neighbourhoodNegotiation
 Appliance_Group, 116
neighbourhoodNegotiationBattery
 Appliance_Group_Battery, 126
nextEventTime
 fmiEventInfo, 263
nodeObject
 Contract_Node_Tree, 250
nonLearningStep
 profileStruct, 449
numberOfStates
 StateMachine, 483
numberOfSubStates
 State, 479
Occupant, 362
 currentlyInZone, 364
getCurrentRadientGains, 365
getDesiredAppliance, 366
getDesiredHeatState, 366
getDesiredLightState, 366
getDesiredShadeState, 367
getDesiredWindowState, 367
getPower, 368
getStateID, 368
InteractionOnZone, 368
isActionAppliance, 369
isActionHeatGains, 369
isActionLearning, 370
isActionLights, 370
isActionShades, 370
isActionWindow, 371
Occupant, 364
postTimeStep, 371
postprocess, 371
previouslyInZone, 372
setBuildingName, 372
setState, 372
setup, 373
step, 374
zoneInteractions, 375
Occupant_Action, 375
 activityAvailable, 377
 availableActivities, 381
 getCurrentDurationOfPresenceState, 377
 getFutureDurationOfAbsenceState, 378
 getPreviousDurationOfAbsenceState, 378
 getResult, 379
 Occupant_Action, 377
 result, 381
 reward, 381
 setAvailableActivities, 379
 setReward, 380
 setZoneld, 380
 zoneld, 381
Occupant_Action_Appliance, 382
 Occupant_Action_Appliance, 383
Occupant_Action_Appliance_BDI, 384
 doRecipe, 386
 Occupant_Action_Appliance_BDI, 386
 setApplianceDuringDay, 386
Occupant_Action_Heat_Gains, 387
 getPMV, 390
 getPPD, 390
 Occupant_Action_Heat_Gains, 390
 prestep, 390
 setup, 391
 step, 391
Occupant_Action_HeatingSetPoints_Learning, 393
 Occupant_Action_HeatingSetPoints_Learning, 396
 print, 396
 reset, 396
 setFile, 397
 setup, 397
 step, 398

Occupant_Action_Lights, 399
 Occupant_Action_Lights, 401
 step, 402
 Occupant_Action_Lights_BDI, 403
 doRecipe, 405
 Occupant_Action_Lights_BDI, 405
 setOffDuringAudioVisual, 405
 setOffDuringSleep, 406
 Occupant_Action_Shades, 407
 Lumint, 411
 Occupant_Action_Shades, 409
 setIndoorIlluminance, 409
 setup, 409
 step, 410
 Occupant_Action_Shades_BDI, 411
 doRecipe, 414
 Occupant_Action_Shades_BDI, 414
 setClosedDuringAudioVisual, 415
 setClosedDuringNight, 415
 setClosedDuringSleep, 415
 setClosedDuringWashing, 416
 Occupant_Action_Window, 417
 dailyMeanTemperature, 421
 durationOpen, 419
 m_window, 421
 Occupant_Action_Window, 419
 saveResult, 419
 setDailyMeanTemperature, 420
 setup, 420
 variableNameWindowDesire, 421
 Occupant_Action_Window_Learning, 422
 Occupant_Action_Window_Learning, 424
 print, 424
 reset, 424
 setup, 425
 step, 426
 Occupant_Action_Window_Stochastic, 427
 Occupant_Action_Window_Stochastic, 429
 setup, 429
 step, 430
 Occupant_Action_Window_Stochastic_BDI, 431
 doRecipe, 434
 enabled, 435
 Occupant_Action_Window_Stochastic_BDI, 434
 setDailyMeanTemperature, 435
 setOpenDuringCooking, 435
 setOpenDuringSleeping, 436
 setOpenDuringWashing, 436
 Occupant_Zone, 437
 actionStep, 438
 getDesiredAppliance, 440
 getDesiredHeatingSetPoint, 440
