

ReGenESyS - Reborned GENeric and Expansible SYstem Simulator  
2019.0507a

Generated by Doxygen 1.8.11



# Contents

<b>1</b>	<b>GenESyS-Reborn</b>	<b>1</b>
<b>2</b>	<b>Hierarchical Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>7</b>
3.1	Class List . . . . .	7
<b>4</b>	<b>File Index</b>	<b>11</b>
4.1	File List . . . . .	11
<b>5</b>	<b>Class Documentation</b>	<b>15</b>
5.1	Assign Class Reference . . . . .	15
5.1.1	Member Enumeration Documentation . . . . .	16
5.1.1.1	DestinationType . . . . .	16
5.1.2	Constructor & Destructor Documentation . . . . .	17
5.1.2.1	Assign(Model *model) . . . . .	17
5.1.2.2	Assign(const Assign &orig) . . . . .	17
5.1.2.3	~Assign() . . . . .	17
5.1.3	Member Function Documentation . . . . .	17
5.1.3.1	_execute(Entity *entity) . . . . .	17
5.1.3.2	_loadInstance(std::list< std::string > words) . . . . .	18
5.1.3.3	_saveInstance() . . . . .	19
5.1.3.4	_verifySymbols(std::string *errorMessage) . . . . .	19
5.1.3.5	getAssignments() const . . . . .	19

5.1.3.6	<code>show()</code>	20
5.2	Assign::Assignment Class Reference	20
5.2.1	Constructor & Destructor Documentation	21
5.2.1.1	<code>Assignment(DestinationType destinationType, std::string destination, std::string expression)</code>	21
5.2.2	Member Function Documentation	21
5.2.2.1	<code>getDestination() const</code>	21
5.2.2.2	<code>getDestinationType() const</code>	21
5.2.2.3	<code>getExpression() const</code>	21
5.2.2.4	<code>setDestination(std::string _destination)</code>	22
5.2.2.5	<code>setDestinationType(DestinationType _destinationType)</code>	22
5.2.2.6	<code>setExpression(std::string _expression)</code>	22
5.3	Attribute Class Reference	22
5.3.1	Constructor & Destructor Documentation	23
5.3.1.1	<code>Attribute()</code>	23
5.3.1.2	<code>Attribute(std::string name)</code>	23
5.3.1.3	<code>Attribute(const Attribute &amp;orig)</code>	23
5.3.1.4	<code>~Attribute()</code>	23
5.3.2	Member Function Documentation	23
5.3.2.1	<code>_loadInstance(std::list&lt; std::string &gt; words)</code>	23
5.3.2.2	<code>_saveInstance()</code>	24
5.3.2.3	<code>_verifySymbols(std::string *errorMessage)</code>	24
5.3.2.4	<code>show()</code>	24
5.4	BuildSimulationModel Class Reference	25
5.4.1	Constructor & Destructor Documentation	25
5.4.1.1	<code>BuildSimulationModel()</code>	25
5.4.2	Member Function Documentation	25
5.4.2.1	<code>main(int argc, char **argv)</code>	25
5.5	Collector_if Class Reference	27
5.5.1	Detailed Description	27
5.5.2	Member Function Documentation	27

5.5.2.1	<a href="#">addValue(double value)=0</a>	27
5.5.2.2	<a href="#">clear()=0</a>	28
5.5.2.3	<a href="#">getLastValue()=0</a>	28
5.5.2.4	<a href="#">numElements()=0</a>	28
5.5.2.5	<a href="#">setAddValueHandler(CollectorAddValueHandler addValueHandler)=0</a>	28
5.5.2.6	<a href="#">setClearHandler(CollectorClearHandler clearHandler)=0</a>	29
5.6	<a href="#">CollectorDatafile_if Class Reference</a>	29
5.6.1	<a href="#">Detailed Description</a>	30
5.6.2	<a href="#">Member Function Documentation</a>	31
5.6.2.1	<a href="#">getDataFilename()=0</a>	31
5.6.2.2	<a href="#">getNextValue()=0</a>	31
5.6.2.3	<a href="#">getValue(unsigned int rank)=0</a>	31
5.6.2.4	<a href="#">seekFirstValue()=0</a>	31
5.6.2.5	<a href="#">setDataFilename(std::string filename)=0</a>	31
5.7	<a href="#">CollectorDatafileDefaultImpl1 Class Reference</a>	32
5.7.1	<a href="#">Constructor &amp; Destructor Documentation</a>	33
5.7.1.1	<a href="#">CollectorDatafileDefaultImpl1()</a>	33
5.7.1.2	<a href="#">CollectorDatafileDefaultImpl1(const CollectorDatafileDefaultImpl1 &amp;orig)</a>	33
5.7.1.3	<a href="#">~CollectorDatafileDefaultImpl1()</a>	33
5.7.2	<a href="#">Member Function Documentation</a>	33
5.7.2.1	<a href="#">addValue(double value)</a>	33
5.7.2.2	<a href="#">clear()</a>	33
5.7.2.3	<a href="#">getDataFilename()</a>	33
5.7.2.4	<a href="#">getLastValue()</a>	33
5.7.2.5	<a href="#">getNextValue()</a>	33
5.7.2.6	<a href="#">getValue(unsigned int num)</a>	34
5.7.2.7	<a href="#">numElements()</a>	34
5.7.2.8	<a href="#">seekFirstValue()</a>	34
5.7.2.9	<a href="#">setAddValueHandler(CollectorAddValueHandler addValueHandler)</a>	34
5.7.2.10	<a href="#">setClearHandler(CollectorClearHandler clearHandler)</a>	34

5.7.2.11	<a href="#">setDataFilename(std::string filename)</a>	34
5.8	<a href="#">CollectorDatafileDummyImpl Class Reference</a>	35
5.8.1	<a href="#">Constructor &amp; Destructor Documentation</a>	36
5.8.1.1	<a href="#">CollectorDatafileDummyImpl()</a>	36
5.8.1.2	<a href="#">CollectorDatafileDummyImpl(const CollectorDatafileDummyImpl &amp;orig)</a>	36
5.8.1.3	<a href="#">~CollectorDatafileDummyImpl()</a>	36
5.8.2	<a href="#">Member Function Documentation</a>	36
5.8.2.1	<a href="#">addValue(double value)</a>	36
5.8.2.2	<a href="#">clear()</a>	36
5.8.2.3	<a href="#">getDataFilename()</a>	36
5.8.2.4	<a href="#">getLastValue()</a>	36
5.8.2.5	<a href="#">getNextValue()</a>	36
5.8.2.6	<a href="#">getValue(unsigned int num)</a>	37
5.8.2.7	<a href="#">numElements()</a>	37
5.8.2.8	<a href="#">seekFirstValue()</a>	37
5.8.2.9	<a href="#">setAddValueHandler(CollectorAddValueHandler addValueHandler)</a>	37
5.8.2.10	<a href="#">setClearHandler(CollectorClearHandler clearHandler)</a>	37
5.8.2.11	<a href="#">setDataFilename(std::string filename)</a>	37
5.9	<a href="#">CollectorDefaultImpl1 Class Reference</a>	38
5.9.1	<a href="#">Constructor &amp; Destructor Documentation</a>	39
5.9.1.1	<a href="#">CollectorDefaultImpl1()</a>	39
5.9.1.2	<a href="#">CollectorDefaultImpl1(const CollectorDefaultImpl1 &amp;orig)</a>	39
5.9.1.3	<a href="#">~CollectorDefaultImpl1()</a>	39
5.9.2	<a href="#">Member Function Documentation</a>	39
5.9.2.1	<a href="#">addValue(double value)</a>	39
5.9.2.2	<a href="#">clear()</a>	39
5.9.2.3	<a href="#">getLastValue()</a>	39
5.9.2.4	<a href="#">numElements()</a>	39
5.9.2.5	<a href="#">setAddValueHandler(CollectorAddValueHandler addValueHandler)</a>	39
5.9.2.6	<a href="#">setClearHandler(CollectorClearHandler clearHandler)</a>	39

5.10 CollectorDummyImpl Class Reference . . . . .	40
5.10.1 Constructor & Destructor Documentation . . . . .	41
5.10.1.1 CollectorDummyImpl() . . . . .	41
5.10.1.2 CollectorDummyImpl(const CollectorDummyImpl &orig) . . . . .	41
5.10.1.3 ~CollectorDummyImpl() . . . . .	41
5.10.2 Member Function Documentation . . . . .	41
5.10.2.1 addValue(double value) . . . . .	41
5.10.2.2 clear() . . . . .	41
5.10.2.3 getLastValue() . . . . .	41
5.10.2.4 numElements() . . . . .	41
5.10.2.5 setAddValueHandler(CollectorAddValueHandler addValueHandler) . . . . .	41
5.10.2.6 setClearHandler(CollectorClearHandler clearHandler) . . . . .	41
5.11 Create Class Reference . . . . .	42
5.11.1 Detailed Description . . . . .	43
5.11.2 Constructor & Destructor Documentation . . . . .	43
5.11.2.1 Create(Model *model) . . . . .	43
5.11.2.2 Create(const Create &orig) . . . . .	44
5.11.2.3 ~Create() . . . . .	44
5.11.3 Member Function Documentation . . . . .	44
5.11.3.1 _execute(Entity *entity) . . . . .	44
5.11.3.2 _loadInstance(std::list< std::string > words) . . . . .	44
5.11.3.3 _saveInstance() . . . . .	45
5.11.3.4 _verifySymbols(std::string *errorMessage) . . . . .	45
5.11.3.5 show() . . . . .	45
5.12 SamplerDefaultImpl1::DefaultImpl1RNG_Parameters Class Reference . . . . .	46
5.12.1 Member Data Documentation . . . . .	46
5.12.1.1 module . . . . .	46
5.12.1.2 multiplier . . . . .	46
5.12.1.3 seed . . . . .	46
5.13 Delay Class Reference . . . . .	47

5.13.1	Constructor & Destructor Documentation	48
5.13.1.1	Delay(Model *model)	48
5.13.1.2	Delay(const Delay &orig)	48
5.13.1.3	~Delay()	48
5.13.2	Member Function Documentation	48
5.13.2.1	_execute(Entity *entity)	48
5.13.2.2	_loadInstance(std::list< std::string > words)	49
5.13.2.3	_saveInstance()	49
5.13.2.4	_verifySymbols(std::string *errorMessage)	49
5.13.2.5	getDelayExpression() const	49
5.13.2.6	getDelayTimeUnit() const	50
5.13.2.7	setDelayExpression(std::string _delayExpression)	50
5.13.2.8	setDelayTimeUnit(Util::TimeUnit _delayTimeUnit)	50
5.13.2.9	show()	50
5.14	Dispose Class Reference	51
5.14.1	Constructor & Destructor Documentation	52
5.14.1.1	Dispose(Model *model)	52
5.14.1.2	Dispose(const Dispose &orig)	52
5.14.1.3	~Dispose()	52
5.14.2	Member Function Documentation	52
5.14.2.1	_execute(Entity *entity)	52
5.14.2.2	_loadInstance(std::list< std::string > words)	52
5.14.2.3	_saveInstance()	53
5.14.2.4	_verifySymbols(std::string *errorMessage)	53
5.14.2.5	isCollectStatistics() const	53
5.14.2.6	setCollectStatistics(bool _collectStatistics)	53
5.14.2.7	show()	53
5.15	ElementManager Class Reference	54
5.15.1	Detailed Description	54
5.15.2	Constructor & Destructor Documentation	54



5.15.2.1	<a href="#">ElementManager(Model *model)</a>	54
5.15.2.2	<a href="#">ElementManager(const ElementManager &amp;orig)</a>	54
5.15.2.3	<a href="#">~ElementManager()</a>	54
5.15.3	<a href="#">Member Function Documentation</a>	54
5.15.3.1	<a href="#">getElement(std::string infraTypename, Util::identification id)</a>	54
5.15.3.2	<a href="#">getElement(std::string infraTypename, std::string name)</a>	55
5.15.3.3	<a href="#">getElements(std::string infraTypename) const</a>	55
5.15.3.4	<a href="#">getElementTypenames() const</a>	56
5.15.3.5	<a href="#">insertElement(std::string infraTypename, ModelElement *infra)</a>	56
5.15.3.6	<a href="#">removeElement(std::string infraTypename, ModelElement *infra)</a>	57
5.15.3.7	<a href="#">show()</a>	58
5.16	<a href="#">ElementManager_if Class Reference</a>	58
5.16.1	<a href="#">Constructor &amp; Destructor Documentation</a>	58
5.16.1.1	<a href="#">ElementManager_if()</a>	58
5.16.1.2	<a href="#">ElementManager_if(const ElementManager_if &amp;orig)</a>	58
5.16.1.3	<a href="#">~ElementManager_if()</a>	58
5.17	<a href="#">Entity Class Reference</a>	59
5.17.1	<a href="#">Constructor &amp; Destructor Documentation</a>	60
5.17.1.1	<a href="#">Entity()</a>	60
5.17.1.2	<a href="#">Entity(const Entity &amp;orig)</a>	60
5.17.1.3	<a href="#">~Entity()</a>	60
5.17.2	<a href="#">Member Function Documentation</a>	60
5.17.2.1	<a href="#">_loadInstance(std::list&lt; std::string &gt; words)</a>	60
5.17.2.2	<a href="#">_saveInstance()</a>	60
5.17.2.3	<a href="#">_verifySymbols(std::string *errorMessage)</a>	60
5.17.2.4	<a href="#">getAttributeValue(std::string attributeName)</a>	60
5.17.2.5	<a href="#">getEntityType() const</a>	60
5.17.2.6	<a href="#">setAttributeValue(std::string attributeName, double value)</a>	61
5.17.2.7	<a href="#">setEntityType(EntityType *entityType)</a>	61
5.17.2.8	<a href="#">show()</a>	61

5.18 EntityType Class Reference . . . . .	62
5.18.1 Constructor & Destructor Documentation . . . . .	63
5.18.1.1 EntityType(ElementManager *elemManager) . . . . .	63
5.18.1.2 EntityType(ElementManager *elemManager, std::string name) . . . . .	63
5.18.1.3 EntityType(ElementManager *elemManager, std::string name, std::string initialPicture, double initialWaitingCost, double initialVACost, double initialNVACost, double initialOtherCost) . . . . .	64
5.18.1.4 EntityType(const EntityType &orig) . . . . .	64
5.18.1.5 ~EntityType() . . . . .	64
5.18.2 Member Function Documentation . . . . .	64
5.18.2.1 _loadInstance(std::list< std::string > words) . . . . .	64
5.18.2.2 _saveInstance() . . . . .	64
5.18.2.3 _verifySymbols(std::string *errorMessage) . . . . .	65
5.18.2.4 getCstatNVATime() const . . . . .	65
5.18.2.5 getCstatOtherTime() const . . . . .	65
5.18.2.6 getCstatTimeInSystem() const . . . . .	65
5.18.2.7 getCstatTransferTime() const . . . . .	65
5.18.2.8 getCstatVATime() const . . . . .	65
5.18.2.9 getCstatWaitingTime() const . . . . .	65
5.18.2.10 getInitialNVACost() const . . . . .	65
5.18.2.11 getInitialOtherCost() const . . . . .	65
5.18.2.12 getInitialPicture() const . . . . .	65
5.18.2.13 getInitialVACost() const . . . . .	65
5.18.2.14 getInitialWaitingCost() const . . . . .	65
5.18.2.15 setInitialNVACost(double _initialNVACost) . . . . .	65
5.18.2.16 setInitialOtherCost(double _initialOtherCost) . . . . .	65
5.18.2.17 setInitialPicture(std::string _initialPicture) . . . . .	65
5.18.2.18 setInitialVACost(double _initialVACost) . . . . .	65
5.18.2.19 setInitialWaitingCost(double _initialWaitingCost) . . . . .	65
5.18.2.20 show() . . . . .	65
5.19 Event Class Reference . . . . .	66

5.19.1	Constructor & Destructor Documentation . . . . .	66
5.19.1.1	Event(double time, Entity *entity, ModelComponent *component) . . . . .	66
5.19.1.2	Event(const Event &orig) . . . . .	66
5.19.1.3	~Event() . . . . .	66
5.19.2	Member Function Documentation . . . . .	66
5.19.2.1	getComponent() const . . . . .	66
5.19.2.2	getEntity() const . . . . .	66
5.19.2.3	getTime() const . . . . .	67
5.19.2.4	show() . . . . .	67
5.20	ExperimentDesign_if Class Reference . . . . .	68
5.20.1	Detailed Description . . . . .	68
5.20.2	Member Function Documentation . . . . .	68
5.20.2.1	calculateContributionAndCoefficients()=0 . . . . .	68
5.20.2.2	generate2krScenarioExperiments()=0 . . . . .	68
5.20.2.3	getContributions() const =0 . . . . .	68
5.20.2.4	getProcessAnalyser() const =0 . . . . .	69
5.21	ExperimentDesignDummyImpl Class Reference . . . . .	69
5.21.1	Constructor & Destructor Documentation . . . . .	70
5.21.1.1	ExperimentDesignDummyImpl() . . . . .	70
5.21.1.2	ExperimentDesignDummyImpl(const ExperimentDesignDummyImpl &orig) . . . . .	70
5.21.1.3	~ExperimentDesignDummyImpl() . . . . .	70
5.21.2	Member Function Documentation . . . . .	70
5.21.2.1	calculateContributionAndCoefficients() . . . . .	70
5.21.2.2	generate2krScenarioExperiments() . . . . .	70
5.21.2.3	getContributions() const . . . . .	70
5.21.2.4	getProcessAnalyser() const . . . . .	70
5.22	FactorOrInteractionContribution Class Reference . . . . .	70
5.22.1	Detailed Description . . . . .	71
5.22.2	Constructor & Destructor Documentation . . . . .	71
5.22.2.1	FactorOrInteractionContribution(double contribution, double modelCoefficient, std::list< SimulationControl * > *controls) . . . . .	71

5.22.2.2	FactorOrInteractionContribution(const FactorOrInteractionContribution &orig)	71
5.22.2.3	~FactorOrInteractionContribution()	71
5.22.3	Member Function Documentation	71
5.22.3.1	getContribution() const	71
5.22.3.2	getControls() const	71
5.22.3.3	getModelCoefficient() const	71
5.23	Fitter_if Class Reference	71
5.23.1	Member Function Documentation	72
5.23.1.1	fitAll(double *sqerror, std::string *name)=0	72
5.23.1.2	fitBeta(double *sqerror, double *alpha, double *beta, double *infLimit, double *supLimit)=0	72
5.23.1.3	fitErlang(double *sqerror, double *avg, double *m)=0	72
5.23.1.4	fitExpo(double *sqerror, double *avg1)=0	72
5.23.1.5	fitNormal(double *sqerror, double *avg, double *stddev)=0	73
5.23.1.6	fitTriangular(double *sqerror, double *min, double *mo, double *max)=0	73
5.23.1.7	fitUniform(double *sqerror, double *min, double *max)=0	73
5.23.1.8	fitWeibull(double *sqerror, double *alpha, double *scale)=0	73
5.23.1.9	getDataFilename()=0	74
5.23.1.10	isNormalDistributed(double confidencelevel)=0	74
5.23.1.11	setDataFilename(std::string dataFilename)=0	74
5.24	FitterDummyImpl Class Reference	74
5.24.1	Constructor & Destructor Documentation	75
5.24.1.1	FitterDummyImpl()	75
5.24.1.2	FitterDummyImpl(const FitterDummyImpl &orig)	75
5.24.1.3	~FitterDummyImpl()	75
5.24.2	Member Function Documentation	75
5.24.2.1	fitAll(double *sqerror, std::string *name)	75
5.24.2.2	fitBeta(double *sqerror, double *alpha, double *beta, double *infLimit, double *supLimit)	76
5.24.2.3	fitErlang(double *sqerror, double *avg, double *m)	76
5.24.2.4	fitExpo(double *sqerror, double *avg1)	76

5.24.2.5	<code>fitNormal(double *sqrrerror, double *avg, double *stddev)</code>	76
5.24.2.6	<code>fitTriangular(double *sqrrerror, double *min, double *mo, double *max)</code>	76
5.24.2.7	<code>fitUniform(double *sqrrerror, double *min, double *max)</code>	76
5.24.2.8	<code>fitWeibull(double *sqrrerror, double *alpha, double *scale)</code>	76
5.24.2.9	<code>getDataFilename()</code>	76
5.24.2.10	<code>isNormalDistributed(double confidencelevel)</code>	76
5.24.2.11	<code>setDataFilename(std::string dataFilename)</code>	76
5.25	GenesysApplication_if Class Reference	77
5.25.1	Member Function Documentation	77
5.25.1.1	<code>main(int argc, char **argv)=0</code>	77
5.26	HypothesisTester_if Class Reference	78
5.26.1	Detailed Description	78
5.26.2	Member Enumeration Documentation	78
5.26.2.1	<code>H1Comparition</code>	78
5.26.3	Member Function Documentation	79
5.26.3.1	<code>getDataFilename()=0</code>	79
5.26.3.2	<code>setDataFilename(std::string dataFilename)=0</code>	79
5.26.3.3	<code>testAverage(double confidencelevel, double avg, H1Comparition comp)=0</code>	79
5.26.3.4	<code>testAverage(double confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp)=0</code>	79
5.26.3.5	<code>testProportion(double confidencelevel, double prop, H1Comparition comp)=0</code>	79
5.26.3.6	<code>testProportion(double confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp)=0</code>	80
5.26.3.7	<code>testVariance(double confidencelevel, double var, H1Comparition comp)=0</code>	80
5.26.3.8	<code>testVariance(double confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp)=0</code>	80
5.27	HypothesisTesterDummyImpl Class Reference	80
5.27.1	Constructor & Destructor Documentation	81
5.27.1.1	<code>HypothesisTesterDummyImpl()</code>	81
5.27.1.2	<code>HypothesisTesterDummyImpl(const HypothesisTesterDummyImpl &amp;orig)</code>	81
5.27.1.3	<code>~HypothesisTesterDummyImpl()</code>	81
5.27.2	Member Function Documentation	81