 getDesiredLightState, 440
 getDesiredShadeState, 440
 getDesiredWindowState, 440
 getHeatgains, 441
 getId, 441
 getPMV, 441
 isActionAppliance, 441
 isActionHeatGains, 442
 isActionLearning, 442
 isActionLights, 442
 isActionShades, 442
 isActionWindow, 443
 Occupant_Zone, 438
 postTimeStep, 443
 postprocess, 443
 setClo, 444
 setMetabolicRate, 444
 setup, 444
 step, 445
 stepPre, 446
 office
 ConfigStructAgent, 192
 on
 Model_Appliance_Large_Usage, 307
 onAt
 Model_Appliance_Large_Usage, 303
 Model_Appliance_Large_Usage_Survival, 312
 onProbabilities
 Model_Appliance_Large_Usage, 307
 onProbabilities10
 Model_Appliance_Large_Usage, 307
 operator<<
 Log, 267
 outputRegexs
 Configuration, 227
 pLeft
 Contract_Node_Tree, 250
 pRight
 Contract_Node_Tree, 251
 parameters
 Appliance, 76
 parametersGrid
 Appliance, 76
 parametersLocal
 Appliance, 76
 parametersNeighbourhood
 Appliance, 76
 parent
 ConfigStructLVNNode, 204
 parseConfiguration
 Configuration, 224
 Model_Activity, 287
 Model_Appliance_Large_Usage, 304
 Model_Appliance_Power_CSV, 318
 Simulation, 464
 parseOther
 Model_Activity, 288
 parseShapeScale
 Model_Appliance_Large_Usage, 304
 popLeftEdge
 Contract_Node_Tree, 250
 postTimeStep
 Agent, 48

Building, 163
Building_Appliances, 170
LVN, 270
Occupant, 371
Occupant_Zone, 443
Simulation, 465
postprocess
 Agent, 48
 Appliance_Battery, 86
 Appliance_Group, 116
 Appliance_Large_Learning, 145
 Building, 163
 Building_Appliances, 170
 Occupant, 371
 Occupant_Zone, 443
 Simulation, 465
power
 ApplianceParameters, 156
 ConfigStructAgent, 192
 Model_Appliance_Power_CSV, 318
 profileStruct, 449
powerAvailable
 Appliance_Battery, 87
powerProfile
 Appliance_Large_Learning, 147
powerShortage
 Appliance_Battery, 93
preProcessActivities
 Model_Activity, 288
preTimeStep
 Simulation, 466
precision
 ConfigStructSimulation, 212
preprocess
 Agent, 48
 Building, 163
 Building_Appliances, 171
 Simulation, 466
 SimulationTime, 470
presencePage
 ConfigStructSimulation, 212
presentForFutureSteps
 Model_Presence, 342
prestep
 Occupant_Action_Heat_Gains, 390
previous_reward
 QLearning, 462
previous_state
 QLearning, 462
previousHourOfDay
 Appliance_Battery, 93
previouslyInZone
 Occupant, 372
print
 DataStore, 257
 Occupant_Action_HeatingSetPoints_Learning, 396
 Occupant_Action_Window_Learning, 424
printLog
 Log, 267
 printQ
 QLearning, 453
 priority
 ConfigStructAppliance, 198
 Contract, 229
 probMap
 Model_Activity, 293
 probOn
 Model_Appliance_Large_Usage, 305
probability
 Model_RandomWeibull, 345
process
 Contract_Negotiation, 234
profile
 ConfigStructAgent, 192
profileCSV
 Appliance_Large, 135
profileStruct, 447
 cost, 448
 isLearningPeriod, 448
 learningStep, 449
 maxTimeRequired, 449
 nonLearningStep, 449
 power, 449
 requestedTime, 449
 startTime, 449
QLearning, 450
 action, 461
 actions, 461
 filename, 461
 getAction, 452
 greedySelection, 452
 id, 461
 learn, 453
 learnNext, 462
 previous_reward, 462
 previous_state, 462
 printQ, 453
 QLearning, 451
 reset, 454
 reward, 462
 setAction, 454
 setActions, 455
 setAlpha, 455
 setEpsilon, 456
 setFilename, 456
 setGamma, 457
 setId, 457
 setReward, 458
 setState, 458
 setStates, 459
 setUpdate, 460
 setup, 459
 state, 462
 states, 462
 updateQ, 460
qLearning

Appliance_Battery, 93
 r
 ModellInstance, 360
 randomDouble
 Model_RandomWeibull, 345
 Utility, 490
 randomInt
 Utility, 491
 randomIntVect
 Utility, 491
 randomWeibull
 Model_RandomWeibull, 346
 readFractions
 Model_Appliance_Small_Usage, 323
 readStateProbabilities
 Model_Appliance_Small_Usage, 323
 readSumRatedPowers
 Model_Appliance_Small_Usage, 324
 readWeibullParameters
 Model_Appliance_Small_Usage, 324
 received
 ApplianceParameters, 157
 Contract, 229
 receivedCost
 ApplianceParameters, 157
 Contract, 229
 requested
 Contract, 230
 requestedTime
 profileStruct, 449
 reset
 Configuration, 225
 Log, 268
 Occupant_Action_HeatingSetPoints_Learning, 396
 Occupant_Action_Window_Learning, 424
 QLearning, 454
 SimulationTime, 470
 resetIterations
 LVN_Node, 276
 result
 Occupant_Action, 381
 retired
 ConfigStructAgent, 192
 Model_Activity, 293
 reward
 Occupant_Action, 381
 QLearning, 462
 rewardFunction
 Appliance_Battery, 87
 Appliance_Battery_GridCost_Reward, 97
 RunLocation
 Configuration, 227
 runUntilConvergence
 LVN_Node, 276
 running
 Appliance_Large_CSV, 139
 s
 ModellInstance, 360
 save
 Appliance, 63
 ConfigStructSimulation, 212
 LVN_Node, 277
 saveGlobal
 Appliance, 63
 saveGlobalCalculate
 Appliance, 63
 Appliance_Battery, 87
 saveLocal
 Appliance, 64
 saveLocalCalculate
 Appliance, 64
 saveNeighbourhood
 Appliance, 65
 saveNeighbourhoodCalculate
 Appliance, 65
 Appliance_Battery, 88
 saveResult
 Occupant_Action_Window, 419
 sendCondition
 Appliance_Group, 117
 Appliance_Group_Battery, 127
 sendContractGlobal
 Appliance_Group, 117
 sendContractLocal
 Appliance_Group, 118
 setAction
 QLearning, 454
 setActions
 QLearning, 455
 setActive
 Building_Zone, 185
 setActivities
 Appliance, 65
 setActivity
 State, 479
 setAge
 Model_Activity, 289
 setAlpha
 QLearning, 455
 setAppFraction
 Building_Zone, 185
 setApplianceDuringDay
 Occupant_Action_Appliance_BDI, 386
 setArrivalVars
 Model_ExternalShading, 330
 Model_Windows, 355
 setAvailableActivities
 Occupant_Action, 379
 setBatteryNeighbourhoodCharge
 Appliance_Battery, 88
 setBatteryNeighbourhoodDischarge
 Appliance_Battery, 88
 setBlindState
 Building_Zone, 186
 setBuildingID

Agent, 49
setBuildingName
 Occupant, 372
setCivstat
 Model_Activity, 289
setClo
 Occupant_Zone, 444
 State, 479
setClosedDuringAudioVisual
 Occupant_Action_Shades_BDI, 415
setClosedDuringNight
 Occupant_Action_Shades_BDI, 415
setClosedDuringSleep
 Occupant_Action_Shades_BDI, 415
setClosedDuringWashing
 Occupant_Action_Shades_BDI, 416