5.27.2.1	<code>getDataFilename()</code> . . . . .	81
5.27.2.2	<code>setDataFilename(std::string dataFilename)</code> . . . . .	82
5.27.2.3	<code>testAverage(double confidencelevel, double avg, H1Comparition comp)</code> . . . . .	82
5.27.2.4	<code>testAverage(double confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp)</code> . . . . .	82
5.27.2.5	<code>testProportion(double confidencelevel, double prop, H1Comparition comp)</code> . . . . .	82
5.27.2.6	<code>testProportion(double confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp)</code> . . . . .	82
5.27.2.7	<code>testVariance(double confidencelevel, double var, H1Comparition comp)</code> . . . . .	82
5.27.2.8	<code>testVariance(double confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp)</code> . . . . .	82
5.28	<code>Integrator_if</code> Class Reference . . . . .	83
5.28.1	Member Function Documentation . . . . .	83
5.28.1.1	<code>getPrecision()=0</code> . . . . .	83
5.28.1.2	<code>integrate(double min, double max, double(*f)(double, double), double p2)=0</code> . . . . .	83
5.28.1.3	<code>integrate(double min, double max, double(*f)(double, double, double), double p2, double p3)=0</code> . . . . .	84
5.28.1.4	<code>integrate(double min, double max, double(*f)(double, double, double, double), double p2, double p3, double p4)=0</code> . . . . .	84
5.28.1.5	<code>integrate(double min, double max, double(*f)(double, double, double, double, double), double p2, double p3, double p4, double p5)=0</code> . . . . .	84
5.28.1.6	<code>setPrecision(double e)=0</code> . . . . .	84
5.29	<code>IntegratorDefaultImpl1</code> Class Reference . . . . .	84
5.29.1	Constructor & Destructor Documentation . . . . .	85
5.29.1.1	<code>IntegratorDefaultImpl1()</code> . . . . .	85
5.29.1.2	<code>IntegratorDefaultImpl1(const IntegratorDefaultImpl1 &amp;orig)</code> . . . . .	85
5.29.1.3	<code>~IntegratorDefaultImpl1()</code> . . . . .	85
5.29.2	Member Function Documentation . . . . .	85
5.29.2.1	<code>getPrecision()</code> . . . . .	85
5.29.2.2	<code>integrate(double min, double max, double(*f)(double, double), double p2)</code> . . . . .	85
5.29.2.3	<code>integrate(double min, double max, double(*f)(double, double, double), double p2, double p3)</code> . . . . .	86
5.29.2.4	<code>integrate(double min, double max, double(*f)(double, double, double, double), double p2, double p3, double p4)</code> . . . . .	86

5.29.2.5	integrate(double min, double max, double(*f)(double, double, double, double, double), double p2, double p3, double p4, double p5) . . . . .	86
5.29.2.6	setPrecision(double e) . . . . .	86
5.30	IntegratorDummyImpl Class Reference . . . . .	86
5.30.1	Constructor & Destructor Documentation . . . . .	87
5.30.1.1	IntegratorDummyImpl() . . . . .	87
5.30.1.2	IntegratorDummyImpl(const IntegratorDummyImpl &orig) . . . . .	87
5.30.1.3	~IntegratorDummyImpl() . . . . .	87
5.30.2	Member Function Documentation . . . . .	87
5.30.2.1	getPrecision() . . . . .	87
5.30.2.2	integrate(double min, double max, double(*f)(double, double), double p2) . . . . .	87
5.30.2.3	integrate(double min, double max, double(*f)(double, double, double), double p2, double p3) . . . . .	88
5.30.2.4	integrate(double min, double max, double(*f)(double, double, double, double), double p2, double p3, double p4) . . . . .	88
5.30.2.5	integrate(double min, double max, double(*f)(double, double, double, double, double), double p2, double p3, double p4, double p5) . . . . .	88
5.30.2.6	setPrecision(double e) . . . . .	88
5.31	LinkedBy Class Reference . . . . .	88
5.31.1	Constructor & Destructor Documentation . . . . .	89
5.31.1.1	LinkedBy() . . . . .	89
5.31.1.2	LinkedBy(const LinkedBy &orig) . . . . .	89
5.31.1.3	~LinkedBy() . . . . .	89
5.31.2	Member Function Documentation . . . . .	89
5.31.2.1	addLink() . . . . .	89
5.31.2.2	isLinked() . . . . .	89
5.31.2.3	removeLink() . . . . .	89
5.32	List< T > Class Template Reference . . . . .	89
5.32.1	Member Typedef Documentation . . . . .	90
5.32.1.1	CompFunc . . . . .	90
5.32.2	Constructor & Destructor Documentation . . . . .	90
5.32.2.1	List() . . . . .	90

5.32.2.2	List(const List &orig)	90
5.32.2.3	~List()	90
5.32.3	Member Function Documentation	90
5.32.3.1	actual()	90
5.32.3.2	clear()	90
5.32.3.3	create()	91
5.32.3.4	create(U arg)	91
5.32.3.5	empty()	91
5.32.3.6	find(T element)	91
5.32.3.7	first()	91
5.32.3.8	getList() const	92
5.32.3.9	insert(T element)	92
5.32.3.10	last()	93
5.32.3.11	next()	93
5.32.3.12	pop_front()	93
5.32.3.13	previous()	93
5.32.3.14	remove(T element)	93
5.32.3.15	setSortFunc(CompFunc_t sortFunc)	94
5.32.3.16	show()	94
5.32.3.17	size()	95
5.32.3.18	sort(Compare comp)	95
5.33	Model Class Reference	95
5.33.1	Detailed Description	96
5.33.2	Constructor & Destructor Documentation	96
5.33.2.1	Model(Simulator *simulator)	96
5.33.2.2	Model(const Model &orig)	97
5.33.2.3	~Model()	97
5.33.3	Member Function Documentation	97
5.33.3.1	checkModel()	97
5.33.3.2	getComponents() const	98



5.33.3.3	<code>getControls()</code> const	98
5.33.3.4	<code>getElementManager()</code> const	99
5.33.3.5	<code>getEvents()</code> const	99
5.33.3.6	<code>getId()</code> const	100
5.33.3.7	<code>getInfos()</code> const	100
5.33.3.8	<code>getEventManager()</code> const	100
5.33.3.9	<code>getParent()</code> const	100
5.33.3.10	<code>getResponses()</code> const	100
5.33.3.11	<code>getSimulation()</code> const	101
5.33.3.12	<code>getTracer()</code> const	101
5.33.3.13	<code>loadModel(std::string filename)</code>	102
5.33.3.14	<code>parseExpression(const std::string expression)</code>	102
5.33.3.15	<code>parseExpression(const std::string expression, bool *success, std::string *error← Message)</code>	103
5.33.3.16	<code>removeEntity(Entity *entity, bool collectStatistics)</code>	103
5.33.3.17	<code>saveModel(std::string filename)</code>	104
5.33.3.18	<code>sendEntityToComponent(Entity *entity, ModelComponent *component, double timeDelay)</code>	104
5.33.3.19	<code>showReports()</code>	105
5.33.3.20	<code>verifySymbol(std::string componentName, std::string expressionName, std::string expression, std::string expressionResult, bool mandatory)</code>	105
5.34	<code>ModelChecker_</code> if Class Reference	105
5.34.1	Detailed Description	106
5.34.2	Member Function Documentation	106
5.34.2.1	<code>checkActivationCode()</code> =0	106
5.34.2.2	<code>checkAll()</code> =0	106
5.34.2.3	<code>checkAndAddInternalLiterals()</code> =0	106
5.34.2.4	<code>checkConnected()</code> =0	106
5.34.2.5	<code>checkPathway()</code> =0	106
5.34.2.6	<code>checkSymbols()</code> =0	106
5.34.2.7	<code>verifySymbol(std::string componentName, std::string expressionName, std::string expression, std::string expressionResult, bool mandatory)</code> =0	107

5.35	ModelCheckerDummyImpl Class Reference	107
5.35.1	Detailed Description	108
5.35.2	Constructor & Destructor Documentation	108
5.35.2.1	ModelCheckerDummyImpl(Model *model)	108
5.35.2.2	ModelCheckerDummyImpl(const ModelCheckerDummyImpl &orig)	108
5.35.2.3	~ModelCheckerDummyImpl()	108
5.35.3	Member Function Documentation	108
5.35.3.1	checkActivationCode()	108
5.35.3.2	checkAll()	109
5.35.3.3	checkAndAddInternalLiterals()	109
5.35.3.4	checkConnected()	110
5.35.3.5	checkPathway()	110
5.35.3.6	checkSymbols()	110
5.35.3.7	verifySymbol(std::string componentName, std::string expressionName, std::string expression, std::string expressionResult, bool mandatory)	111
5.36	ModelComponent Class Reference	111
5.36.1	Detailed Description	113
5.36.2	Constructor & Destructor Documentation	113
5.36.2.1	ModelComponent(Model *model)	113
5.36.2.2	ModelComponent(const ModelComponent &orig)	113
5.36.2.3	~ModelComponent()	113
5.36.3	Member Function Documentation	113
5.36.3.1	_execute(Entity *entity)=0	113
5.36.3.2	_saveInstance()	114
5.36.3.3	_saveInstance(std::string type)	114
5.36.3.4	Execute(Entity *entity, ModelComponent *component)	115
5.36.3.5	getNextComponents() const	115
5.36.3.6	SaveInstance(ModelComponent *component)	116
5.36.3.7	show()	116
5.36.3.8	VerifySymbols(ModelComponent *component, std::string *errorMessage)	117
5.36.4	Member Data Documentation	117

5.36.4.1	<code>_model</code>	117
5.37	ModelComponentManager_if Class Reference	117
5.37.1	Constructor & Destructor Documentation	118
5.37.1.1	<code>ModelComponentManager_if()</code>	118
5.37.1.2	<code>ModelComponentManager_if(const ModelComponentManager_if &amp;orig)</code>	118
5.37.1.3	<code>~ModelComponentManager_if()</code>	118
5.38	ModelElement Class Reference	118
5.38.1	Detailed Description	119
5.38.2	Constructor & Destructor Documentation	119
5.38.2.1	<code>ModelElement(std::string elementTypename)</code>	119
5.38.2.2	<code>ModelElement(const ModelElement &amp;orig)</code>	119
5.38.2.3	<code>~ModelElement()</code>	119
5.38.3	Member Function Documentation	120
5.38.3.1	<code>_loadInstance(std::list&lt; std::string &gt; words)=0</code>	120
5.38.3.2	<code>_saveInstance()</code>	120
5.38.3.3	<code>_saveInstance(std::string type)</code>	120
5.38.3.4	<code>_verifySymbols(std::string *errorMessage)=0</code>	121
5.38.3.5	<code>getId() const</code>	121
5.38.3.6	<code>getName() const</code>	122
5.38.3.7	<code>LoadInstance(std::list&lt; std::string &gt; words)</code>	122
5.38.3.8	<code>SaveInstance(ModelElement *element)</code>	122
5.38.3.9	<code>setName(std::string _name)</code>	122
5.38.3.10	<code>show()</code>	122
5.38.3.11	<code>VerifySymbols(ModelElement *element, std::string *errorMessage)</code>	123
5.38.4	Member Data Documentation	123
5.38.4.1	<code>_id</code>	123
5.38.4.2	<code>_name</code>	123
5.39	ModelInfo Class Reference	123
5.39.1	Detailed Description	124
5.39.2	Constructor & Destructor Documentation	124

5.39.2.1	ModelInfo()	124
5.39.2.2	ModelInfo(const ModelInfo &orig)	124
5.39.2.3	~ModelInfo()	124
5.39.3	Member Function Documentation	124
5.39.3.1	getAnalystName() const	124
5.39.3.2	getDescription() const	124
5.39.3.3	getName() const	124
5.39.3.4	getNumberOfReplications() const	125
5.39.3.5	getProjectTitle() const	125
5.39.3.6	getReplicationLength() const	125
5.39.3.7	getReplicationLengthTimeUnit() const	125
5.39.3.8	getTerminatingCondition() const	126
5.39.3.9	getVersion() const	126
5.39.3.10	getWarmUpPeriod() const	126
5.39.3.11	getWarmUpPeriodTimeUnit() const	126
5.39.3.12	setAnalystName(std::string _analystName)	126
5.39.3.13	setDescription(std::string _description)	126
5.39.3.14	setName(std::string _name)	127
5.39.3.15	setNumberOfReplications(unsigned int _numberOfReplications)	127
5.39.3.16	setProjectTitle(std::string _projectTitle)	127
5.39.3.17	setReplicationLength(double _replicationLength)	127
5.39.3.18	setReplicationLengthTimeUnit(Util::TimeUnit _replicationLengthTimeUnit)	127
5.39.3.19	setTerminatingCondition(std::string _terminatingCondition)	128
5.39.3.20	setVersion(std::string _version)	128
5.39.3.21	setWarmUpPeriod(double _warmUpPeriod)	128
5.39.3.22	setWarmUpPeriodTimeUnit(Util::TimeUnit _warmUpPeriodTimeUnit)	128
5.40	ModelPersistence_if Class Reference	128
5.40.1	Detailed Description	128
5.40.2	Member Function Documentation	129
5.40.2.1	isSaved()=0	129

5.40.2.2	<code>load(std::string filename)=0</code>	129
5.40.2.3	<code>loadAsTXT(std::string filename)=0</code>	129
5.40.2.4	<code>loadAsXML(std::string filename)=0</code>	129
5.40.2.5	<code>save(std::string filename)=0</code>	129
5.40.2.6	<code>saveAsTXT(std::string filename)=0</code>	130
5.40.2.7	<code>saveAsXML(std::string filename)=0</code>	130
5.41	<b>ModelPersistenceDummyImpl Class Reference</b>	130
5.41.1	<b>Constructor &amp; Destructor Documentation</b>	131
5.41.1.1	<code>ModelPersistenceDummyImpl(Model *model)</code>	131
5.41.1.2	<code>ModelPersistenceDummyImpl(const ModelPersistenceDummyImpl &amp;orig)</code>	131
5.41.1.3	<code>~ModelPersistenceDummyImpl()</code>	131
5.41.2	<b>Member Function Documentation</b>	131
5.41.2.1	<code>isSaved()</code>	131
5.41.2.2	<code>load(std::string filename)</code>	131
5.41.2.3	<code>loadAsTXT(std::string filename)</code>	132
5.41.2.4	<code>loadAsXML(std::string filename)</code>	132
5.41.2.5	<code>save(std::string filename)</code>	132
5.41.2.6	<code>saveAsTXT(std::string filename)</code>	133
5.41.2.7	<code>saveAsXML(std::string filename)</code>	133
5.42	<b>ModelSimulation Class Reference</b>	134
5.42.1	<b>Detailed Description</b>	134
5.42.2	<b>Constructor &amp; Destructor Documentation</b>	134
5.42.2.1	<code>ModelSimulation(Model *model)</code>	134
5.42.2.2	<code>ModelSimulation(const ModelSimulation &amp;orig)</code>	135
5.42.2.3	<code>~ModelSimulation()</code>	135
5.42.3	<b>Member Function Documentation</b>	135
5.42.3.1	<code>getCurrentComponent() const</code>	135
5.42.3.2	<code>getCurrentEntity() const</code>	135
5.42.3.3	<code>getCurrentReplicationNumber() const</code>	135
5.42.3.4	<code>getSimulatedTime() const</code>	135

5.42.3.5	<code>isInitializeStatistics() const</code>	136
5.42.3.6	<code>isInitializeSystem() const</code>	136
5.42.3.7	<code>isPauseOnEvent() const</code>	136
5.42.3.8	<code>isPauseOnReplication() const</code>	136
5.42.3.9	<code>isRunning() const</code>	136
5.42.3.10	<code>isStepByStep() const</code>	136
5.42.3.11	<code>pauseSimulation()</code>	136
5.42.3.12	<code>restartSimulation()</code>	136
5.42.3.13	<code>setInitializeStatistics(bool _initializeStatistics)</code>	136
5.42.3.14	<code>setInitializeSystem(bool _initializeSystem)</code>	136
5.42.3.15	<code>setPauseOnEvent(bool _pauseOnEvent)</code>	136
5.42.3.16	<code>setPauseOnReplication(bool _pauseBetweenReplications)</code>	136
5.42.3.17	<code>setStepByStep(bool _stepByStep)</code>	136
5.42.3.18	<code>startSimulation()</code>	136
5.42.3.19	<code>stepSimulation()</code>	137
5.42.3.20	<code>stopSimulation()</code>	137
5.43	<code>SamplerDummyImpl::MyRNG_Parameters Class Reference</code>	137
5.43.1	<code>Member Data Documentation</code>	138
5.43.1.1	<code>module</code>	138
5.43.1.2	<code>multiplier</code>	138
5.43.1.3	<code>seed</code>	138
5.44	<code>OnEventManager Class Reference</code>	138
5.44.1	<code>Detailed Description</code>	139
5.44.2	<code>Constructor &amp; Destructor Documentation</code>	139
5.44.2.1	<code>OnEventManager()</code>	139
5.44.2.2	<code>OnEventManager(const OnEventManager &amp;orig)</code>	139
5.44.2.3	<code>~OnEventManager()</code>	139
5.44.3	<code>Member Function Documentation</code>	139
5.44.3.1	<code>addOnProcessEventHandler(simulationEventHandler EventHandler)</code>	139
5.44.3.2	<code>addOnReplicationEndHandler(simulationEventHandler EventHandler)</code>	139

5.44.3.3	<a href="#">addOnReplicationStartHandler(simulationEventHandler EventHandler)</a>	140
5.44.3.4	<a href="#">addOnReplicationStepHandler(simulationEventHandler EventHandler)</a>	140
5.44.3.5	<a href="#">addOnSimulationEndHandler(simulationEventHandler EventHandler)</a>	140
5.44.3.6	<a href="#">addOnSimulationStartHandler(simulationEventHandler EventHandler)</a>	140
5.44.3.7	<a href="#">NotifyProcessEventHandlers(SimulationEvent *se)</a>	140
5.44.3.8	<a href="#">NotifyReplicationEndHandlers(SimulationEvent *se)</a>	141
5.44.3.9	<a href="#">NotifyReplicationStartHandlers(SimulationEvent *se)</a>	141
5.44.3.10	<a href="#">NotifyReplicationStepHandlers(SimulationEvent *se)</a>	141
5.44.3.11	<a href="#">NotifySimulationEndHandlers(SimulationEvent *se)</a>	141
5.44.3.12	<a href="#">NotifySimulationStartHandlers(SimulationEvent *se)</a>	142
5.45	<a href="#">Parser_if Class Reference</a>	142
5.45.1	<a href="#">Member Function Documentation</a>	142
5.45.1.1	<a href="#">getErrorMessage()=0</a>	142
5.45.1.2	<a href="#">parse(const std::string expression)=0</a>	143
5.45.1.3	<a href="#">parse(const std::string expression, bool *success, std::string *errorMessage)=0</a>	143
5.46	<a href="#">ParserDefaultImpl1 Class Reference</a>	143
5.46.1	<a href="#">Constructor &amp; Destructor Documentation</a>	144
5.46.1.1	<a href="#">ParserDefaultImpl1(Model *model)</a>	144
5.46.1.2	<a href="#">ParserDefaultImpl1(const ParserDefaultImpl1 &amp;orig)</a>	144
5.46.1.3	<a href="#">~ParserDefaultImpl1()</a>	144
5.46.2	<a href="#">Member Function Documentation</a>	144
5.46.2.1	<a href="#">getErrorMessage()</a>	144
5.46.2.2	<a href="#">parse(const std::string expression)</a>	144
5.46.2.3	<a href="#">parse(const std::string expression, bool *success, std::string *errorMessage)</a>	145
5.47	<a href="#">ParserDummyImpl Class Reference</a>	145
5.47.1	<a href="#">Constructor &amp; Destructor Documentation</a>	146
5.47.1.1	<a href="#">ParserDummyImpl(Model *model)</a>	146
5.47.1.2	<a href="#">ParserDummyImpl(const ParserDummyImpl &amp;orig)</a>	146
5.47.1.3	<a href="#">~ParserDummyImpl()</a>	146
5.47.2	<a href="#">Member Function Documentation</a>	146