setComputer
 Model_Activity, 289
setConfigurationuratonFile
 Simulation, 467
setCountry
 Model_Appliance_Large_Usage, 305
setCurrentOccupantGains
 Building_Zone, 186
setDailyMeanTemperature
 Occupant_Action_Window, 420
 Occupant_Action_Window_Stochastic_BDI, 435
setDepartureVars
 Model_Windows, 356
setDuration
 Model_Appliance_Large_Usage_Survival, 313
setDurationOpen
 Model_Windows, 356
setDurationVars
 Model_ExternalShading, 331
 Model_Windows, 357
setEdtry
 Model_Activity, 290
setEpsilon
 QLearning, 456
setFMIVariableName
 Appliance_FMI, 100
setFamstat
 Model_Activity, 290
setFile
 Appliance_Large, 132
 Occupant_Action_HeatingSetPoints_Learning, 397
setFileDemand
 Appliance_Generic_CSV, 104
setFileSupply
 Appliance_Generic_CSV, 105
setFilename
 QLearning, 456
setFolderLocation
 Model_Appliance_Small_Usage, 325
setFullVars
 Model_ExternalShading, 331
setGamma
 QLearning, 457
setGlobal
 Appliance, 66
setGroundFloor
 Building_Zone, 186
setHeatingState
 Building_Zone, 186
setHourlyPriority
 Appliance, 66
setHourlyTimeRequired
 Appliance_Large_Learning, 145
setHourlyCost
 Appliance, 67
setIdString
 Agent, 49
 Appliance_Group, 118
 Building_Zone, 187
setId
 Agent, 49
 LVN_Node, 278
 Model_Appliance_Large_Usage, 306
setId
 QLearning, 457
 State, 479
setImpedance
 LVN_Node, 278
setIndoorIlluminance
 Occupant_Action_Shades, 409
setInterVars
 Model_ExternalShading, 332
 Model_Windows, 357
setLightState
 Building_Zone, 187
setLocal
 Appliance, 67
setMetabolicRate
 Occupant_Zone, 444
 State, 479
setName
 Building_Zone, 187
setNodeLoad
 LVN_Node, 279
setNominalVoltage
 LVN_Node, 279
setOccupantFraction
 Building_Zone, 187
setOffDuringAudioVisual
 Occupant_Action_Lights_BDI, 405
setOffDuringSleep
 Occupant_Action_Lights_BDI, 406
setOpenDuringCooking
 Occupant_Action_Window_Stochastic_BDI, 435
setOpenDuringSleeping
 Occupant_Action_Window_Stochastic_BDI, 436
setOpenDuringWashing
 Occupant_Action_Window_Stochastic_BDI, 436
setPower
 Appliance, 68

setPowerForID
 LVN_Node, 279
 LVN, 270
setPowerShortage
 Appliance_Battery, 88
 Appliance_Group_Battery, 127
setProbMap
 Model_Activity, 291
 Model_Presence, 342
setRatedPowerAt
 Model_Appliance_Small_Usage, 325
setReceived
 Appliance, 68
setReceivedCost
 Appliance, 69
setRetired
 Model_Activity, 291
setReward
 Occupant_Action, 380
 QLearning, 458
setScale
 Model_Appliance_Large_Usage_Survival, 313
setSeed
 Utility, 491
setSex
 Model_Activity, 291
setShape
 Model_Appliance_Large_Usage_Survival, 313
setState
 Occupant, 372
 QLearning, 458
setStates
 QLearning, 459
setStepCount
 Configuration, 225
setSupply
 Appliance, 69
setSupplyCost
 Appliance, 70
setSupplyLeft
 Appliance, 70
setUnemp
 Model_Activity, 292
setUpdate
 QLearning, 460
setWindowDurationOpen
 Building_Zone, 188
setWindowState
 Building_Zone, 188
 Model_Windows, 357
setZoneId
 Occupant_Action, 380
setZonePtr
 State, 479
setup
 Agent, 50
 Appliance, 71
 Appliance_Battery, 89
 Appliance_FMI, 100
 Appliance_Generic_CSV, 105
 Appliance_Group, 119
 Appliance_Large, 132
 Appliance_Large_Learning, 146
 Appliance_Small, 154
 Building, 164
 Building_Applications, 171
 Building_Zone, 187
 LVN_Node, 280
 LVN, 271
 Occupant, 373
 Occupant_Action_Heat_Gains, 391
 Occupant_Action_HeatingSetPoints_Learning, 397
 Occupant_Action_Shades, 409
 Occupant_Action_Window, 420
 Occupant_Action_Window_Learning, 425
 Occupant_Action_Window_Stochastic, 429
 Occupant_Zone, 444
 QLearning, 459
setupModel
 Appliance_Battery, 89
 Appliance_Large, 133
 Appliance_Large_CSV, 138
 Appliance_Large_Learning_CSV, 150
setupSave
 Appliance, 71
 Appliance_Group, 119
setupSimulationModel
 Simulation, 467
sex
 ConfigStructAgent, 192
 Model_Activity, 294
ShadeClosedDuringNight
 ConfigStructSimulation, 212
ShadeClosedDuringSleep
 ConfigStructAgent, 193
ShadeClosedDuringWashing
 ConfigStructAgent, 193
ShadeDuringAudioVisual
 ConfigStructAgent, 193
ShadeDuringNight
 ConfigStructAgent, 193
shadeld
 ConfigStructAgent, 193
shades
 Configuration, 227
shading
 ConfigStructSimulation, 212
shapelower
 ConfigStructShade, 209
shapeop
 ConfigStructWindow, 219
shuffleAppliances
 Appliance_Group, 120
sim
 ModellInstance, 361
 Simulation, 463

getGridCost, 464
parseConfiguration, 464
postTimeStep, 465
postprocess, 465
preTimeStep, 466
preprocess, 466
setConfigurationurationFile, 467
setupSimulationModel, 467
Simulation, 464
timeStep, 467
SimulationTime, 468
databaseIdDay, 472
databaseIdHour, 472
databaseIdHourOfDay, 472
databaseIdMinute, 472
databaseIdMinuteOfDay, 472
databaseIdMonth, 472
databaseIdStepCount, 473
day, 473
hour, 473
hourOfDay, 473
minute, 473
minuteOfDay, 473
month, 474
preprocess, 470
reset, 470
stepCount, 474
trackTime, 471
size
 Model_Presence, 342
splitCSV
 Utility, 492
startDay
 ConfigStructSimulation, 213
startDayOfWeek
 ConfigStructSimulation, 213
startMonth
 ConfigStructSimulation, 213
startTime
 profileStruct, 449
State, 474
 ~State, 476
 activity, 480
 addState, 476
 clo, 480
 getClo, 476
 getId, 477
 getMetabolicRate, 477
 getState, 478
 getZonePtr, 478
 hasState, 478
 id, 480
 isInActivity, 478
 metabolicRate, 480
 numberOfSubStates, 479
 setActivity, 479
 setClo, 479
 setId, 479
 setMetabolicRate, 479
 setZonePtr, 479
 State, 475
 states, 481
 zone, 481
state
 Model_Appliance_Large_Usage, 307
 Modellnstance, 361
 QLearning, 462
StateMachine, 481
 addState, 482
 hasState, 483
 numberOfStates, 483
 StateMachine, 482
 transistionTo, 483
stateOfCharge
 Appliance_Battery, 93
StateProbabilities
 ConfigStructAppliance, 198
stateValueReferencesChanged
 fmiEventInfo, 264
stateValuesChanged
 fmiEventInfo, 264
states
 QLearning, 462
 State, 481
step
 Agent, 50
 Appliance_Battery, 90
 Appliance_FMI, 101
 Appliance_Generic_CSV, 106
 Appliance_Group, 120
 Appliance_Large, 134