5.47.2.1	<code>getErrorMessage()</code>	146
5.47.2.2	<code>parse(const std::string expression)</code>	146
5.47.2.3	<code>parse(const std::string expression, bool *success, std::string *errorMessage)</code>	147
5.48	Plugin Class Reference	147
5.48.1	Detailed Description	147
5.48.2	Constructor & Destructor Documentation	148
5.48.2.1	<code>Plugin(std::string name, bool source, bool drain)</code>	148
5.48.2.2	<code>Plugin(const Plugin &amp;orig)</code>	148
5.48.2.3	<code>~Plugin()</code>	148
5.48.3	Member Function Documentation	148
5.48.3.1	<code>isDrain() const</code>	148
5.48.3.2	<code>isSource() const</code>	148
5.49	ProbDistrib Class Reference	148
5.49.1	Member Function Documentation	149
5.49.1.1	<code>beta(double x, double alpha, double beta)</code>	149
5.49.1.2	<code>erlang(double x, double mean, double M)</code>	149
5.49.1.3	<code>exponential(double x, double mean)</code>	149
5.49.1.4	<code>gamma(double x, double mean, double alpha)</code>	149
5.49.1.5	<code>logNormal(double x, double mean, double stddev)</code>	149
5.49.1.6	<code>normal(double x, double mean, double stddev)</code>	149
5.49.1.7	<code>triangular(double x, double min, double mode, double max)</code>	149
5.49.1.8	<code>uniform(double x, double min, double max)</code>	149
5.49.1.9	<code>weibull(double x, double alpha, double scale)</code>	149
5.50	ProcessAnalyser_if Class Reference	150
5.50.1	Detailed Description	150
5.50.2	Member Function Documentation	150
5.50.2.1	<code>addTraceSimulationHandler(traceSimulationProcessListener traceSimulation↔ ProcessListener)=0</code>	150
5.50.2.2	<code>extractControlsFromModel(std::string modelFilename) const =0</code>	150
5.50.2.3	<code>extractResponsesFromModel(std::string modelFilename) const =0</code>	151
5.50.2.4	<code>getControls() const =0</code>	151



5.50.2.5	<code>getResponses() const =0</code>	151
5.50.2.6	<code>getScenarios() const =0</code>	151
5.50.2.7	<code>startSimulation()=0</code>	151
5.50.2.8	<code>startSimulationOfScenario(SimulationScenario *scenario)=0</code>	151
5.50.2.9	<code>stopSimulation()=0</code>	151
5.51	ProcessAnalyserDummyImpl Class Reference	152
5.51.1	Constructor & Destructor Documentation	153
5.51.1.1	<code>ProcessAnalyserDummyImpl()</code>	153
5.51.1.2	<code>ProcessAnalyserDummyImpl(const ProcessAnalyserDummyImpl &amp;orig)</code>	153
5.51.1.3	<code>~ProcessAnalyserDummyImpl()</code>	153
5.51.2	Member Function Documentation	153
5.51.2.1	<code>addTraceSimulationHandler(traceSimulationProcessListener traceSimulation↔ ProcessListener)</code>	153
5.51.2.2	<code>extractControlsFromModel(std::string modelFilename) const</code>	153
5.51.2.3	<code>extractResponsesFromModel(std::string modelFilename) const</code>	153
5.51.2.4	<code>getControls() const</code>	153
5.51.2.5	<code>getResponses() const</code>	153
5.51.2.6	<code>getScenarios() const</code>	153
5.51.2.7	<code>startSimulation()</code>	153
5.51.2.8	<code>startSimulationOfScenario(SimulationScenario *scenario)</code>	154
5.51.2.9	<code>stopSimulation()</code>	154
5.52	Queue Class Reference	154
5.52.1	Member Enumeration Documentation	155
5.52.1.1	<code>OrderRule</code>	155
5.52.2	Constructor & Destructor Documentation	156
5.52.2.1	<code>Queue(ElementManager *elems)</code>	156
5.52.2.2	<code>Queue(ElementManager *elems, std::string name)</code>	156
5.52.2.3	<code>Queue(const Queue &amp;orig)</code>	156
5.52.2.4	<code>~Queue()</code>	156
5.52.3	Member Function Documentation	156
5.52.3.1	<code>_loadInstance(std::list&lt; std::string &gt; words)</code>	156

5.52.3.2	<code>_saveInstance()</code>	157
5.52.3.3	<code>_verifySymbols(std::string *errorMessage)</code>	157
5.52.3.4	<code>first()</code>	157
5.52.3.5	<code>getAttributeName() const</code>	158
5.52.3.6	<code>getOrderRule() const</code>	158
5.52.3.7	<code>insertElement(Waiting *element)</code>	158
5.52.3.8	<code>removeElement(Waiting *element, double tnow)</code>	159
5.52.3.9	<code>setAttributeName(std::string _attributeName)</code>	159
5.52.3.10	<code>setOrderRule(OrderRule _orderRule)</code>	159
5.52.3.11	<code>show()</code>	160
5.52.3.12	<code>size()</code>	160
5.53	Release Class Reference	161
5.53.1	Constructor & Destructor Documentation	162
5.53.1.1	<code>Release(Model *model)</code>	162
5.53.1.2	<code>Release(const Release &amp;orig)</code>	162
5.53.1.3	<code>~Release()</code>	162
5.53.2	Member Function Documentation	162
5.53.2.1	<code>_execute(Entity *entity)</code>	162
5.53.2.2	<code>_loadInstance(std::list&lt; std::string &gt; words)</code>	163
5.53.2.3	<code>_saveInstance()</code>	163
5.53.2.4	<code>_verifySymbols(std::string *errorMessage)</code>	163
5.53.2.5	<code>getPriority() const</code>	164
5.53.2.6	<code>getQuantity() const</code>	164
5.53.2.7	<code>getResource() const</code>	164
5.53.2.8	<code>getResourceType() const</code>	164
5.53.2.9	<code>getRule() const</code>	164
5.53.2.10	<code>getSaveAttribute() const</code>	164
5.53.2.11	<code>setPriority(unsigned short _priority)</code>	164
5.53.2.12	<code>setQuantity(std::string _quantity)</code>	164
5.53.2.13	<code>setResource(Resource *_resource)</code>	164

5.53.2.14	<a href="#">setResourceType(Resource::ResourceType _resourceType)</a>	164
5.53.2.15	<a href="#">setRule(Resource::ResourceRule _rule)</a>	164
5.53.2.16	<a href="#">setSaveAttribute(std::string _saveAttribute)</a>	164
5.53.2.17	<a href="#">show()</a>	164
5.54	<a href="#">Resource Class Reference</a>	165
5.54.1	<a href="#">Member Enumeration Documentation</a>	166
5.54.1.1	<a href="#">ResourceRule</a>	166
5.54.1.2	<a href="#">ResourceState</a>	166
5.54.1.3	<a href="#">ResourceType</a>	167
5.54.2	<a href="#">Constructor &amp; Destructor Documentation</a>	167
5.54.2.1	<a href="#">Resource(ElementManager *elems)</a>	167
5.54.2.2	<a href="#">Resource(ElementManager *elems, std::string name)</a>	167
5.54.2.3	<a href="#">Resource(const Resource &amp;orig)</a>	167
5.54.2.4	<a href="#">~Resource()</a>	167
5.54.3	<a href="#">Member Function Documentation</a>	167
5.54.3.1	<a href="#">_loadInstance(std::list&lt; std::string &gt; words)</a>	167
5.54.3.2	<a href="#">_saveInstance()</a>	168
5.54.3.3	<a href="#">_verifySymbols(std::string *errorMessage)</a>	168
5.54.3.4	<a href="#">getCapacity() const</a>	168
5.54.3.5	<a href="#">getCostBusyHour() const</a>	168
5.54.3.6	<a href="#">getCostIdleHour() const</a>	168
5.54.3.7	<a href="#">getCostPerUse() const</a>	168
5.54.3.8	<a href="#">getNumberBusy() const</a>	168
5.54.3.9	<a href="#">getNumberOut() const</a>	169
5.54.3.10	<a href="#">getResourceState() const</a>	169
5.54.3.11	<a href="#">release(unsigned int quantity, double tnow)</a>	169
5.54.3.12	<a href="#">seize(unsigned int quantity, double tnow)</a>	169
5.54.3.13	<a href="#">setCapacity(unsigned int _capacity)</a>	170
5.54.3.14	<a href="#">setCostBusyHour(double _costBusyHour)</a>	170
5.54.3.15	<a href="#">setCostIdleHour(double _costIdleHour)</a>	170

5.54.3.16	setCostPerUse(double _costPerUse)	170
5.54.3.17	setResourceState(ResourceState _resourceState)	170
5.54.3.18	show()	170
5.55	Sampler_if::RNG_Parameters Class Reference	171
5.55.1	Detailed Description	171
5.56	Sampler_if Class Reference	171
5.56.1	Detailed Description	172
5.56.2	Member Function Documentation	172
5.56.2.1	getRNGparameters() const =0	172
5.56.2.2	random()=0	172
5.56.2.3	sampleBeta(double alpha, double beta, double infLimit, double supLimit)=0	172
5.56.2.4	sampleDiscrete(double value, double acumProb,...)=0	173
5.56.2.5	sampleErlang(double mean, int M)=0	173
5.56.2.6	sampleExponential(double mean)=0	173
5.56.2.7	sampleGamma(double mean, double alpha)=0	173
5.56.2.8	sampleLogNormal(double mean, double stddev)=0	173
5.56.2.9	sampleNormal(double mean, double stddev)=0	173
5.56.2.10	sampleTriangular(double min, double mode, double max)=0	174
5.56.2.11	sampleUniform(double min, double max)=0	174
5.56.2.12	sampleWeibull(double alpha, double scale)=0	174
5.56.2.13	setRNGparameters(RNG_Parameters *param)=0	174
5.57	SamplerDefaultImpl1 Class Reference	175
5.57.1	Constructor & Destructor Documentation	176
5.57.1.1	SamplerDefaultImpl1()	176
5.57.1.2	SamplerDefaultImpl1(const SamplerDefaultImpl1 &orig)	176
5.57.1.3	~SamplerDefaultImpl1()	176
5.57.2	Member Function Documentation	176
5.57.2.1	getRNGparameters() const	176
5.57.2.2	random()	176
5.57.2.3	reset()	177

5.57.2.4	<a href="#">sampleBeta(double alpha, double beta, double infLimit, double supLimit)</a>	177
5.57.2.5	<a href="#">sampleDiscrete(double value, double acumProb,...)</a>	177
5.57.2.6	<a href="#">sampleErlang(double mean, int M)</a>	177
5.57.2.7	<a href="#">sampleExponential(double mean)</a>	178
5.57.2.8	<a href="#">sampleGamma(double mean, double alpha)</a>	178
5.57.2.9	<a href="#">sampleLogNormal(double mean, double stddev)</a>	178
5.57.2.10	<a href="#">sampleNormal(double mean, double stddev)</a>	178
5.57.2.11	<a href="#">sampleTriangular(double min, double mode, double max)</a>	178
5.57.2.12	<a href="#">sampleUniform(double min, double max)</a>	179
5.57.2.13	<a href="#">sampleWeibull(double alpha, double scale)</a>	179
5.57.2.14	<a href="#">setRNGparameters(RNG_Parameters *param)</a>	179
5.58	<a href="#">SamplerDummyImpl Class Reference</a>	179
5.58.1	<a href="#">Constructor &amp; Destructor Documentation</a>	180
5.58.1.1	<a href="#">SamplerDummyImpl()</a>	180
5.58.1.2	<a href="#">SamplerDummyImpl(const SamplerDummyImpl &amp;orig)</a>	180
5.58.1.3	<a href="#">~SamplerDummyImpl()</a>	180
5.58.2	<a href="#">Member Function Documentation</a>	180
5.58.2.1	<a href="#">getRNGparameters() const</a>	180
5.58.2.2	<a href="#">random()</a>	181
5.58.2.3	<a href="#">sampleBeta(double alpha, double beta, double infLimit, double supLimit)</a>	181
5.58.2.4	<a href="#">sampleDiscrete(double value, double acumProb,...)</a>	181
5.58.2.5	<a href="#">sampleErlang(double mean, int M)</a>	181
5.58.2.6	<a href="#">sampleExponential(double mean)</a>	181
5.58.2.7	<a href="#">sampleGamma(double mean, double alpha)</a>	181
5.58.2.8	<a href="#">sampleLogNormal(double mean, double stddev)</a>	181
5.58.2.9	<a href="#">sampleNormal(double mean, double stddev)</a>	181
5.58.2.10	<a href="#">sampleTriangular(double min, double mode, double max)</a>	181
5.58.2.11	<a href="#">sampleUniform(double min, double max)</a>	181
5.58.2.12	<a href="#">sampleWeibull(double alpha, double scale)</a>	182
5.58.2.13	<a href="#">setRNGparameters(RNG_Parameters *param)</a>	182

5.59 ScenarioExperiment_if Class Reference . . . . .	182
5.60 Seize Class Reference . . . . .	182
5.60.1 Detailed Description . . . . .	184
5.60.2 Constructor & Destructor Documentation . . . . .	184
5.60.2.1 Seize(Model *model) . . . . .	184
5.60.2.2 Seize(const Seize &orig) . . . . .	184
5.60.2.3 ~Seize() . . . . .	184
5.60.3 Member Function Documentation . . . . .	184
5.60.3.1 _execute(Entity *entity) . . . . .	184
5.60.3.2 _loadInstance(std::list< std::string > words) . . . . .	185
5.60.3.3 _saveInstance() . . . . .	185
5.60.3.4 _verifySymbols(std::string *errorMessage) . . . . .	185
5.60.3.5 getAllocationType() const . . . . .	185
5.60.3.6 getLastMemberSeized() const . . . . .	185
5.60.3.7 getPriority() const . . . . .	185
5.60.3.8 getQuantity() const . . . . .	185
5.60.3.9 getQueue() const . . . . .	185
5.60.3.10 getQueueName() const . . . . .	185
5.60.3.11 getResource() const . . . . .	186
5.60.3.12 getResourceName() const . . . . .	186
5.60.3.13 getResourceType() const . . . . .	186
5.60.3.14 getRule() const . . . . .	186
5.60.3.15 getSaveAttribute() const . . . . .	186
5.60.3.16 setAllocationType(unsigned int _allocationType) . . . . .	186
5.60.3.17 setLastMemberSeized(unsigned int _lastMemberSeized) . . . . .	186
5.60.3.18 setPriority(unsigned short _priority) . . . . .	186
5.60.3.19 setQuantity(std::string _quantity) . . . . .	186
5.60.3.20 setQueue(Queue *queue) . . . . .	186
5.60.3.21 setQueueName(std::string queueName) . . . . .	187
5.60.3.22 setResource(Resource *resource) . . . . .	187

5.60.3.23	setResourceName(std::string _resourceName)	187
5.60.3.24	setResourceType(Resource::ResourceType _resourceType)	187
5.60.3.25	setRule(Resource::ResourceRule _rule)	187
5.60.3.26	setSaveAttribute(std::string _saveAttribute)	187
5.60.3.27	show()	187
5.61	SimulationControl Class Reference	188
5.61.1	Detailed Description	189
5.61.2	Constructor & Destructor Documentation	189
5.61.2.1	SimulationControl(std::string type, std::string name, GetterMember getter← Member, SetterMember setterMember)	189
5.61.2.2	SimulationControl(const SimulationControl &orig)	189
5.61.2.3	~SimulationControl()	189
5.61.3	Member Function Documentation	189
5.61.3.1	setValue(double value)	189
5.62	SimulationEvent Class Reference	189
5.62.1	Constructor & Destructor Documentation	190
5.62.1.1	SimulationEvent(unsigned int replicationNumber, Event *event)	190
5.62.2	Member Function Documentation	190
5.62.2.1	getEventProcessed() const	190
5.62.2.2	getReplicationNumber() const	190
5.63	SimulationResponse Class Reference	190
5.63.1	Detailed Description	191
5.63.2	Constructor & Destructor Documentation	191
5.63.2.1	SimulationResponse(std::string type, std::string name, GetterMember getter← Member)	191
5.63.2.2	SimulationResponse(const SimulationResponse &orig)	191
5.63.2.3	~SimulationResponse()	191
5.63.3	Member Function Documentation	191
5.63.3.1	getName() const	192
5.63.3.2	getType() const	192
5.63.3.3	getValue()	192

5.63.4	Member Data Documentation . . . . .	192
5.63.4.1	_getterMemberFunction . . . . .	192
5.63.4.2	_name . . . . .	192
5.63.4.3	_type . . . . .	192
5.64	SimulationScenario Class Reference . . . . .	192
5.64.1	Detailed Description . . . . .	192
5.64.2	Constructor & Destructor Documentation . . . . .	193
5.64.2.1	SimulationScenario() . . . . .	193
5.64.2.2	SimulationScenario(const SimulationScenario &orig) . . . . .	193
5.64.2.3	~SimulationScenario() . . . . .	193
5.64.3	Member Function Documentation . . . . .	193
5.64.3.1	getControlValue(SimulationControl *control) . . . . .	193
5.64.3.2	getControlValues() const . . . . .	193
5.64.3.3	getModelFilename() const . . . . .	193
5.64.3.4	getName() const . . . . .	193
5.64.3.5	getResponseValue(SimulationResponse *value) . . . . .	193
5.64.3.6	getResponseValues() const . . . . .	193
5.64.3.7	setControlValue(SimulationControl *control, double value) . . . . .	193
5.64.3.8	setModelFilename(std::string _modelName) . . . . .	193
5.64.3.9	setName(std::string _name) . . . . .	193
5.65	Simulator Class Reference . . . . .	193
5.65.1	Detailed Description . . . . .	194
5.65.2	Constructor & Destructor Documentation . . . . .	194
5.65.2.1	Simulator() . . . . .	194
5.65.2.2	Simulator(const Simulator &orig) . . . . .	194
5.65.2.3	~Simulator() . . . . .	194
5.65.3	Member Function Documentation . . . . .	194
5.65.3.1	getFitter() const . . . . .	194
5.65.3.2	getLicense() const . . . . .	195
5.65.3.3	getModels() const . . . . .	195



5.65.3.4	<a href="#">getName() const</a>	195
5.65.3.5	<a href="#">getPlugins() const</a>	195
5.65.3.6	<a href="#">getSampler() const</a>	195
5.65.3.7	<a href="#">getVersion() const</a>	196
5.66	<a href="#">SinkModelComponent Class Reference</a>	196
5.66.1	<a href="#">Detailed Description</a>	197
5.66.2	<a href="#">Constructor &amp; Destructor Documentation</a>	197
5.66.2.1	<a href="#">SinkModelComponent(Model *model)</a>	197
5.66.2.2	<a href="#">SinkModelComponent(const SinkModelComponent &amp;orig)</a>	197
5.66.2.3	<a href="#">~SinkModelComponent()</a>	197
5.66.3	<a href="#">Member Function Documentation</a>	197
5.66.3.1	<a href="#">isCollectStatistics() const</a>	197
5.66.3.2	<a href="#">setCollectStatistics(bool _collectStatistics)</a>	197
5.67	<a href="#">SourceModelComponent Class Reference</a>	198
5.67.1	<a href="#">Detailed Description</a>	199
5.67.2	<a href="#">Constructor &amp; Destructor Documentation</a>	200
5.67.2.1	<a href="#">SourceModelComponent(Model *model)</a>	200
5.67.2.2	<a href="#">SourceModelComponent(const SourceModelComponent &amp;orig)</a>	200
5.67.2.3	<a href="#">~SourceModelComponent()</a>	200
5.67.3	<a href="#">Member Function Documentation</a>	200
5.67.3.1	<a href="#">getEntitiesCreated() const</a>	200
5.67.3.2	<a href="#">getEntitiesPerCreation() const</a>	200
5.67.3.3	<a href="#">getEntityType() const</a>	200
5.67.3.4	<a href="#">getFirstCreation() const</a>	200
5.67.3.5	<a href="#">getMaxCreations() const</a>	200
5.67.3.6	<a href="#">getTimeBetweenCreationsExpression() const</a>	200
5.67.3.7	<a href="#">getTimeUnit() const</a>	200
5.67.3.8	<a href="#">isCollectStatistics() const</a>	200
5.67.3.9	<a href="#">setCollectStatistics(bool _collectStatistics)</a>	200
5.67.3.10	<a href="#">setEntitiesCreated(unsigned int _entitiesCreated)</a>	201

5.67.3.11	<a href="#">setEntitiesPerCreation(unsigned int _entitiesPerCreation)</a>	201
5.67.3.12	<a href="#">setEntityType(EntityType *_entityType)</a>	201
5.67.3.13	<a href="#">setFirstCreation(double _firstCreation)</a>	201
5.67.3.14	<a href="#">setMaxCreations(unsigned int _maxCreations)</a>	201
5.67.3.15	<a href="#">setTimeBetweenCreationsExpression(std::string _timeBetweenCreations)</a>	201
5.67.3.16	<a href="#">setTimeUnit(Util::TimeUnit _timeUnit)</a>	202
5.67.3.17	<a href="#">show()</a>	202
5.67.4	<a href="#">Member Data Documentation</a>	202
5.67.4.1	<a href="#">_collectStatistics</a>	202
5.67.4.2	<a href="#">_entitiesCreatedSoFar</a>	202
5.67.4.3	<a href="#">_entitiesPerCreation</a>	202
5.67.4.4	<a href="#">_entityType</a>	203
5.67.4.5	<a href="#">_firstCreation</a>	203
5.67.4.6	<a href="#">_maxCreations</a>	203
5.67.4.7	<a href="#">_timeBetweenCreationsExpression</a>	203
5.67.4.8	<a href="#">_timeBetweenCreationsTimeUnit</a>	203
5.68	<a href="#">Statistics_if Class Reference</a>	203
5.68.1	<a href="#">Detailed Description</a>	204
5.68.2	<a href="#">Member Function Documentation</a>	204
5.68.2.1	<a href="#">average()=0</a>	204
5.68.2.2	<a href="#">getCollector()=0</a>	204
5.68.2.3	<a href="#">halfWidthConfidenceInterval(double confidencelevel)=0</a>	204
5.68.2.4	<a href="#">max()=0</a>	205
5.68.2.5	<a href="#">min()=0</a>	205
5.68.2.6	<a href="#">newSampleSize(double confidencelevel, double halfWidth)=0</a>	205
5.68.2.7	<a href="#">numElements()=0</a>	205
5.68.2.8	<a href="#">setCollector(Collector_if *collector)=0</a>	206
5.68.2.9	<a href="#">stddeviation()=0</a>	206
5.68.2.10	<a href="#">variance()=0</a>	206
5.68.2.11	<a href="#">variationCoef()=0</a>	206

5.69 StatisticsCollector Class Reference . . . . .	207
5.69.1 Constructor & Destructor Documentation . . . . .	208
5.69.1.1 StatisticsCollector() . . . . .	208
5.69.1.2 StatisticsCollector(std::string name) . . . . .	208
5.69.1.3 StatisticsCollector(std::string name, ModelElement *parent) . . . . .	208
5.69.1.4 StatisticsCollector(const StatisticsCollector &orig) . . . . .	208
5.69.1.5 ~StatisticsCollector() . . . . .	208
5.69.2 Member Function Documentation . . . . .	208
5.69.2.1 _loadInstance(std::list< std::string > words) . . . . .	208
5.69.2.2 _saveInstance() . . . . .	208
5.69.2.3 _verifySymbols(std::string *errorMessage) . . . . .	208
5.69.2.4 getParent() const . . . . .	209
5.69.2.5 show() . . . . .	209
5.70 StatisticsDatafile_if Class Reference . . . . .	209
5.70.1 Member Function Documentation . . . . .	210
5.70.1.1 centil(unsigned short num)=0 . . . . .	210
5.70.1.2 decil(unsigned short num)=0 . . . . .	211
5.70.1.3 histogramClassFrequency(unsigned short classNum)=0 . . . . .	211
5.70.1.4 histogramClassLowerLimit(unsigned short classNum)=0 . . . . .	211
5.70.1.5 histogramNumClasses()=0 . . . . .	211
5.70.1.6 mediane()=0 . . . . .	211
5.70.1.7 mode()=0 . . . . .	211
5.70.1.8 quartil(unsigned short num)=0 . . . . .	211
5.70.1.9 setHistogramNumClasses(unsigned short num)=0 . . . . .	211
5.71 StatisticsDataFileDummyImpl Class Reference . . . . .	212
5.71.1 Constructor & Destructor Documentation . . . . .	213
5.71.1.1 StatisticsDataFileDummyImpl() . . . . .	213
5.71.1.2 StatisticsDataFileDummyImpl(const StatisticsDataFileDummyImpl &orig) . . . . .	213
5.71.1.3 ~StatisticsDataFileDummyImpl() . . . . .	213
5.71.2 Member Function Documentation . . . . .	213

5.71.2.1	<a href="#">average()</a>	213
5.71.2.2	<a href="#">centil(unsigned short num)</a>	213
5.71.2.3	<a href="#">decil(unsigned short num)</a>	213
5.71.2.4	<a href="#">getCollector()</a>	213
5.71.2.5	<a href="#">halfWidthConfidenceInterval(double confidencelevel)</a>	214
5.71.2.6	<a href="#">histogramClassFrequency(unsigned short classNum)</a>	214
5.71.2.7	<a href="#">histogramClassLowerLimit(unsigned short classNum)</a>	214
5.71.2.8	<a href="#">histogramNumClasses()</a>	214
5.71.2.9	<a href="#">max()</a>	214
5.71.2.10	<a href="#">mediane()</a>	214
5.71.2.11	<a href="#">min()</a>	214
5.71.2.12	<a href="#">mode()</a>	214
5.71.2.13	<a href="#">newSampleSize(double confidencelevel, double halfWidth)</a>	214
5.71.2.14	<a href="#">numElements()</a>	214
5.71.2.15	<a href="#">quartil(unsigned short num)</a>	215
5.71.2.16	<a href="#">setCollector(Collector_if *collector)</a>	215
5.71.2.17	<a href="#">setHistogramNumClasses(unsigned short num)</a>	215
5.71.2.18	<a href="#">stddeviation()</a>	215
5.71.2.19	<a href="#">variance()</a>	215
5.71.2.20	<a href="#">variationCoef()</a>	215
5.72	<a href="#">StatisticsDefaultImpl1 Class Reference</a>	215
5.72.1	<a href="#">Constructor &amp; Destructor Documentation</a>	216
5.72.1.1	<a href="#">StatisticsDefaultImpl1()</a>	217
5.72.1.2	<a href="#">StatisticsDefaultImpl1(const StatisticsDefaultImpl1 &amp;orig)</a>	217
5.72.1.3	<a href="#">~StatisticsDefaultImpl1()</a>	217
5.72.2	<a href="#">Member Function Documentation</a>	217
5.72.2.1	<a href="#">average()</a>	217
5.72.2.2	<a href="#">getCollector()</a>	218
5.72.2.3	<a href="#">halfWidthConfidenceInterval(double confidencelevel)</a>	218
5.72.2.4	<a href="#">max()</a>	218

5.72.2.5	<code>min()</code>	218
5.72.2.6	<code>newSampleSize(double confidencelevel, double halfWidth)</code>	218
5.72.2.7	<code>numElements()</code>	218
5.72.2.8	<code>setCollector(Collector_if *collector)</code>	219
5.72.2.9	<code>stddeviation()</code>	219
5.72.2.10	<code>variance()</code>	219
5.72.2.11	<code>variationCoef()</code>	219
5.73	StatisticsDummyImpl Class Reference	219
5.73.1	Constructor & Destructor Documentation	220
5.73.1.1	<code>StatisticsDummyImpl()</code>	221
5.73.1.2	<code>StatisticsDummyImpl(const StatisticsDummyImpl &amp;orig)</code>	221
5.73.1.3	<code>~StatisticsDummyImpl()</code>	221
5.73.2	Member Function Documentation	221
5.73.2.1	<code>average()</code>	221
5.73.2.2	<code>getCollector()</code>	221
5.73.2.3	<code>halfWidthConfidenceInterval(double confidencelevel)</code>	222
5.73.2.4	<code>max()</code>	222
5.73.2.5	<code>min()</code>	222
5.73.2.6	<code>newSampleSize(double confidencelevel, double halfWidth)</code>	223
5.73.2.7	<code>numElements()</code>	223
5.73.2.8	<code>setCollector(Collector_if *collector)</code>	223
5.73.2.9	<code>stddeviation()</code>	223
5.73.2.10	<code>variance()</code>	223
5.73.2.11	<code>variationCoef()</code>	223
5.74	TestInputAnalyserTools Class Reference	224
5.74.1	Constructor & Destructor Documentation	224
5.74.1.1	<code>TestInputAnalyserTools()</code>	224
5.74.2	Member Function Documentation	224
5.74.2.1	<code>main(int argc, char **argv)</code>	224
5.75	TestParser Class Reference	226

5.75.1	Constructor & Destructor Documentation . . . . .	226
5.75.1.1	TestParser() . . . . .	226
5.75.1.2	TestParser(const TestParser &orig) . . . . .	226
5.75.1.3	~TestParser() . . . . .	227
5.75.2	Member Function Documentation . . . . .	227
5.75.2.1	main(int argc, char **argv) . . . . .	227
5.76	TestStatistics Class Reference . . . . .	227
5.76.1	Constructor & Destructor Documentation . . . . .	228
5.76.1.1	TestStatistics() . . . . .	228
5.76.2	Member Function Documentation . . . . .	228
5.76.2.1	main(int argc, char **argv) . . . . .	228
5.77	TraceErrorEvent Class Reference . . . . .	229
5.77.1	Constructor & Destructor Documentation . . . . .	230
5.77.1.1	TraceErrorEvent(std::string text, std::exception e) . . . . .	230
5.77.2	Member Function Documentation . . . . .	230
5.77.2.1	getException() const . . . . .	230
5.78	TraceEvent Class Reference . . . . .	231
5.78.1	Constructor & Destructor Documentation . . . . .	231
5.78.1.1	TraceEvent(Util::TraceLevel tracelevel, std::string text) . . . . .	231
5.78.2	Member Function Documentation . . . . .	231
5.78.2.1	getText() const . . . . .	231
5.78.2.2	getTracelevel() const . . . . .	231
5.79	TraceManager Class Reference . . . . .	232
5.79.1	Detailed Description . . . . .	232
5.79.2	Constructor & Destructor Documentation . . . . .	232
5.79.2.1	TraceManager(Model *model) . . . . .	232
5.79.2.2	TraceManager(const TraceManager &orig) . . . . .	232
5.79.2.3	~TraceManager() . . . . .	232
5.79.3	Member Function Documentation . . . . .	232
5.79.3.1	addTraceErrorHandler(traceErrorListener traceErrorListener) . . . . .	232

5.79.3.2	<a href="#">addTraceHandler(traceListener traceListener)</a>	232
5.79.3.3	<a href="#">addTraceReportHandler(traceListener traceReportListener)</a>	233
5.79.3.4	<a href="#">addTraceSimulationHandler(traceSimulationListener traceSimulationListener)</a>	233
5.79.3.5	<a href="#">getErrorMessage() const</a>	233
5.79.3.6	<a href="#">getTraceLevel() const</a>	233
5.79.3.7	<a href="#">setTraceLevel(Util::TraceLevel _traceLevel)</a>	233
5.79.3.8	<a href="#">trace(Util::TraceLevel tracelevel, std::string text)</a>	234
5.79.3.9	<a href="#">traceError(std::exception e, std::string text)</a>	234
5.79.3.10	<a href="#">traceReport(Util::TraceLevel tracelevel, std::string text)</a>	235
5.79.3.11	<a href="#">traceSimulation(Util::TraceLevel tracelevel, double time, Entity *entity, ModelComponent *component, std::string text)</a>	235
5.80	<a href="#">TraceSimulationEvent Class Reference</a>	236
5.80.1	<a href="#">Constructor &amp; Destructor Documentation</a>	237
5.80.1.1	<a href="#">TraceSimulationEvent(Util::TraceLevel tracelevel, double time, Entity *entity, ModelComponent *component, std::string text)</a>	237
5.80.2	<a href="#">Member Function Documentation</a>	237
5.80.2.1	<a href="#">getComponent() const</a>	237
5.80.2.2	<a href="#">getEntity() const</a>	237
5.80.2.3	<a href="#">getTime() const</a>	237
5.81	<a href="#">TraceSimulationProcess Class Reference</a>	237
5.81.1	<a href="#">Detailed Description</a>	238
5.81.2	<a href="#">Constructor &amp; Destructor Documentation</a>	238
5.81.2.1	<a href="#">TraceSimulationProcess(Util::TraceLevel tracelevel, std::string text)</a>	238
5.82	<a href="#">Traits&lt; T &gt; Struct Template Reference</a>	238
5.83	<a href="#">Traits&lt; Collector_if &gt; Struct Template Reference</a>	238
5.83.1	<a href="#">Member Typedef Documentation</a>	238
5.83.1.1	<a href="#">Implementation</a>	238
5.84	<a href="#">Traits&lt; ExperimentDesign_if &gt; Struct Template Reference</a>	239
5.84.1	<a href="#">Member Typedef Documentation</a>	239
5.84.1.1	<a href="#">Implementation</a>	239
5.85	<a href="#">Traits&lt; Fitter_if &gt; Struct Template Reference</a>	239

5.85.1	Member Typedef Documentation . . . . .	239
5.85.1.1	Implementation . . . . .	239
5.86	Traits< GenesysApplication_if > Struct Template Reference . . . . .	239
5.86.1	Member Typedef Documentation . . . . .	240
5.86.1.1	Application . . . . .	240
5.87	Traits< HypothesisTester_if > Struct Template Reference . . . . .	240
5.87.1	Member Typedef Documentation . . . . .	240
5.87.1.1	Implementation . . . . .	240
5.88	Traits< Integrator_if > Struct Template Reference . . . . .	240
5.88.1	Member Typedef Documentation . . . . .	241
5.88.1.1	Implementation . . . . .	241
5.88.2	Member Data Documentation . . . . .	241
5.88.2.1	MaxIterations . . . . .	241
5.88.2.2	Precision . . . . .	241
5.89	Traits< Model > Struct Template Reference . . . . .	241
5.89.1	Member Data Documentation . . . . .	241
5.89.1.1	debugged . . . . .	241
5.89.1.2	traceLevel . . . . .	241
5.90	Traits< ModelChecker_if > Struct Template Reference . . . . .	241
5.90.1	Member Typedef Documentation . . . . .	242
5.90.1.1	Implementation . . . . .	242
5.91	Traits< ModelComponent > Struct Template Reference . . . . .	242
5.91.1	Member Typedef Documentation . . . . .	242
5.91.1.1	CollectorImplementation . . . . .	242
5.91.1.2	StatisticsCollectorImplementation . . . . .	242
5.92	Traits< ModelPersistence_if > Struct Template Reference . . . . .	242
5.92.1	Member Typedef Documentation . . . . .	242
5.92.1.1	Implementation . . . . .	242
5.93	Traits< Parser_if > Struct Template Reference . . . . .	243
5.93.1	Member Typedef Documentation . . . . .	243



5.93.1.1	Implementation . . . . .	243
5.94	Traits< ProcessAnalyser_if > Struct Template Reference . . . . .	243
5.94.1	Member Typedef Documentation . . . . .	243
5.94.1.1	Implementation . . . . .	243
5.95	Traits< Sampler_if > Struct Template Reference . . . . .	243
5.95.1	Member Typedef Documentation . . . . .	244
5.95.1.1	Implementation . . . . .	244
5.95.1.2	Parameters . . . . .	244
5.96	Traits< Statistics_if > Struct Template Reference . . . . .	244
5.96.1	Member Typedef Documentation . . . . .	244
5.96.1.1	CollectorImplementation . . . . .	244
5.96.1.2	Implementation . . . . .	244
5.97	Util Class Reference . . . . .	244
5.97.1	Member Typedef Documentation . . . . .	245
5.97.1.1	identitification . . . . .	245
5.97.1.2	rank . . . . .	245
5.97.2	Member Enumeration Documentation . . . . .	245
5.97.2.1	TimeUnit . . . . .	245
5.97.2.2	TraceLevel . . . . .	245
5.97.3	Member Function Documentation . . . . .	246
5.97.3.1	DecIndent() . . . . .	246
5.97.3.2	GenerateNewId() . . . . .	246
5.97.3.3	GenerateNewIdOfType(std::string objtyp) . . . . .	246
5.97.3.4	GenerateNewIdOfType() . . . . .	246
5.97.3.5	InclIndent() . . . . .	247
5.97.3.6	Indent() . . . . .	247
5.97.3.7	TimeUnitConvert(Util::TimeUnit timeUnit1, Util::TimeUnit timeUnit2) . . . . .	248
5.97.3.8	TypeOf() . . . . .	248
5.98	Variable Class Reference . . . . .	248
5.98.1	Constructor & Destructor Documentation . . . . .	249

5.98.1.1	Variable()	249
5.98.1.2	Variable(std::string name)	249
5.98.1.3	Variable(const Variable &orig)	249
5.98.1.4	~Variable()	249
5.98.2	Member Function Documentation	249
5.98.2.1	_loadInstance(std::list< std::string > words)	249
5.98.2.2	_saveInstance()	250
5.98.2.3	_verifySymbols(std::string *errorMessage)	250
5.98.2.4	getValue()	250
5.98.2.5	getValue(std::string index)	250
5.98.2.6	setValue(double value)	250
5.98.2.7	setValue(std::string index, double value)	250
5.98.2.8	show()	250
5.99	Waiting Class Reference	251
5.99.1	Constructor & Destructor Documentation	252
5.99.1.1	Waiting(Entity *entity, ModelComponent *component, double timeStartedWaiting)	252
5.99.1.2	Waiting(const Waiting &orig)	252
5.99.1.3	~Waiting()	252
5.99.2	Member Function Documentation	252
5.99.2.1	getComponent() const	252
5.99.2.2	getEntity() const	252
5.99.2.3	getTimeStartedWaiting() const	252
5.99.2.4	show()	253
5.100	WaitingResource Class Reference	253
5.100.1	Constructor & Destructor Documentation	254
5.100.1.1	WaitingResource(Entity *entity, ModelComponent *component, double timeStartedWaiting, unsigned int quantity)	254
5.100.1.2	WaitingResource(const WaitingResource &orig)	254
5.100.1.3	~WaitingResource()	254
5.100.2	Member Function Documentation	254
5.100.2.1	getQuantity() const	254
5.100.2.2	show()	255

<b>6 File Documentation</b>	<b>257</b>
6.1 .dep.inc File Reference	257
6.2 Assign.cpp File Reference	257
6.3 Assign.h File Reference	257
6.4 Attribute.cpp File Reference	258
6.5 Attribute.h File Reference	259
6.6 BuildSimulationModel.cpp File Reference	260
6.6.1 Function Documentation	261
6.6.1.1 buildModel(Model *model)	261
6.6.1.2 buildSimulationSystem()	262
6.6.1.3 buildVerySimplifiedModel(Model *model)	265
6.6.1.4 onEntityRemoveHandler(SimulationEvent *re)	266
6.6.1.5 onProcessEventHandler(SimulationEvent *re)	266
6.6.1.6 onReplicationEndHandler(SimulationEvent *re)	267
6.6.1.7 onReplicationStartHandler(SimulationEvent *re)	267
6.6.1.8 onSimulationStartHandler(SimulationEvent *re)	268
6.6.1.9 traceHandler(TraceEvent e)	268
6.6.1.10 traceSimulationHandler(TraceSimulationEvent e)	268
6.7 BuildSimulationModel.h File Reference	269
6.8 Collector_if.h File Reference	269
6.8.1 Typedef Documentation	271
6.8.1.1 CollectorAddValueHandler	271
6.8.1.2 CollectorClearHandler	271
6.8.2 Function Documentation	271
6.8.2.1 SetCollectorAddValueHandler(void(Class::*function)(double), Class *object)	271
6.8.2.2 SetCollectorClearHandler(void(Class::*function)(), Class *object)	271
6.9 CollectorDatafile_if.h File Reference	272
6.10 CollectorDatafileDefaultImpl1.cpp File Reference	272
6.11 CollectorDatafileDefaultImpl1.h File Reference	273
6.12 CollectorDatafileDummyImpl.cpp File Reference	274

6.13	CollectorDatafileDummyImpl.h File Reference	275
6.14	CollectorDefaultImpl1.cpp File Reference	276
6.15	CollectorDefaultImpl1.h File Reference	277
6.16	CollectorDummyImpl.cpp File Reference	278
6.17	CollectorDummyImpl.h File Reference	278
6.18	Create.cpp File Reference	279
6.19	Create.h File Reference	280
6.20	DefineGetterSetter.h File Reference	281
6.20.1	Typedef Documentation	282
6.20.1.1	GetterMember	282
6.20.1.2	SetterMember	282
6.20.2	Function Documentation	282
6.20.2.1	DefineGetterMember(Class *object, double(Class::*function)())	282
6.20.2.2	DefineGetterMember(Class *object, unsigned int(Class::*function)() const)	282
6.20.2.3	DefineGetterMember(Class *object, bool(Class::*function)() const)	282
6.20.2.4	DefineGetterMember(Class *object, std::string(Class::*function)() const)	282
6.20.2.5	DefineGetterMember(Class *object, Util::TimeUnit(Class::*function)() const)	282
6.20.2.6	DefineSetterMember(Class *object, void(Class::*function)(double))	283
6.20.2.7	DefineSetterMember(Class *object, void(Class::*function)(unsigned int))	283
6.20.2.8	DefineSetterMember(Class *object, void(Class::*function)(bool))	283
6.20.2.9	DefineSetterMember(Class *object, void(Class::*function)(std::string))	283
6.20.2.10	DefineSetterMember(Class *object, void(Class::*function)(Util::TimeUnit))	283
6.21	Delay.cpp File Reference	283
6.22	Delay.h File Reference	283
6.23	Dispose.cpp File Reference	284
6.24	Dispose.h File Reference	285
6.25	ElementManager.cpp File Reference	286
6.26	ElementManager.h File Reference	286
6.27	ElementManager_if.h File Reference	287
6.28	Entity.cpp File Reference	287