 Appliance_Large_CSV, 138
 Appliance_Large_Learning, 147
 Appliance_Small, 155
 Building, 164
 Building_Zone, 188
 Configuration, 226
 Occupant, 374
 Occupant_Action_Heat_Gains, 391
 Occupant_Action_HeatingSetPoints_Learning, 398
 Occupant_Action_Lights, 402
 Occupant_Action_Shades, 410
 Occupant_Action_Window_Learning, 426
 Occupant_Action_Window_Stochastic, 430
 Occupant_Zone, 445
stepApp
 Appliance_Group, 120
 Appliance_Group_Battery, 127
stepAppliancesNegotiation
 Building, 165
stepAppliancesNegotiationNeighbourhood
 Building, 165
stepAppliancesUse
 Building, 166
stepAppliancesUseBatteries
 Building, 166

Building_Appliances, 172
 stepCount
 SimulationTime, 474
 stepFinished
 fmiCallbackFunctions, 262
 stepGlobalNegotiation
 Building_Appliances, 173
 stepLocal
 Building_Appliances, 173
 stepLocalNegotiation
 Building_Appliances, 174
 stepNeighbourhood
 Appliance_Battery, 90
 stepNeighbourhoodNegotiation
 Building_Appliances, 175
 stepPre
 Occupant_Zone, 446
 submit
 Contract_Negotiation, 235
 SumRatedPowers
 ConfigStructAppliance, 198
 sumShort
 Appliance_Battery, 93
 sumSupply
 Appliance_Battery, 94
 supplied
 Contract, 230
 suppliedCost
 Contract, 230
 suppliedLeft
 ApplianceParameters, 157
 Contract, 230
 supply
 ApplianceParameters, 157
 supplyCost
 ApplianceParameters, 157
 terminateSimulation
 fmiEventInfo, 264
 time
 ModellInstance, 361
 timeRequired
 ConfigStructAppliance, 198
 timeStep
 Simulation, 467
 timeSteps
 ConfigStructSimulation, 213
 timeStepsPerHour
 ConfigStructSimulation, 213
 tossACoin
 Utility, 492
 trackTime
 SimulationTime, 471
 transistionTo
 StateMachine, 483
 transitions
 Model_Appliance_Large_Usage, 308
 tvless21
 Model_Appliance_Ownership, 316
 tvmore21
 Model_Appliance_Ownership, 316
 uTable
 Utility, 486
 unemp
 ConfigStructAgent, 193
 Model_Activity, 294
 upcomingTimeEvent
 fmiEventInfo, 264
 update
 ConfigStructAppliance, 199
 updateQ
 QLearning, 460
 Utility, 484
 calculateNumberOfDays, 486
 csv.ToDouble, 486
 csv.ToInt, 487
 csv.ToTable, 488
 csv.ToTableHead, 488
 cumulativeProbability, 489, 490
 engine, 493
 randomDouble, 490
 randomInt, 491
 randomIntVect, 491
 setSeed, 491
 splitCSV, 492
 tossACoin, 492
 uTable, 486
 variableName
 ConfigStructAppliance, 199
 variableNameWindowDesire
 Occupant_Action_Window, 421
 washingMachine
 Model_Appliance_Ownership, 316
 weibullInvCdf
 Model_Appliance_Small_Usage, 325
 WeibullParameters
 ConfigStructAppliance, 199
 windowCount
 ConfigStructZone, 220
 windowId
 ConfigStructAgent, 194
 WindowOpenDuringCooking
 ConfigStructAgent, 194
 WindowOpenDuringSleeping
 ConfigStructAgent, 194
 WindowOpenDuringWashing
 ConfigStructAgent, 194
 windows
 ConfigStructSimulation, 213
 Configuration, 227
 windowsLearn
 ConfigStructSimulation, 214
 zone
 State, 481

zonelD
 Occupant_Action, [381](#)
zoneInteractions
 Occupant, [375](#)
zones
 ConfigStructBuilding, [203](#)