6.29 Entity.h File Reference . . . . .	288
6.30 EntityType.cpp File Reference . . . . .	289
6.31 EntityType.h File Reference . . . . .	289
6.32 Event.cpp File Reference . . . . .	290
6.33 Event.h File Reference . . . . .	290
6.34 ExperimentDesign_if.h File Reference . . . . .	291
6.35 ExperimentDesignDummyImpl.cpp File Reference . . . . .	292
6.36 ExperimentDesignDummyImpl.h File Reference . . . . .	293
6.37 FactorOrInteractionContribution.cpp File Reference . . . . .	294
6.38 FactorOrInteractionContribution.h File Reference . . . . .	294
6.39 Fitter_if.h File Reference . . . . .	295
6.40 FitterDummyImpl.cpp File Reference . . . . .	296
6.41 FitterDummyImpl.h File Reference . . . . .	297
6.42 Functor.h File Reference . . . . .	298
6.43 GenesysApplication_if.h File Reference . . . . .	298
6.44 HypothesisTester_if.h File Reference . . . . .	299
6.45 HypothesisTesterDummyImpl.cpp File Reference . . . . .	299
6.46 HypothesisTesterDummyImpl.h File Reference . . . . .	300
6.47 Integrator_if.h File Reference . . . . .	301
6.48 IntegratorDefaultImpl1.cpp File Reference . . . . .	301
6.49 IntegratorDefaultImpl1.h File Reference . . . . .	302
6.50 IntegratorDummyImpl.cpp File Reference . . . . .	302
6.51 IntegratorDummyImpl.h File Reference . . . . .	303
6.52 LinkedBy.cpp File Reference . . . . .	303
6.53 LinkedBy.h File Reference . . . . .	304
6.54 List.h File Reference . . . . .	304
6.55 main.cpp File Reference . . . . .	305
6.55.1 Function Documentation . . . . .	306
6.55.1.1 main(int argc, char **argv) . . . . .	306
6.56 Model.cpp File Reference . . . . .	306

6.56.1	Function Documentation	306
6.56.1.1	EventCompare(const Event *a, const Event *b)	307
6.56.1.2	getReplicationLengthNotMemberFunction()	307
6.56.1.3	setReplicationLengthNotMemberFunction(double value)	307
6.57	Model.h File Reference	307
6.58	ModelChecker_if.h File Reference	308
6.59	ModelCheckerDummyImpl.cpp File Reference	309
6.60	ModelCheckerDummyImpl.h File Reference	309
6.61	ModelComponent.cpp File Reference	310
6.62	ModelComponent.h File Reference	310
6.63	ModelComponentManager_if.h File Reference	311
6.64	ModelElement.cpp File Reference	311
6.65	ModelElement.h File Reference	312
6.66	ModelInfo.cpp File Reference	312
6.67	ModelInfo.h File Reference	313
6.68	ModelPersistence_if.h File Reference	314
6.69	ModelPersistenceDummyImpl.cpp File Reference	314
6.70	ModelPersistenceDummyImpl.h File Reference	315
6.71	ModelSimulation.cpp File Reference	316
6.72	ModelSimulation.h File Reference	316
6.73	OnEventManager.cpp File Reference	317
6.74	OnEventManager.h File Reference	317
6.74.1	Typedef Documentation	318
6.74.1.1	simulationEventHandler	318
6.75	Parser_if.h File Reference	319
6.76	ParserDefaultImpl1.cpp File Reference	319
6.77	ParserDefaultImpl1.h File Reference	320
6.78	ParserDummyImpl.cpp File Reference	321
6.79	ParserDummyImpl.h File Reference	322
6.80	Plugin.cpp File Reference	322

6.81 Plugin.h File Reference . . . . .	323
6.82 ProbDistrib.cpp File Reference . . . . .	324
6.83 ProbDistrib.h File Reference . . . . .	324
6.84 ProcessAnalyser_if.h File Reference . . . . .	325
6.85 ProcessAnalyserDummyImpl.cpp File Reference . . . . .	326
6.86 ProcessAnalyserDummyImpl.h File Reference . . . . .	326
6.87 Queue.cpp File Reference . . . . .	327
6.88 Queue.h File Reference . . . . .	328
6.89 README.md File Reference . . . . .	329
6.90 Release.cpp File Reference . . . . .	329
6.91 Release.h File Reference . . . . .	330
6.92 Resource.cpp File Reference . . . . .	331
6.93 Resource.h File Reference . . . . .	331
6.94 Sampler_if.h File Reference . . . . .	332
6.95 SamplerDefaultImpl1.cpp File Reference . . . . .	332
6.96 SamplerDefaultImpl1.h File Reference . . . . .	333
6.97 SamplerDummyImpl.cpp File Reference . . . . .	334
6.98 SamplerDummyImpl.h File Reference . . . . .	334
6.99 ScenarioExperiment_if.h File Reference . . . . .	335
6.100Seize.cpp File Reference . . . . .	335
6.101Seize.h File Reference . . . . .	336
6.102SimulationControl.cpp File Reference . . . . .	336
6.103SimulationControl.h File Reference . . . . .	337
6.104SimulationResponse.cpp File Reference . . . . .	338
6.105SimulationResponse.h File Reference . . . . .	339
6.106SimulationScenario.cpp File Reference . . . . .	340
6.107SimulationScenario.h File Reference . . . . .	341
6.108Simulator.cpp File Reference . . . . .	342
6.109Simulator.h File Reference . . . . .	343
6.110SinkModelComponent.cpp File Reference . . . . .	344

6.111 SinkModelComponent.h File Reference . . . . .	344
6.112 SourceModelComponent.cpp File Reference . . . . .	345
6.113 SourceModelComponent.h File Reference . . . . .	345
6.114 Statistics_if.h File Reference . . . . .	346
6.115 StatisticsCollector.cpp File Reference . . . . .	347
6.116 StatisticsCollector.h File Reference . . . . .	348
6.117 StatisticsDataFile_if.h File Reference . . . . .	349
6.118 StatisticsDataFileDummyImpl.cpp File Reference . . . . .	350
6.119 StatisticsDataFileDummyImpl.h File Reference . . . . .	350
6.120 StatisticsDefaultImpl1.cpp File Reference . . . . .	352
6.121 StatisticsDefaultImpl1.h File Reference . . . . .	352
6.122 StatisticsDummyImpl.cpp File Reference . . . . .	353
6.123 StatisticsDummyImpl.h File Reference . . . . .	353
6.124 TestInputAnalyserTools.cpp File Reference . . . . .	354
6.124.1 Function Documentation . . . . .	354
6.124.1.1 testStudentSoftwareDevelopments() . . . . .	355
6.125 TestInputAnalyserTools.h File Reference . . . . .	356
6.126 TestParser.cpp File Reference . . . . .	356
6.127 TestParser.h File Reference . . . . .	357
6.128 TestStatistics.cpp File Reference . . . . .	358
6.129 TestStatistics.h File Reference . . . . .	358
6.130 TraceManager.cpp File Reference . . . . .	359
6.131 TraceManager.h File Reference . . . . .	359
6.131.1 Typedef Documentation . . . . .	360
6.131.1.1 traceErrorListener . . . . .	360
6.131.1.2 traceListener . . . . .	360
6.131.1.3 traceSimulationListener . . . . .	360
6.131.1.4 traceSimulationProcessListener . . . . .	360
6.132 Traits.h File Reference . . . . .	360
6.133 Util.cpp File Reference . . . . .	361
6.134 Util.h File Reference . . . . .	362
6.135 Variable.cpp File Reference . . . . .	362
6.136 Variable.h File Reference . . . . .	363
6.137 Waiting.cpp File Reference . . . . .	364
6.138 Waiting.h File Reference . . . . .	365
6.139 WaitingResource.cpp File Reference . . . . .	366
6.140 WaitingResource.h File Reference . . . . .	367



# Chapter 1

## GenESyS-Reborn

Generic and Expansible System [Simulator](#)

(Work in progress C++ port from the original in Pascal)

Developed by [rlcancian](#)



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

Assign::Assignment . . . . .	20
Collector_if . . . . .	27
CollectorDatafile_if . . . . .	29
CollectorDatafileDefaultImpl1 . . . . .	32
CollectorDatafileDummyImpl . . . . .	35
CollectorDefaultImpl1 . . . . .	38
CollectorDummyImpl . . . . .	40
ElementManager . . . . .	54
ElementManager_if . . . . .	58
Event . . . . .	66
ExperimentDesign_if . . . . .	68
ExperimentDesignDummyImpl . . . . .	69
FactorOrInteractionContribution . . . . .	70
Fitter_if . . . . .	71
FitterDummyImpl . . . . .	74
GenesysApplication_if . . . . .	77
BuildSimulationModel . . . . .	25
TestInputAnalyserTools . . . . .	224
TestParser . . . . .	226
TestStatistics . . . . .	227
HypothesisTester_if . . . . .	78
HypothesisTesterDummyImpl . . . . .	80
Integrator_if . . . . .	83
IntegratorDefaultImpl1 . . . . .	84
IntegratorDummyImpl . . . . .	86
LinkedBy . . . . .	88
Queue . . . . .	154
Resource . . . . .	165
List< T > . . . . .	89
List< Assign::Assignment * > . . . . .	89
List< double > . . . . .	89
List< Event * > . . . . .	89
List< Model * > . . . . .	89

List< ModelComponent * > . . . . .	89
List< Plugin * > . . . . .	89
List< SimulationControl * > . . . . .	89
List< SimulationResponse * > . . . . .	89
List< std::string > . . . . .	89
List< Waiting * > . . . . .	89
Model . . . . .	95
ModelChecker_if . . . . .	105
ModelCheckerDummyImpl . . . . .	107
ModelComponentManager_if . . . . .	117
ModelElement . . . . .	118
Attribute . . . . .	22
Entity . . . . .	59
EntityType . . . . .	62
ModelComponent . . . . .	111
Assign . . . . .	15
Delay . . . . .	47
Release . . . . .	161
Seize . . . . .	182
SinkModelComponent . . . . .	196
Dispose . . . . .	51
SourceModelComponent . . . . .	198
Create . . . . .	42
Queue . . . . .	154
Resource . . . . .	165
StatisticsCollector . . . . .	207
Variable . . . . .	248
ModelInfo . . . . .	123
ModelPersistence_if . . . . .	128
ModelPersistenceDummyImpl . . . . .	130
ModelSimulation . . . . .	134
OnEventManager . . . . .	138
Parser_if . . . . .	142
ParserDefaultImpl1 . . . . .	143
ParserDummyImpl . . . . .	145
Plugin . . . . .	147
ProbDistrib . . . . .	148
ProcessAnalyser_if . . . . .	150
ProcessAnalyserDummyImpl . . . . .	152
Sampler_if::RNG_Parameters . . . . .	171
SamplerDefaultImpl1::DefaultImpl1RNG_Parameters . . . . .	46
SamplerDummyImpl::MyRNG_Parameters . . . . .	137
Sampler_if . . . . .	171
SamplerDefaultImpl1 . . . . .	175
SamplerDummyImpl . . . . .	179
ScenarioExperiment_if . . . . .	182
SimulationEvent . . . . .	189
SimulationResponse . . . . .	190
SimulationControl . . . . .	188
SimulationScenario . . . . .	192
Simulator . . . . .	193
Statistics_if . . . . .	203
StatisticsDatafile_if . . . . .	209
StatisticsDataFileDummyImpl . . . . .	212
StatisticsDefaultImpl1 . . . . .	215

StatisticsDummyImpl . . . . .	219
StatisticsCollector . . . . .	207
TraceEvent . . . . .	231
TraceErrorEvent . . . . .	229
TraceSimulationEvent . . . . .	236
TraceSimulationProcess . . . . .	237
TraceManager . . . . .	232
Traits< T > . . . . .	238
Traits< Collector_if > . . . . .	238
Traits< ExperimentDesign_if > . . . . .	239
Traits< Fitter_if > . . . . .	239
Traits< GenesysApplication_if > . . . . .	239
Traits< HypothesisTester_if > . . . . .	240
Traits< Integrator_if > . . . . .	240
Traits< Model > . . . . .	241
Traits< ModelChecker_if > . . . . .	241
Traits< ModelComponent > . . . . .	242
Traits< ModelPersistence_if > . . . . .	242
Traits< Parser_if > . . . . .	243
Traits< ProcessAnalyser_if > . . . . .	243
Traits< Sampler_if > . . . . .	243
Traits< Statistics_if > . . . . .	244
Util . . . . .	244
Waiting . . . . .	251
WaitingResource . . . . .	253



## Chapter 3

# Class Index

### 3.1 Class List

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

Assign	15
Assign::Assignment	20
Attribute	22
BuildSimulationModel	25
Collector_if	27
CollectorDatafile_if	29
CollectorDatafileDefaultImpl1	32
CollectorDatafileDummyImpl	35
CollectorDefaultImpl1	38
CollectorDummyImpl	40
Create	42
SamplerDefaultImpl1::DefaultImpl1RNG_Parameters	46
Delay	47
Dispose	51
ElementManager	54
ElementManager_if	58
Entity	59
EntityType	62
Event	66
ExperimentDesign_if	68
ExperimentDesignDummyImpl	69
FactorOrInteractionContribution	70
Fitter_if	71
FitterDummyImpl	74
GenesysApplication_if	77
HypothesisTester_if	78
HypothesisTesterDummyImpl	80
Integrator_if	83
IntegratorDefaultImpl1	84
IntegratorDummyImpl	86
LinkedBy	88
List< T >	89
Model	95
ModelChecker_if	105
ModelCheckerDummyImpl	107

ModelComponent	111
ModelComponentManager_if	117
ModelElement	118
ModelInfo	123
ModelPersistence_if	128
ModelPersistenceDummyImpl	130
ModelSimulation	134
SamplerDummyImpl::MyRNG_Parameters	137
OnEventManager	138
Parser_if	142
ParserDefaultImpl1	143
ParserDummyImpl	145
Plugin	147
ProbDistrib	148
ProcessAnalyser_if	150
ProcessAnalyserDummyImpl	152
Queue	154
Release	161
Resource	165
Sampler_if::RNG_Parameters	171
Sampler_if	171
SamplerDefaultImpl1	175
SamplerDummyImpl	179
ScenarioExperiment_if	182
Seize	182
SimulationControl	188
SimulationEvent	189
SimulationResponse	190
SimulationScenario	192
Simulator	193
SinkModelComponent	196
SourceModelComponent	198
Statistics_if	203
StatisticsCollector	207
StatisticsDatafile_if	209
StatisticsDataFileDummyImpl	212
StatisticsDefaultImpl1	215
StatisticsDummyImpl	219
TestInputAnalyserTools	224
TestParser	226
TestStatistics	227
TraceErrorEvent	229
TraceEvent	231
TraceManager	232
TraceSimulationEvent	236
TraceSimulationProcess	237
Traits< T >	238
Traits< Collector_if >	238
Traits< ExperimentDesign_if >	239
Traits< Fitter_if >	239
Traits< GenesysApplication_if >	239
Traits< HypothesisTester_if >	240
Traits< Integrator_if >	240
Traits< Model >	241
Traits< ModelChecker_if >	241
Traits< ModelComponent >	242
Traits< ModelPersistence_if >	242
Traits< Parser_if >	243



<a href="#">Traits&lt; ProcessAnalyser_if &gt;</a>	243
<a href="#">Traits&lt; Sampler_if &gt;</a>	243
<a href="#">Traits&lt; Statistics_if &gt;</a>	244
<a href="#">Util</a>	244
<a href="#">Variable</a>	248
<a href="#">Waiting</a>	251
<a href="#">WaitingResource</a>	253



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

.dep.inc	257
Assign.cpp	257
Assign.h	257
Attribute.cpp	258
Attribute.h	259
BuildSimulationModel.cpp	260
BuildSimulationModel.h	269
Collector_if.h	269
CollectorDatafile_if.h	272
CollectorDatafileDefaultImpl1.cpp	272
CollectorDatafileDefaultImpl1.h	273
CollectorDatafileDummyImpl.cpp	274
CollectorDatafileDummyImpl.h	275
CollectorDefaultImpl1.cpp	276
CollectorDefaultImpl1.h	277
CollectorDummyImpl.cpp	278
CollectorDummyImpl.h	278
Create.cpp	279
Create.h	280
DefineGetterSetter.h	281
Delay.cpp	283
Delay.h	283
Dispose.cpp	284
Dispose.h	285
ElementManager.cpp	286
ElementManager.h	286
ElementManager_if.h	287
Entity.cpp	287
Entity.h	288
EntityType.cpp	289
EntityType.h	289
Event.cpp	290
Event.h	290
ExperimentDesign_if.h	291
ExperimentDesignDummyImpl.cpp	292

ExperimentDesignDummyImpl.h	293
FactorOrInteractionContribution.cpp	294
FactorOrInteractionContribution.h	294
Fitter_if.h	295
FitterDummyImpl.cpp	296
FitterDummyImpl.h	297
Functor.h	298
GenesysApplication_if.h	298
HypothesisTester_if.h	299
HypothesisTesterDummyImpl.cpp	299
HypothesisTesterDummyImpl.h	300
Integrator_if.h	301
IntegratorDefaultImpl1.cpp	301
IntegratorDefaultImpl1.h	302
IntegratorDummyImpl.cpp	302
IntegratorDummyImpl.h	303
LinkedBy.cpp	303
LinkedBy.h	304
List.h	304
main.cpp	305
Model.cpp	306
Model.h	307
ModelChecker_if.h	308
ModelCheckerDummyImpl.cpp	309
ModelCheckerDummyImpl.h	309
ModelComponent.cpp	310
ModelComponent.h	310
ModelComponentManager_if.h	311
ModelElement.cpp	311
ModelElement.h	312
ModelInfo.cpp	312
ModelInfo.h	313
ModelPersistence_if.h	314
ModelPersistenceDummyImpl.cpp	314
ModelPersistenceDummyImpl.h	315
ModelSimulation.cpp	316
ModelSimulation.h	316
OnEventManager.cpp	317
OnEventManager.h	317
Parser_if.h	319
ParserDefaultImpl1.cpp	319
ParserDefaultImpl1.h	320
ParserDummyImpl.cpp	321
ParserDummyImpl.h	322
Plugin.cpp	322
Plugin.h	323
ProbDistrib.cpp	324
ProbDistrib.h	324
ProcessAnalyser_if.h	325
ProcessAnalyserDummyImpl.cpp	326
ProcessAnalyserDummyImpl.h	326
Queue.cpp	327
Queue.h	328
Release.cpp	329
Release.h	330
Resource.cpp	331
Resource.h	331
Sampler_if.h	332

SamplerDefaultImpl1.cpp	332
SamplerDefaultImpl1.h	333
SamplerDummyImpl.cpp	334
SamplerDummyImpl.h	334
ScenarioExperiment_if.h	335
Seize.cpp	335
Seize.h	336
SimulationControl.cpp	336
SimulationControl.h	337
SimulationResponse.cpp	338
SimulationResponse.h	339
SimulationScenario.cpp	340
SimulationScenario.h	341
Simulator.cpp	342
Simulator.h	343
SinkModelComponent.cpp	344
SinkModelComponent.h	344
SourceModelComponent.cpp	345
SourceModelComponent.h	345
Statistics_if.h	346
StatisticsCollector.cpp	347
StatisticsCollector.h	348
StatisticsDataFile_if.h	349
StatisticsDataFileDummyImpl.cpp	350
StatisticsDataFileDummyImpl.h	350
StatisticsDefaultImpl1.cpp	352
StatisticsDefaultImpl1.h	352
StatisticsDummyImpl.cpp	353
StatisticsDummyImpl.h	353
TestInputAnalyserTools.cpp	354
TestInputAnalyserTools.h	356
TestParser.cpp	356
TestParser.h	357
TestStatistics.cpp	358
TestStatistics.h	358
TraceManager.cpp	359
TraceManager.h	359
Traits.h	360
Util.cpp	361
Util.h	362
Variable.cpp	362
Variable.h	363
Waiting.cpp	364
Waiting.h	365
WaitingResource.cpp	366
WaitingResource.h	367



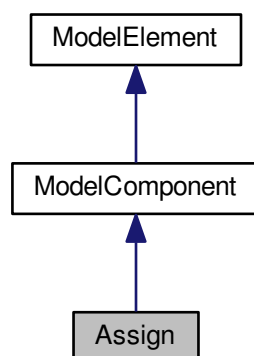
## Chapter 5

# Class Documentation

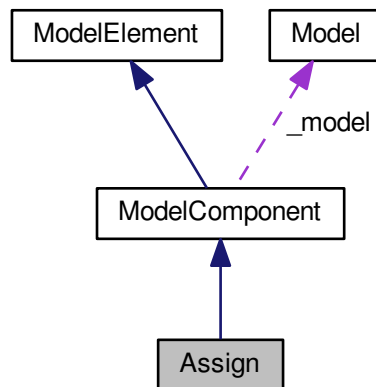
### 5.1 Assign Class Reference

```
#include <Assign.h>
```

Inheritance diagram for Assign:



Collaboration diagram for Assign:



## Classes

- class [Assignment](#)

## Public Types

- enum [DestinationType](#) : int { [DestinationType::Attribute](#), [DestinationType::Variable](#) }

## Public Member Functions

- [Assign](#) ([Model](#) \*model)
- [Assign](#) (const [Assign](#) &orig)
- virtual [~Assign](#) ()
- virtual std::string [show](#) ()
- [List](#)< [Assignment](#) \* > \* [getAssignments](#) () const

## Protected Member Functions

- virtual void [\\_execute](#) ([Entity](#) \*entity)
- virtual void [\\_loadInstance](#) (std::list< std::string > words)
- virtual std::list< std::string > \* [\\_saveInstance](#) ()
- virtual bool [\\_verifySymbols](#) (std::string \*errorMessage)

## Additional Inherited Members

### 5.1.1 Member Enumeration Documentation

#### 5.1.1.1 enum [Assign::DestinationType](#) : int [strong]

Enumerator

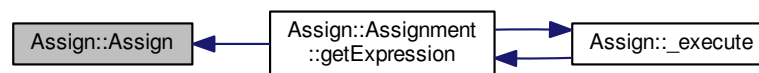
***Attribute***  
***Variable***



## 5.1.2 Constructor & Destructor Documentation

### 5.1.2.1 Assign::Assign ( Model \* *model* )

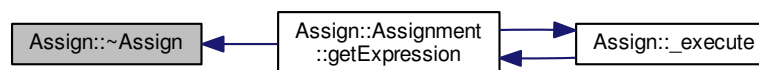
Here is the caller graph for this function:



### 5.1.2.2 Assign::Assign ( const Assign & *orig* )

### 5.1.2.3 Assign::~~Assign ( ) [virtual]

Here is the caller graph for this function:

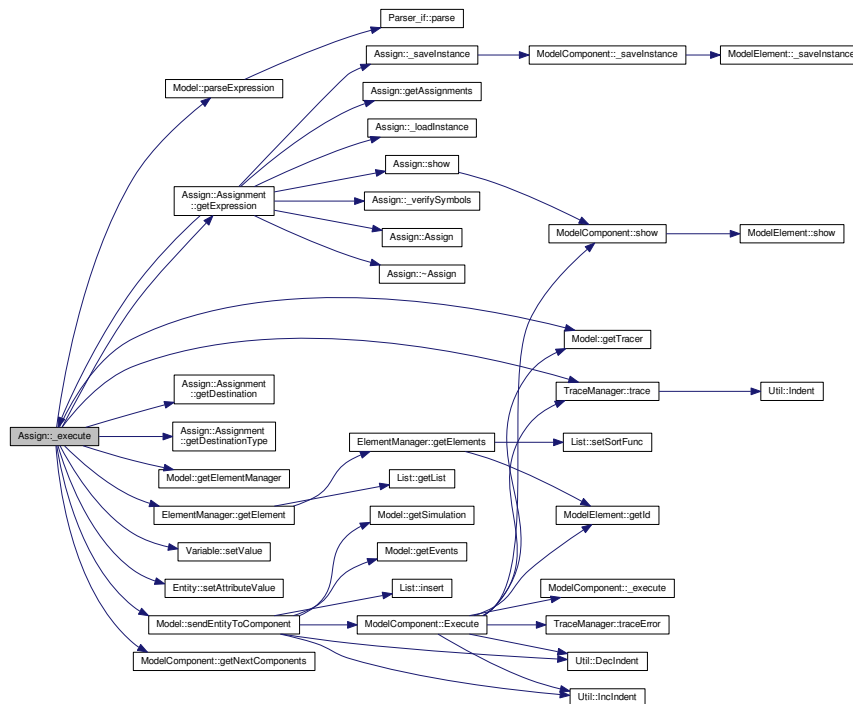


## 5.1.3 Member Function Documentation

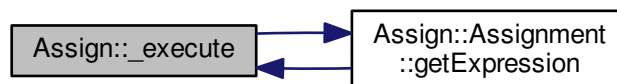
### 5.1.3.1 void Assign::\_execute ( Entity \* *entity* ) [protected], [virtual]

Implements [ModelComponent](#).

Here is the call graph for this function:



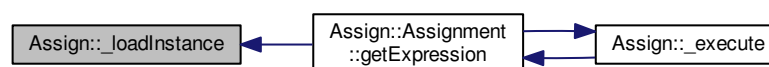
Here is the caller graph for this function:



5.1.3.2 void Assign::\_loadInstance ( std::list< std::string > words ) [protected],[virtual]

Implements [ModelElement](#).

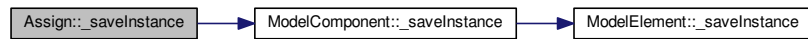
Here is the caller graph for this function:



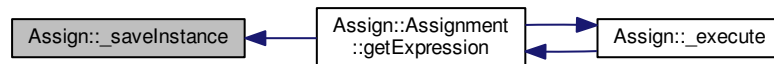
5.1.3.3 `std::list< std::string > * Assign::_saveInstance ( )` [protected],[virtual]

Reimplemented from [ModelComponent](#).

Here is the call graph for this function:



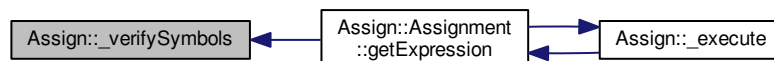
Here is the caller graph for this function:



5.1.3.4 `bool Assign::_verifySymbols ( std::string * errorMessage )` [protected],[virtual]

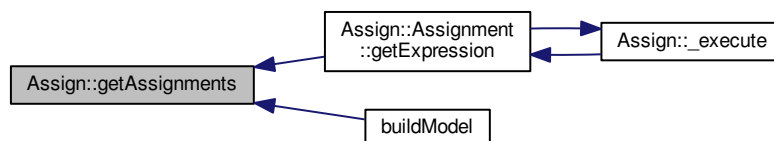
Implements [ModelElement](#).

Here is the caller graph for this function:



5.1.3.5 `List< Assign::Assignment * > * Assign::getAssignments ( )` const

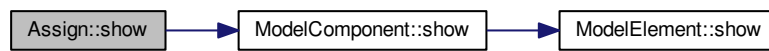
Here is the caller graph for this function:



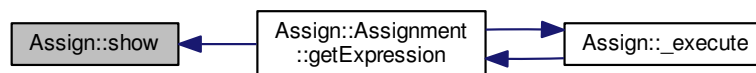
### 5.1.3.6 `std::string Assign::show ( )` [virtual]

Reimplemented from [ModelComponent](#).

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:

- [Assign.h](#)
- [Assign.cpp](#)

## 5.2 `Assign::Assignment` Class Reference

```
#include <Assign.h>
```

### Public Member Functions

- [Assignment](#) ([DestinationType](#) destinationType, std::string destination, std::string expression)
- void [setDestination](#) (std::string \_destination)
- std::string [getDestination](#) () const
- void [setDestinationType](#) ([DestinationType](#) \_destinationType)
- [DestinationType](#) [getDestinationType](#) () const
- void [setExpression](#) (std::string \_expression)
- std::string [getExpression](#) () const

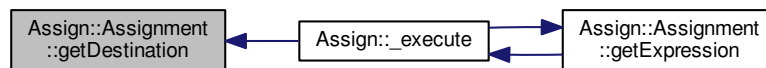
### 5.2.1 Constructor & Destructor Documentation

5.2.1.1 `Assign::Assignment::Assignment ( DestinationType destinationType, std::string destination, std::string expression ) [inline]`

### 5.2.2 Member Function Documentation

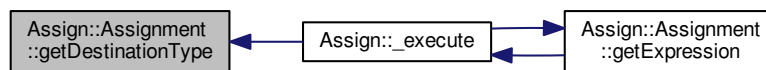
5.2.2.1 `std::string Assign::Assignment::getDestination ( ) const [inline]`

Here is the caller graph for this function:



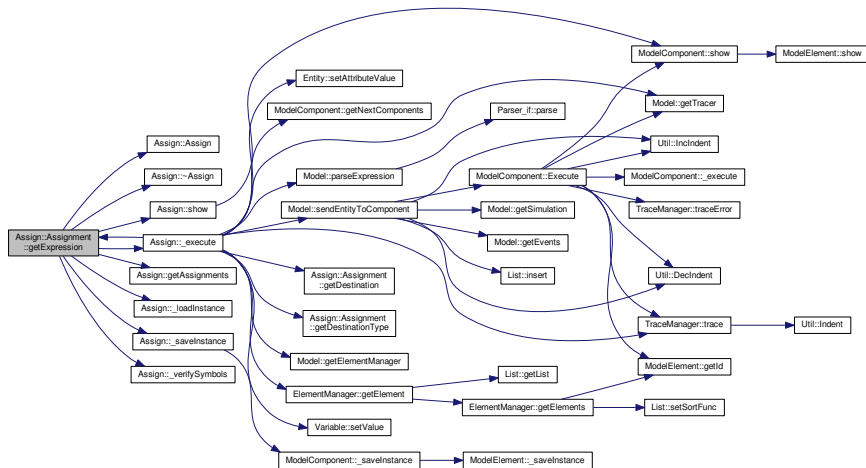
5.2.2.2 `DestinationType Assign::Assignment::getDestinationType ( ) const [inline]`

Here is the caller graph for this function:

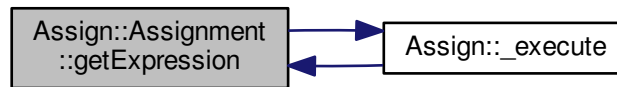


5.2.2.3 `std::string Assign::Assignment::getExpression ( ) const [inline]`

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.2.4 void Assign::Assignment::setDestination ( std::string *\_destination* ) [inline]

5.2.2.5 void Assign::Assignment::setDestinationType ( DestinationType *\_destinationType* ) [inline]

5.2.2.6 void Assign::Assignment::setExpression ( std::string *\_expression* ) [inline]

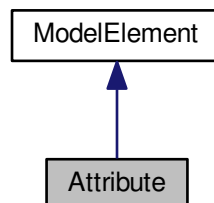
The documentation for this class was generated from the following file:

- [Assign.h](#)

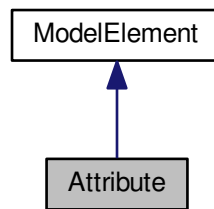
## 5.3 Attribute Class Reference

```
#include <Attribute.h>
```

Inheritance diagram for Attribute:



Collaboration diagram for Attribute:



### Public Member Functions

- [Attribute](#) ()
- [Attribute](#) (std::string name)
- [Attribute](#) (const [Attribute](#) &orig)
- virtual [~Attribute](#) ()
- virtual std::string [show](#) ()

### Protected Member Functions

- virtual void [\\_loadInstance](#) (std::list< std::string > words)
- virtual std::list< std::string > \* [\\_saveInstance](#) ()
- virtual bool [\\_verifySymbols](#) (std::string \*errorMessage)

### Additional Inherited Members

#### 5.3.1 Constructor & Destructor Documentation

5.3.1.1 `Attribute::Attribute ( )`

5.3.1.2 `Attribute::Attribute ( std::string name )`

5.3.1.3 `Attribute::Attribute ( const Attribute & orig )`

5.3.1.4 `Attribute::~~Attribute ( )` [virtual]

#### 5.3.2 Member Function Documentation

5.3.2.1 `void Attribute::_loadInstance ( std::list< std::string > words )` [protected], [virtual]

Implements [ModelElement](#).

5.3.2.2 `std::list< std::string > * Attribute::_saveInstance ( )` [protected],[virtual]

Reimplemented from [ModelElement](#).

Here is the call graph for this function:



5.3.2.3 `bool Attribute::_verifySymbols ( std::string * errorMessage )` [protected],[virtual]

Implements [ModelElement](#).

5.3.2.4 `std::string Attribute::show ( )` [virtual]

Reimplemented from [ModelElement](#).

Here is the call graph for this function:



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

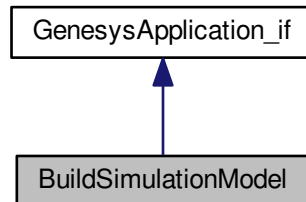
- [Attribute.h](#)
- [Attribute.cpp](#)



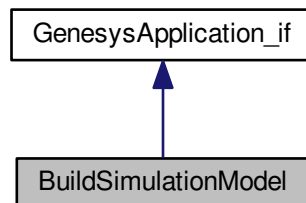
## 5.4 BuildSimulationModel Class Reference

```
#include <BuildSimulationModel.h>
```

Inheritance diagram for BuildSimulationModel:



Collaboration diagram for BuildSimulationModel:



### Public Member Functions

- [BuildSimulationModel](#) ()
- [int main](#) (int argc, char \*\*argv)

#### 5.4.1 Constructor & Destructor Documentation

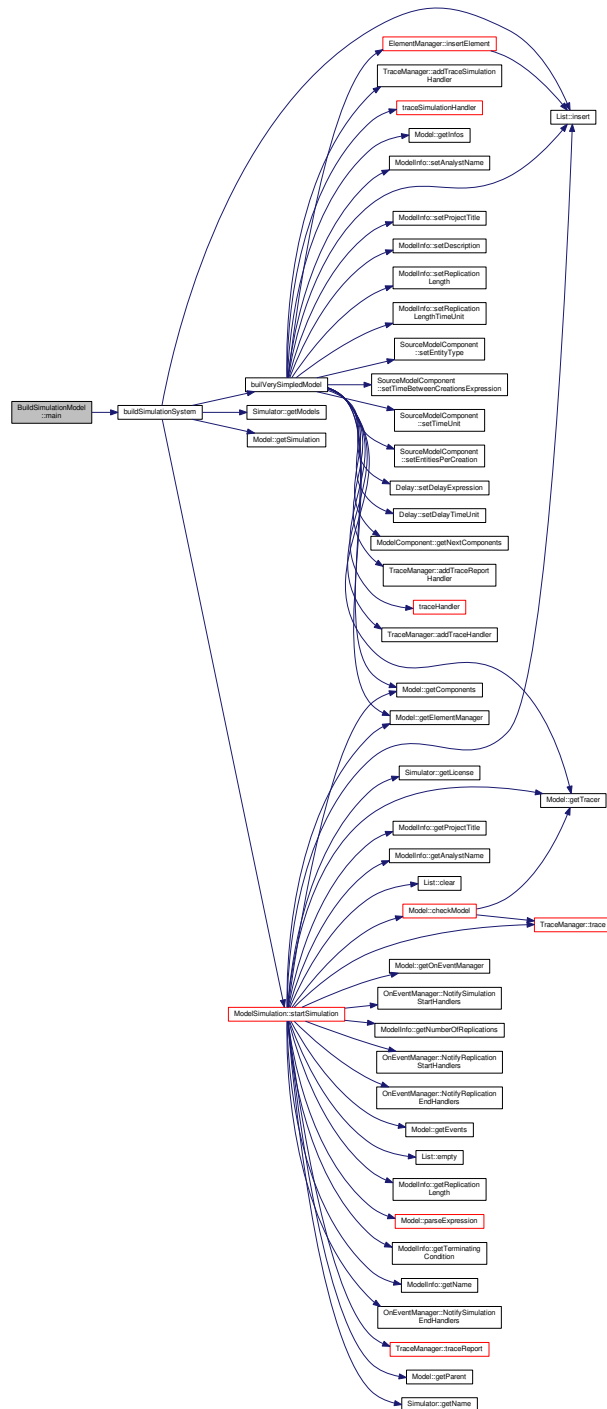
5.4.1.1 [BuildSimulationModel::BuildSimulationModel](#) ( )

#### 5.4.2 Member Function Documentation

5.4.2.1 [int BuildSimulationModel::main](#) ( int *argc*, char \*\* *argv* ) [virtual]

Implements [GenesysApplication\\_if](#).

Here is the call graph for this function:



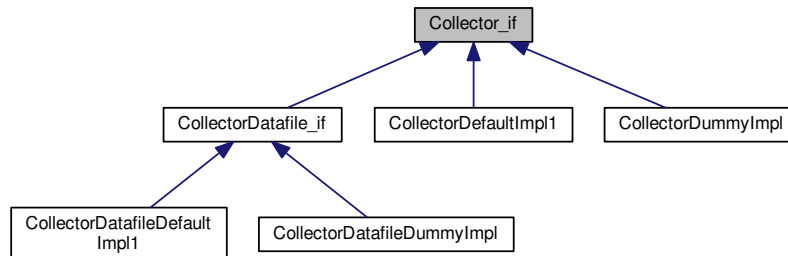
The documentation for this class was generated from the following files:

- [BuildSimulationModel.h](#)
- [BuildSimulationModel.cpp](#)

## 5.5 Collector\_if Class Reference

```
#include <Collector_if.h>
```

Inheritance diagram for Collector\_if:



### Public Member Functions

- virtual void [clear](#) ()=0
- virtual void [addValue](#) (double value)=0
- virtual double [getLastValue](#) ()=0
- virtual unsigned long [numElements](#) ()=0
- virtual void [setAddValueHandler](#) (CollectorAddValueHandler addValueHandler)=0
- virtual void [setClearHandler](#) (CollectorClearHandler clearHandler)=0

#### 5.5.1 Detailed Description

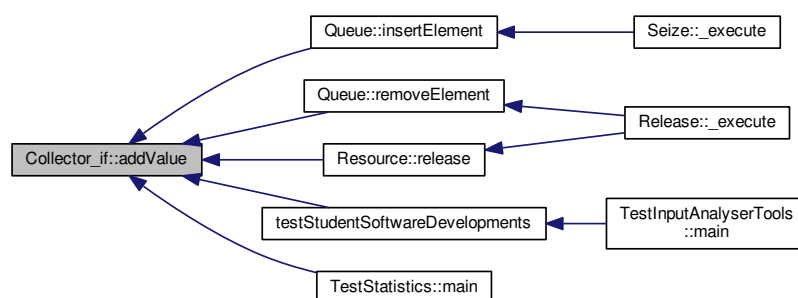
Interface for collecting values of a single stochastic variable. Values collected can be used as base for statistical analysis.

#### 5.5.2 Member Function Documentation

5.5.2.1 virtual void Collector\_if::addValue ( double value ) [pure virtual]

Implemented in [CollectorDatafileDefaultImpl1](#), [CollectorDummyImpl](#), [CollectorDatafileDummyImpl](#), and [CollectorDefaultImpl1](#).

Here is the caller graph for this function:



### 5.5.2.2 virtual void Collector\_if::clear ( ) [pure virtual]

Implemented in [CollectorDatafileDefaultImpl1](#), [CollectorDummyImpl](#), [CollectorDatafileDummyImpl](#), and [CollectorDatafileDefaultImpl1](#).

Here is the caller graph for this function:



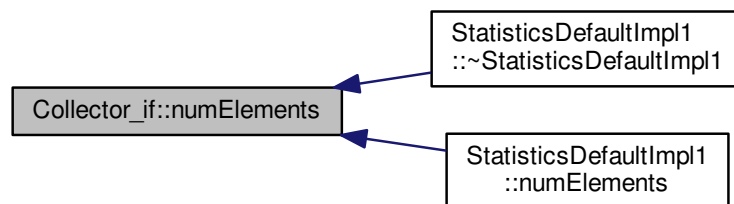
### 5.5.2.3 virtual double Collector\_if::getLastValue ( ) [pure virtual]

Implemented in [CollectorDatafileDefaultImpl1](#), [CollectorDummyImpl](#), [CollectorDatafileDummyImpl](#), and [CollectorDatafileDefaultImpl1](#).

### 5.5.2.4 virtual unsigned long Collector\_if::numElements ( ) [pure virtual]

Implemented in [CollectorDatafileDefaultImpl1](#), [CollectorDummyImpl](#), [CollectorDatafileDummyImpl](#), and [CollectorDatafileDefaultImpl1](#).

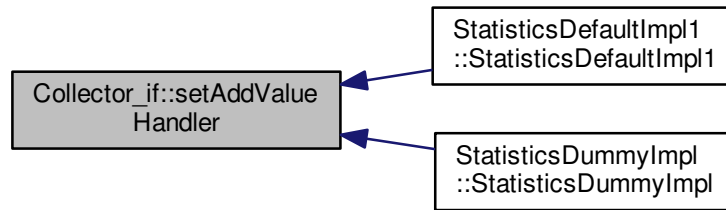
Here is the caller graph for this function:



### 5.5.2.5 virtual void Collector\_if::setAddValueHandler ( CollectorAddValueHandler addValueHandler ) [pure virtual]

Implemented in [CollectorDatafileDefaultImpl1](#), [CollectorDatafileDummyImpl](#), [CollectorDummyImpl](#), and [CollectorDatafileDefaultImpl1](#).

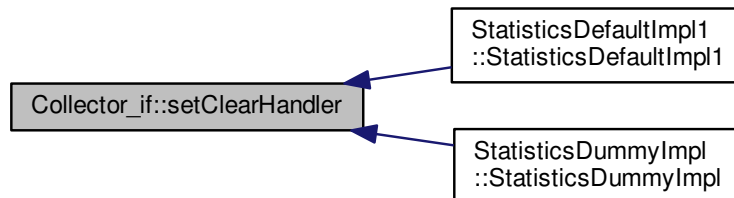
Here is the caller graph for this function:



5.5.2.6 `virtual void Collector_if::setClearHandler ( CollectorClearHandler clearHandler )` [pure virtual]

Implemented in [CollectorDatafileDefaultImpl1](#), [CollectorDatafileDummyImpl](#), [CollectorDummyImpl](#), and [CollectorDatafileDefaultImpl1](#).

Here is the caller graph for this function:



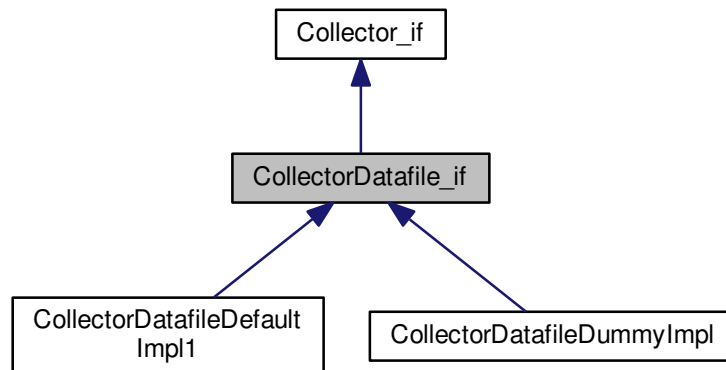
The documentation for this class was generated from the following file:

- [Collector\\_if.h](#)

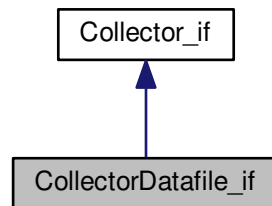
## 5.6 CollectorDatafile\_if Class Reference

```
#include <CollectorDatafile_if.h>
```

Inheritance diagram for CollectorDatafile\_if:



Collaboration diagram for CollectorDatafile\_if:



## Public Member Functions

- virtual double [getValue](#) (unsigned int rank)=0
- virtual void [seekFirstValue](#) ()=0
- virtual double [getNextValue](#) ()=0
- virtual std::string [getDataFilename](#) ()=0
- virtual void [setDataFilename](#) (std::string filename)=0

### 5.6.1 Detailed Description

Interface for collecting values of a stochastic variable that will be stores in a datafile.

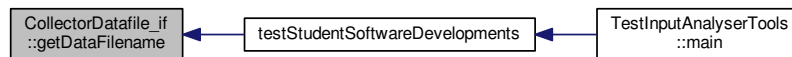
## 5.6.2 Member Function Documentation

### 5.6.2.1 `virtual std::string CollectorDatafile_if::getDataFilename ( )` [pure virtual]

Get the next value in the file and advances the pointer

Implemented in [CollectorDatafileDefaultImpl1](#), and [CollectorDatafileDummyImpl](#).

Here is the caller graph for this function:



### 5.6.2.2 `virtual double CollectorDatafile_if::getNextValue ( )` [pure virtual]

Set the pointer to the first value in the file

Implemented in [CollectorDatafileDefaultImpl1](#), and [CollectorDatafileDummyImpl](#).

### 5.6.2.3 `virtual double CollectorDatafile_if::getValue ( unsigned int rank )` [pure virtual]

Implemented in [CollectorDatafileDefaultImpl1](#), and [CollectorDatafileDummyImpl](#).

### 5.6.2.4 `virtual void CollectorDatafile_if::seekFirstValue ( )` [pure virtual]

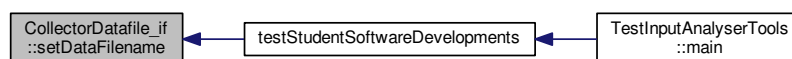
Get a value from a specific position

Implemented in [CollectorDatafileDefaultImpl1](#), and [CollectorDatafileDummyImpl](#).

### 5.6.2.5 `virtual void CollectorDatafile_if::setDataFilename ( std::string filename )` [pure virtual]

Implemented in [CollectorDatafileDefaultImpl1](#), and [CollectorDatafileDummyImpl](#).

Here is the caller graph for this function:



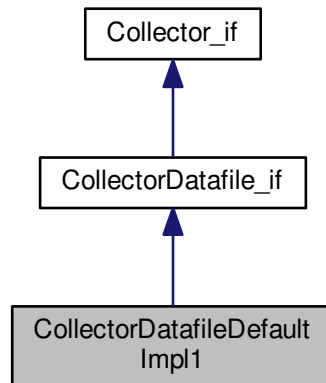
The documentation for this class was generated from the following file:

- [CollectorDatafile\\_if.h](#)

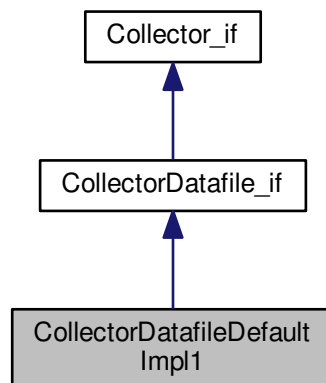
## 5.7 CollectorDatafileDefaultImpl1 Class Reference

```
#include <CollectorDatafileDefaultImpl1.h>
```

Inheritance diagram for CollectorDatafileDefaultImpl1:



Collaboration diagram for CollectorDatafileDefaultImpl1:



### Public Member Functions

- [CollectorDatafileDefaultImpl1](#) ()
- [CollectorDatafileDefaultImpl1](#) (const [CollectorDatafileDefaultImpl1](#) &orig)
- virtual [~CollectorDatafileDefaultImpl1](#) ()
- void [clear](#) ()



- void [addValue](#) (double value)
- double [getLastValue](#) ()
- unsigned long [numElements](#) ()
- double [getValue](#) (unsigned int num)
- double [getNextValue](#) ()
- void [seekFirstValue](#) ()
- std::string [getDataFilename](#) ()
- void [setDataFilename](#) (std::string filename)
- void [setAddValueHandler](#) ([CollectorAddValueHandler](#) addValueHandler)
- void [setClearHandler](#) ([CollectorClearHandler](#) clearHandler)

### 5.7.1 Constructor & Destructor Documentation

5.7.1.1 `CollectorDatafileDefaultImpl1::CollectorDatafileDefaultImpl1 ( )`

5.7.1.2 `CollectorDatafileDefaultImpl1::CollectorDatafileDefaultImpl1 ( const CollectorDatafileDefaultImpl1 & orig )`

5.7.1.3 `CollectorDatafileDefaultImpl1::~~CollectorDatafileDefaultImpl1 ( )` [virtual]

### 5.7.2 Member Function Documentation

5.7.2.1 `void CollectorDatafileDefaultImpl1::addValue ( double value )` [virtual]

Implements [Collector\\_if](#).

5.7.2.2 `void CollectorDatafileDefaultImpl1::clear ( )` [virtual]

Implements [Collector\\_if](#).

5.7.2.3 `std::string CollectorDatafileDefaultImpl1::getDataFilename ( )` [virtual]

Get the next value in the file and advances the pointer

Implements [CollectorDatafile\\_if](#).

5.7.2.4 `double CollectorDatafileDefaultImpl1::getLastValue ( )` [virtual]

Implements [Collector\\_if](#).

5.7.2.5 `double CollectorDatafileDefaultImpl1::getNextValue ( )` [virtual]

Set the pointer to the first value in the file

Implements [CollectorDatafile\\_if](#).

5.7.2.6 `double CollectorDatafileDefaultImpl1::getValue ( unsigned int num )` [virtual]

Implements [CollectorDatafile\\_if](#).

5.7.2.7 `unsigned long CollectorDatafileDefaultImpl1::numElements ( )` [virtual]

Implements [Collector\\_if](#).

5.7.2.8 `void CollectorDatafileDefaultImpl1::seekFirstValue ( )` [virtual]

Get a value from a specific position

Implements [CollectorDatafile\\_if](#).

5.7.2.9 `void CollectorDatafileDefaultImpl1::setAddValueHandler ( CollectorAddValueHandler addValueHandler )`  
[virtual]

Implements [Collector\\_if](#).

5.7.2.10 `void CollectorDatafileDefaultImpl1::setClearHandler ( CollectorClearHandler clearHandler )` [virtual]

Implements [Collector\\_if](#).

5.7.2.11 `void CollectorDatafileDefaultImpl1::setDataFilename ( std::string filename )` [virtual]

Implements [CollectorDatafile\\_if](#).

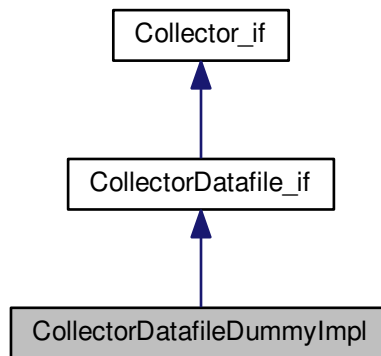
The documentation for this class was generated from the following files:

- [CollectorDatafileDefaultImpl1.h](#)
- [CollectorDatafileDefaultImpl1.cpp](#)

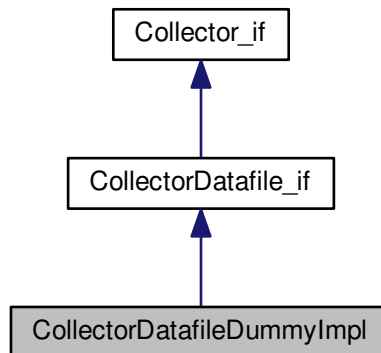
## 5.8 CollectorDatafileDummyImpl Class Reference

```
#include <CollectorDatafileDummyImpl.h>
```

Inheritance diagram for CollectorDatafileDummyImpl:



Collaboration diagram for CollectorDatafileDummyImpl:



### Public Member Functions

- [CollectorDatafileDummyImpl](#) ()
- [CollectorDatafileDummyImpl](#) (const [CollectorDatafileDummyImpl](#) &orig)
- [~CollectorDatafileDummyImpl](#) ()
- void [clear](#) ()
- void [addValue](#) (double value)

- double [getLastValue](#) ()
- unsigned long [numElements](#) ()
- double [getValue](#) (unsigned int num)
- double [getNextValue](#) ()
- void [seekFirstValue](#) ()
- std::string [getDataFilename](#) ()
- void [setDataFilename](#) (std::string filename)
- void [setAddValueHandler](#) ([CollectorAddValueHandler](#) addValueHandler)
- void [setClearHandler](#) ([CollectorClearHandler](#) clearHandler)

## 5.8.1 Constructor & Destructor Documentation

5.8.1.1 [CollectorDatafileDummyImpl::CollectorDatafileDummyImpl](#) ( )

5.8.1.2 [CollectorDatafileDummyImpl::CollectorDatafileDummyImpl](#) ( const [CollectorDatafileDummyImpl](#) & *orig* )

5.8.1.3 [CollectorDatafileDummyImpl::~~CollectorDatafileDummyImpl](#) ( )

## 5.8.2 Member Function Documentation

5.8.2.1 void [CollectorDatafileDummyImpl::addValue](#) ( double *value* ) [virtual]

Implements [Collector\\_if](#).

5.8.2.2 void [CollectorDatafileDummyImpl::clear](#) ( ) [virtual]

Implements [Collector\\_if](#).

5.8.2.3 std::string [CollectorDatafileDummyImpl::getDataFilename](#) ( ) [virtual]

Get the next value in the file and advances the pointer

Implements [CollectorDatafile\\_if](#).

5.8.2.4 double [CollectorDatafileDummyImpl::getLastValue](#) ( ) [virtual]

Implements [Collector\\_if](#).

5.8.2.5 double [CollectorDatafileDummyImpl::getNextValue](#) ( ) [virtual]

Set the pointer to the first value in the file

Implements [CollectorDatafile\\_if](#).

5.8.2.6 `double CollectorDatafileDummyImpl::getValue ( unsigned int num )` [virtual]

Implements [CollectorDatafile\\_if](#).

5.8.2.7 `unsigned long CollectorDatafileDummyImpl::numElements ( )` [virtual]

Implements [Collector\\_if](#).

5.8.2.8 `void CollectorDatafileDummyImpl::seekFirstValue ( )` [virtual]

Get a value from a specific position

Implements [CollectorDatafile\\_if](#).

5.8.2.9 `void CollectorDatafileDummyImpl::setAddValueHandler ( CollectorAddValueHandler addValueHandler )`  
[virtual]

Implements [Collector\\_if](#).

5.8.2.10 `void CollectorDatafileDummyImpl::setClearHandler ( CollectorClearHandler clearHandler )` [virtual]

Implements [Collector\\_if](#).

5.8.2.11 `void CollectorDatafileDummyImpl::setDataFilename ( std::string filename )` [virtual]

Implements [CollectorDatafile\\_if](#).

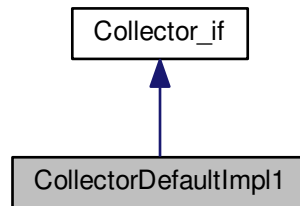
The documentation for this class was generated from the following files:

- [CollectorDatafileDummyImpl.h](#)
- [CollectorDatafileDummyImpl.cpp](#)

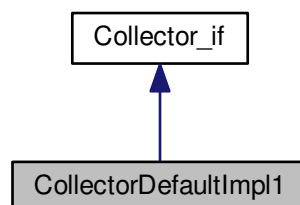
## 5.9 CollectorDefaultImpl1 Class Reference

```
#include <CollectorDefaultImpl1.h>
```

Inheritance diagram for CollectorDefaultImpl1:



Collaboration diagram for CollectorDefaultImpl1:



### Public Member Functions

- [CollectorDefaultImpl1](#) ()
- [CollectorDefaultImpl1](#) (const [CollectorDefaultImpl1](#) &orig)
- virtual [~CollectorDefaultImpl1](#) ()
- void [clear](#) ()
- void [addValue](#) (double value)
- double [getLastValue](#) ()
- unsigned long [numElements](#) ()
- void [setAddValueHandler](#) ([CollectorAddValueHandler](#) addValueHandler)
- void [setClearHandler](#) ([CollectorClearHandler](#) clearHandler)

## 5.9.1 Constructor & Destructor Documentation

5.9.1.1 `CollectorDefaultImpl1::CollectorDefaultImpl1 ( )`

5.9.1.2 `CollectorDefaultImpl1::CollectorDefaultImpl1 ( const CollectorDefaultImpl1 & orig )`

5.9.1.3 `CollectorDefaultImpl1::~~CollectorDefaultImpl1 ( )` [virtual]

## 5.9.2 Member Function Documentation

5.9.2.1 `void CollectorDefaultImpl1::addValue ( double value )` [virtual]

Implements [Collector\\_if](#).

5.9.2.2 `void CollectorDefaultImpl1::clear ( )` [virtual]

Implements [Collector\\_if](#).

5.9.2.3 `double CollectorDefaultImpl1::getLastValue ( )` [virtual]

Implements [Collector\\_if](#).

5.9.2.4 `unsigned long CollectorDefaultImpl1::numElements ( )` [virtual]

Implements [Collector\\_if](#).

5.9.2.5 `void CollectorDefaultImpl1::setAddValueHandler ( CollectorAddValueHandler addValueHandler )` [virtual]

Implements [Collector\\_if](#).

5.9.2.6 `void CollectorDefaultImpl1::setClearHandler ( CollectorClearHandler clearHandler )` [virtual]

Implements [Collector\\_if](#).

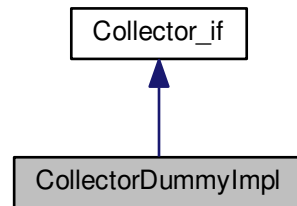
The documentation for this class was generated from the following files:

- [CollectorDefaultImpl1.h](#)
- [CollectorDefaultImpl1.cpp](#)

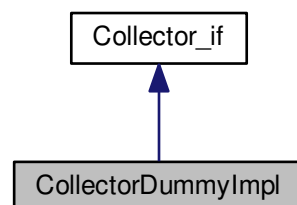
## 5.10 CollectorDummyImpl Class Reference

```
#include <CollectorDummyImpl.h>
```

Inheritance diagram for CollectorDummyImpl:



Collaboration diagram for CollectorDummyImpl:



### Public Member Functions

- [CollectorDummyImpl](#) ()
- [CollectorDummyImpl](#) (const [CollectorDummyImpl](#) &orig)
- [~CollectorDummyImpl](#) ()
- void [clear](#) ()
- void [addValue](#) (double value)
- double [getLastValue](#) ()
- unsigned long [numElements](#) ()
- void [setAddValueHandler](#) ([CollectorAddValueHandler](#) addValueHandler)
- void [setClearHandler](#) ([CollectorClearHandler](#) clearHandler)



### 5.10.1 Constructor & Destructor Documentation

5.10.1.1 `CollectorDummyImpl::CollectorDummyImpl ( )`

5.10.1.2 `CollectorDummyImpl::CollectorDummyImpl ( const CollectorDummyImpl & orig )`

5.10.1.3 `CollectorDummyImpl::~~CollectorDummyImpl ( )`

### 5.10.2 Member Function Documentation

5.10.2.1 `void CollectorDummyImpl::addValue ( double value ) [virtual]`

Implements [Collector\\_if](#).

5.10.2.2 `void CollectorDummyImpl::clear ( ) [virtual]`

Implements [Collector\\_if](#).

5.10.2.3 `double CollectorDummyImpl::getLastValue ( ) [virtual]`

Implements [Collector\\_if](#).

5.10.2.4 `unsigned long CollectorDummyImpl::numElements ( ) [virtual]`

Implements [Collector\\_if](#).

5.10.2.5 `void CollectorDummyImpl::setAddValueHandler ( CollectorAddValueHandler addValueHandler ) [virtual]`

Implements [Collector\\_if](#).

5.10.2.6 `void CollectorDummyImpl::setClearHandler ( CollectorClearHandler clearHandler ) [virtual]`

Implements [Collector\\_if](#).

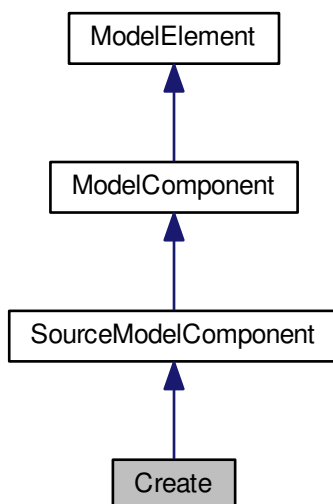
The documentation for this class was generated from the following files:

- [CollectorDummyImpl.h](#)
- [CollectorDummyImpl.cpp](#)

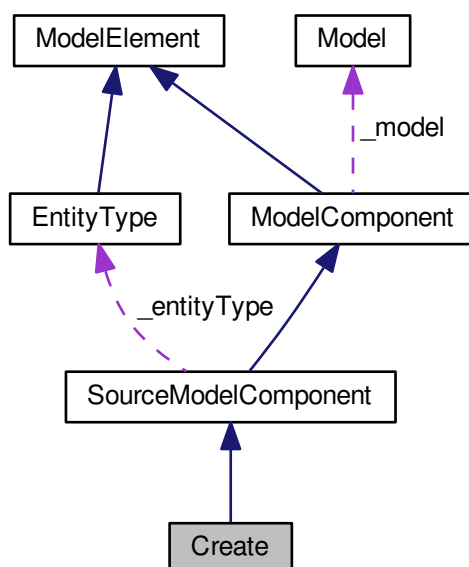
## 5.11 Create Class Reference

```
#include <Create.h>
```

Inheritance diagram for Create:



Collaboration diagram for Create:



## Public Member Functions

- [Create](#) ([Model](#) \*model)
- [Create](#) (const [Create](#) &orig)
- virtual [~Create](#) ()
- virtual std::string [show](#) ()

## Protected Member Functions

- virtual void [\\_execute](#) ([Entity](#) \*entity)
- virtual void [\\_loadInstance](#) (std::list< std::string > words)
- virtual std::list< std::string > \* [\\_saveInstance](#) ()
- virtual bool [\\_verifySymbols](#) (std::string \*errorMessage)

## Additional Inherited Members

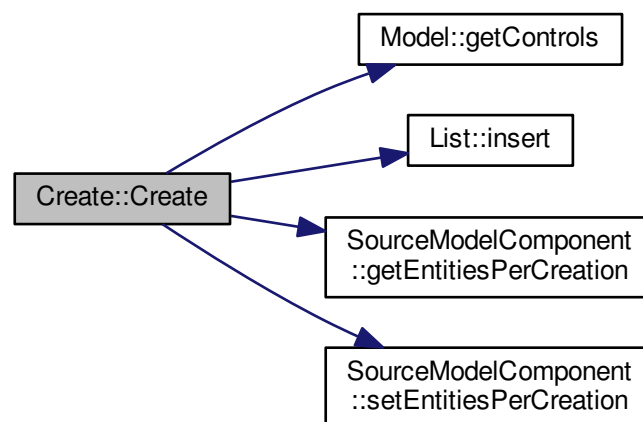
### 5.11.1 Detailed Description

[Create](#) is the most basic component to include the first entities into the model, and therefore is a source component (derived from [SourceModelComponent](#))

### 5.11.2 Constructor & Destructor Documentation

#### 5.11.2.1 [Create::Create](#) ( [Model](#) \* *model* )

Here is the call graph for this function:



5.11.2.2 `Create::Create ( const Create & orig )`

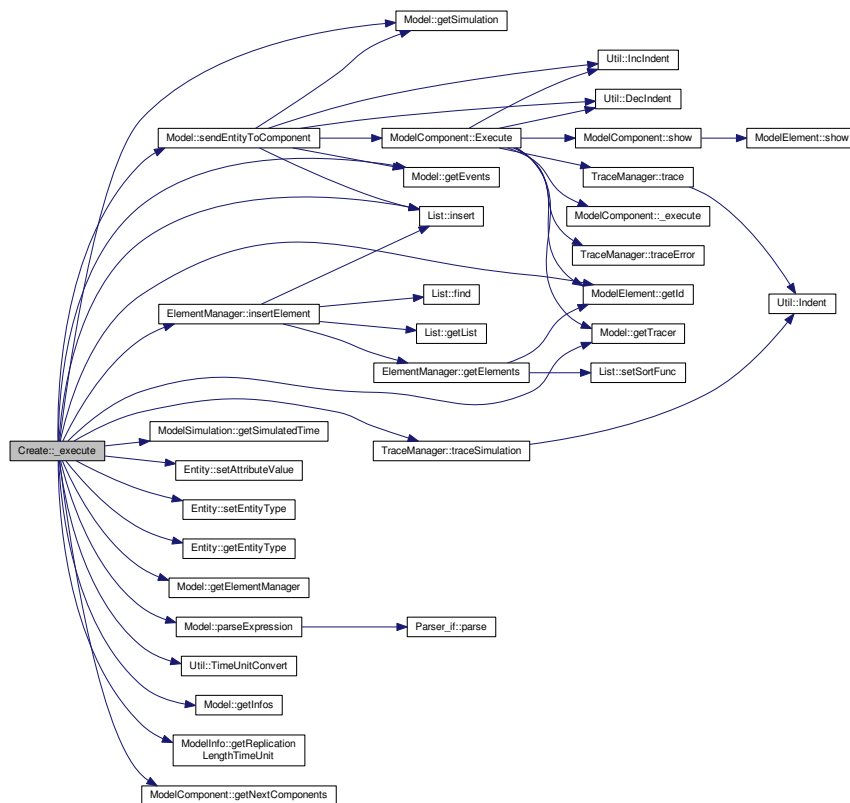
5.11.2.3 `Create::~~Create ( )` [virtual]

### 5.11.3 Member Function Documentation

5.11.3.1 `void Create::_execute ( Entity * entity )` [protected],[virtual]

Implements [ModelComponent](#).

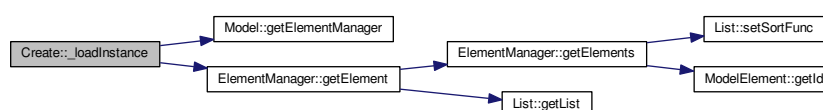
Here is the call graph for this function:



5.11.3.2 `void Create::_loadInstance ( std::list< std::string > words )` [protected],[virtual]

Implements [ModelElement](#).

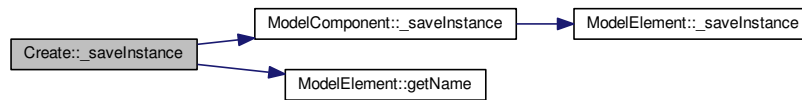
Here is the call graph for this function:



5.11.3.3 `std::list< std::string > * Create::_saveInstance ( )` [protected],[virtual]

Reimplemented from [ModelComponent](#).

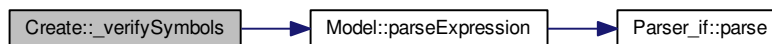
Here is the call graph for this function:



5.11.3.4 `bool Create::_verifySymbols ( std::string * errorMessage )` [protected],[virtual]

Implements [ModelElement](#).

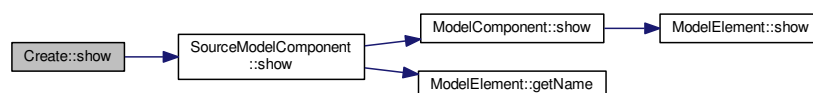
Here is the call graph for this function:



5.11.3.5 `std::string Create::show ( )` [virtual]

Reimplemented from [SourceModelComponent](#).

Here is the call graph for this function:



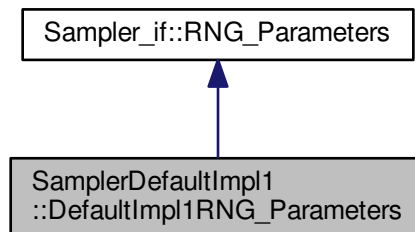
The documentation for this class was generated from the following files:

- [Create.h](#)
- [Create.cpp](#)

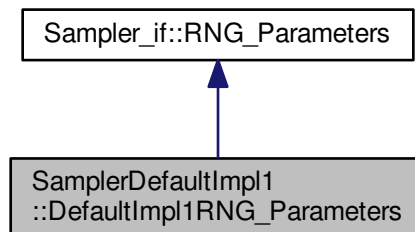
## 5.12 SamplerDefaultImpl1::DefaultImpl1RNG\_Parameters Class Reference

```
#include <SamplerDefaultImpl1.h>
```

Inheritance diagram for SamplerDefaultImpl1::DefaultImpl1RNG\_Parameters:



Collaboration diagram for SamplerDefaultImpl1::DefaultImpl1RNG\_Parameters:



### Public Attributes

- unsigned int [seed](#) = 666
- unsigned int [module](#) = 2147483647
- unsigned int [multiplier](#) = 950706376

### 5.12.1 Member Data Documentation

5.12.1.1 unsigned int SamplerDefaultImpl1::DefaultImpl1RNG\_Parameters::module = 2147483647

5.12.1.2 unsigned int SamplerDefaultImpl1::DefaultImpl1RNG\_Parameters::multiplier = 950706376

5.12.1.3 unsigned int SamplerDefaultImpl1::DefaultImpl1RNG\_Parameters::seed = 666

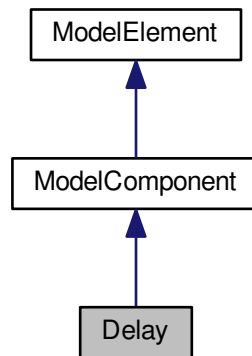
The documentation for this class was generated from the following file:

- [SamplerDefaultImpl1.h](#)

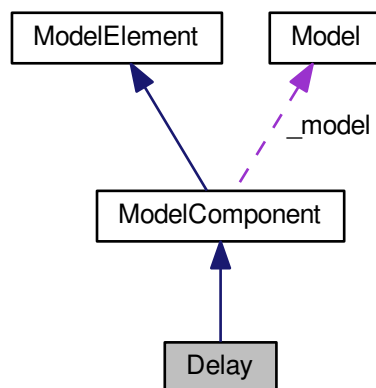
## 5.13 Delay Class Reference

```
#include <Delay.h>
```

Inheritance diagram for Delay:



Collaboration diagram for Delay:



### Public Member Functions

- [Delay](#) ([Model](#) \*model)
- [Delay](#) (const [Delay](#) &orig)
- virtual [~Delay](#) ()
- void [setDelayExpression](#) (std::string \_delayExpression)
- std::string [getDelayExpression](#) () const
- void [setDelayTimeUnit](#) ([Util::TimeUnit](#) \_delayTimeUnit)
- [Util::TimeUnit](#) [getDelayTimeUnit](#) () const
- virtual std::string [show](#) ()

## Protected Member Functions

- virtual void `_execute` (`Entity *entity`)
- virtual void `_loadInstance` (`std::list< std::string > words`)
- virtual `std::list< std::string > * _saveInstance` ()
- virtual bool `_verifySymbols` (`std::string *errorMessage`)

## Additional Inherited Members

### 5.13.1 Constructor & Destructor Documentation

5.13.1.1 `Delay::Delay ( Model * model )`

5.13.1.2 `Delay::Delay ( const Delay & orig )`

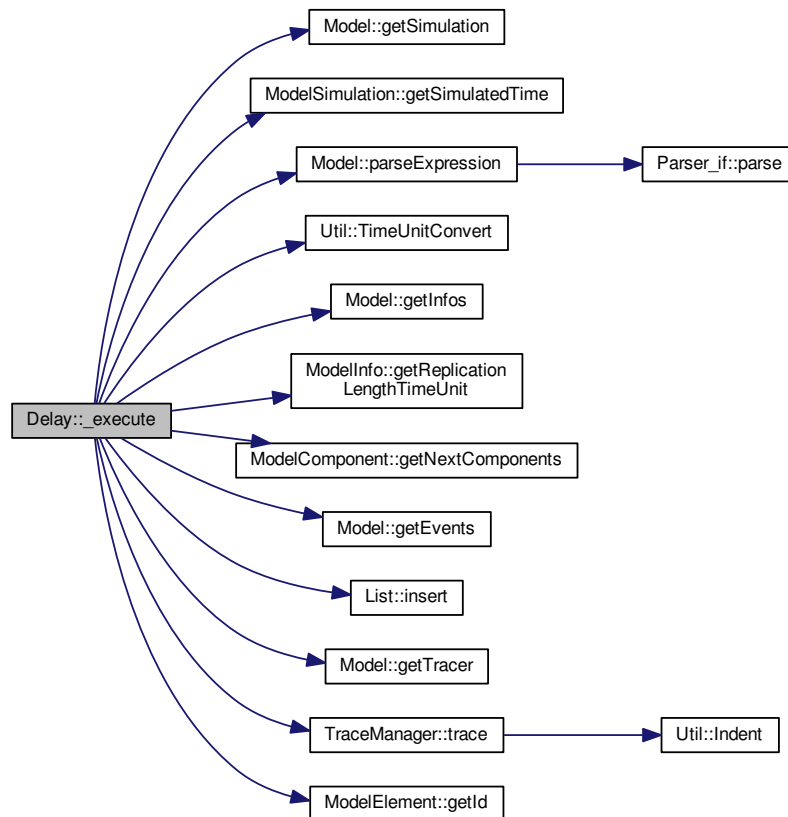
5.13.1.3 `Delay::~Delay ( )` [virtual]

### 5.13.2 Member Function Documentation

5.13.2.1 `void Delay::_execute ( Entity * entity )` [protected],[virtual]

Implements [ModelComponent](#).

Here is the call graph for this function:





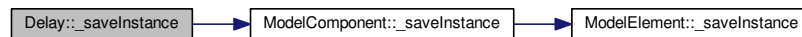
5.13.2.2 `void Delay::_loadInstance ( std::list< std::string > words )` [protected],[virtual]

Implements [ModelElement](#).

5.13.2.3 `std::list< std::string > * Delay::_saveInstance ( )` [protected],[virtual]

Reimplemented from [ModelComponent](#).

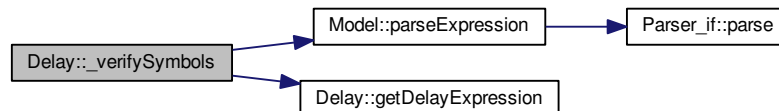
Here is the call graph for this function:



5.13.2.4 `bool Delay::_verifySymbols ( std::string * errorMessage )` [protected],[virtual]

Implements [ModelElement](#).

Here is the call graph for this function:



5.13.2.5 `std::string Delay::getDelayExpression ( ) const`

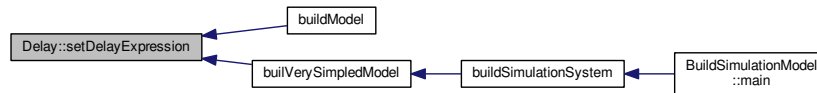
Here is the caller graph for this function:



#### 5.13.2.6 Util::TimeUnit Delay::getDelayTimeUnit ( ) const

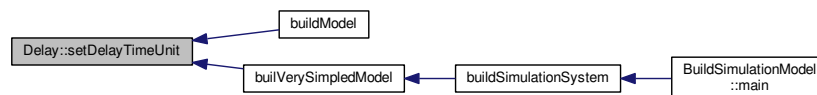
#### 5.13.2.7 void Delay::setDelayExpression ( std::string \_delayExpression )

Here is the caller graph for this function:



#### 5.13.2.8 void Delay::setDelayTimeUnit ( Util::TimeUnit \_delayTimeUnit )

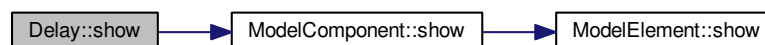
Here is the caller graph for this function:



#### 5.13.2.9 std::string Delay::show ( ) [virtual]

Reimplemented from [ModelComponent](#).

Here is the call graph for this function:



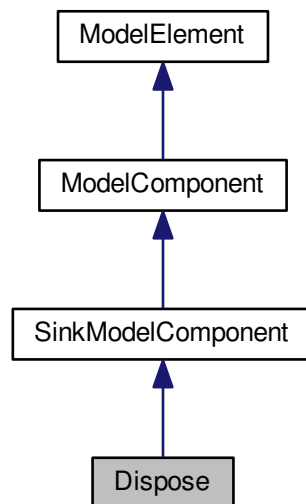
The documentation for this class was generated from the following files:

- [Delay.h](#)
- [Delay.cpp](#)

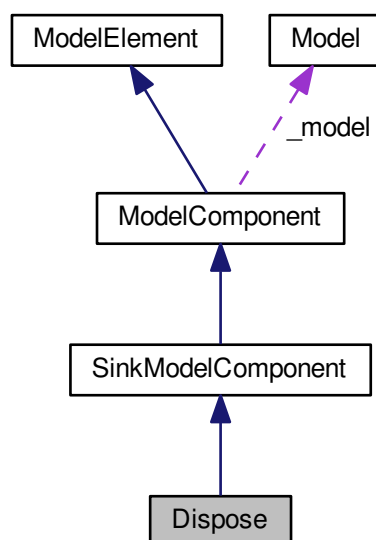
## 5.14 Dispose Class Reference

```
#include <Dispose.h>
```

Inheritance diagram for Dispose:



Collaboration diagram for Dispose:



## Public Member Functions

- [Dispose](#) ([Model](#) \*model)
- [Dispose](#) (const [Dispose](#) &orig)
- virtual [~Dispose](#) ()
- virtual std::string [show](#) ()
- void [setCollectStatistics](#) (bool \_collectStatistics)
- bool [isCollectStatistics](#) () const

## Protected Member Functions

- virtual void [\\_execute](#) ([Entity](#) \*entity)
- virtual void [\\_loadInstance](#) (std::list< std::string > words)
- virtual std::list< std::string > \* [\\_saveInstance](#) ()
- virtual bool [\\_verifySymbols](#) (std::string \*errorMessage)

## Additional Inherited Members

### 5.14.1 Constructor & Destructor Documentation

5.14.1.1 `Dispose::Dispose ( Model * model )`

5.14.1.2 `Dispose::Dispose ( const Dispose & orig )`

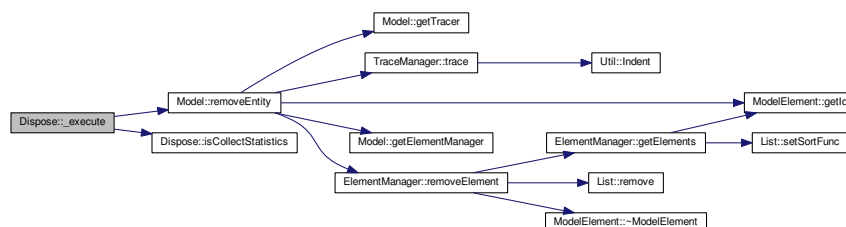
5.14.1.3 `Dispose::~~Dispose ( )` [virtual]

### 5.14.2 Member Function Documentation

5.14.2.1 `void Dispose::_execute ( Entity * entity )` [protected],[virtual]

Implements [ModelComponent](#).

Here is the call graph for this function:



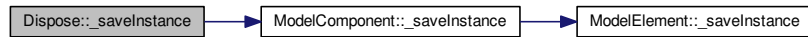
5.14.2.2 `void Dispose::_loadInstance ( std::list< std::string > words )` [protected],[virtual]

Implements [ModelElement](#).

5.14.2.3 `std::list< std::string > * Dispose::_saveInstance ( )` [protected],[virtual]

Reimplemented from [ModelComponent](#).

Here is the call graph for this function:



5.14.2.4 `bool Dispose::_verifySymbols ( std::string * errorMessage )` [protected],[virtual]

Implements [ModelElement](#).

5.14.2.5 `bool Dispose::isCollectStatistics ( ) const`

Here is the caller graph for this function:

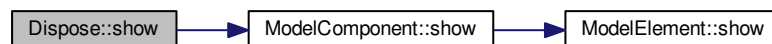


5.14.2.6 `void Dispose::setCollectStatistics ( bool _collectStatistics )`

5.14.2.7 `std::string Dispose::show ( )` [virtual]

Reimplemented from [ModelComponent](#).

Here is the call graph for this function:



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

- [Dispose.h](#)
- [Dispose.cpp](#)

## 5.15 ElementManager Class Reference

```
#include <ElementManager.h>
```

### Public Member Functions

- [ElementManager](#) ([Model](#) \*model)
- [ElementManager](#) (const [ElementManager](#) &orig)
- virtual [~ElementManager](#) ()
- bool [insertElement](#) (std::string infraTypename, [ModelElement](#) \*infra)
- void [removeElement](#) (std::string infraTypename, [ModelElement](#) \*infra)
- [List](#)< [ModelElement](#) \* > \* [getElements](#) (std::string infraTypename) const
- [ModelElement](#) \* [getElement](#) (std::string infraTypename, [Util::identification](#) id)
- [ModelElement](#) \* [getElement](#) (std::string infraTypename, std::string name)
- std::list< std::string > \* [getElementTypenames](#) () const
- void [show](#) ()

### 5.15.1 Detailed Description

The [ElementManager](#) is responsible for inserting and removing elements ([ModelElement](#)) used by components, in a consistent way. TO FIX: No direct access for insertion or deletion should be allow

### 5.15.2 Constructor & Destructor Documentation

#### 5.15.2.1 [ElementManager::ElementManager](#) ( [Model](#) \* *model* )

Elements are organized as a map from a string (key), the type of an element, and a list of elements of that type

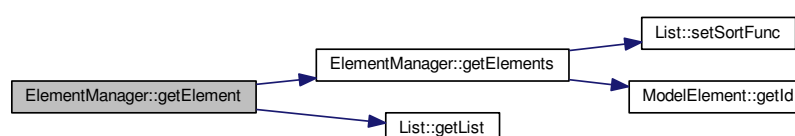
#### 5.15.2.2 [ElementManager::ElementManager](#) ( const [ElementManager](#) & *orig* )

#### 5.15.2.3 [ElementManager::~~ElementManager](#) ( ) [virtual]

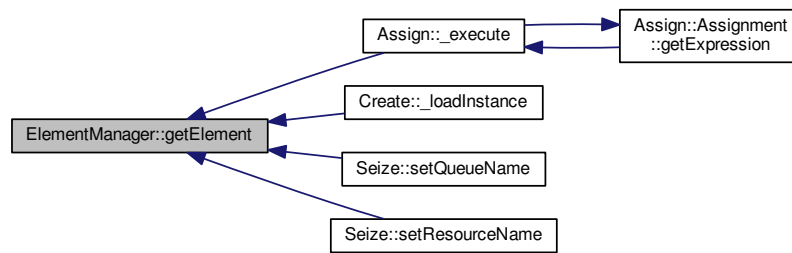
### 5.15.3 Member Function Documentation

#### 5.15.3.1 [ModelElement](#) \* [ElementManager::getElement](#) ( std::string *infraTypename*, [Util::identification](#) *id* )

Here is the call graph for this function:

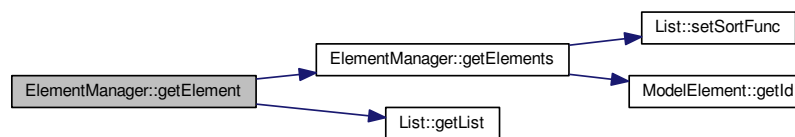


Here is the caller graph for this function:



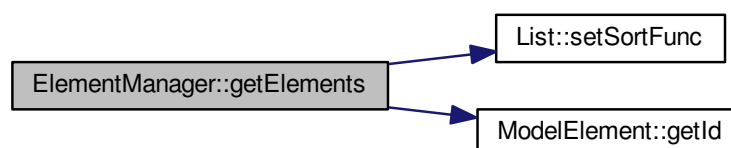
### 5.15.3.2 `ModelElement *` `ElementManager::getElement ( std::string infraTypename, std::string name )`

Here is the call graph for this function:

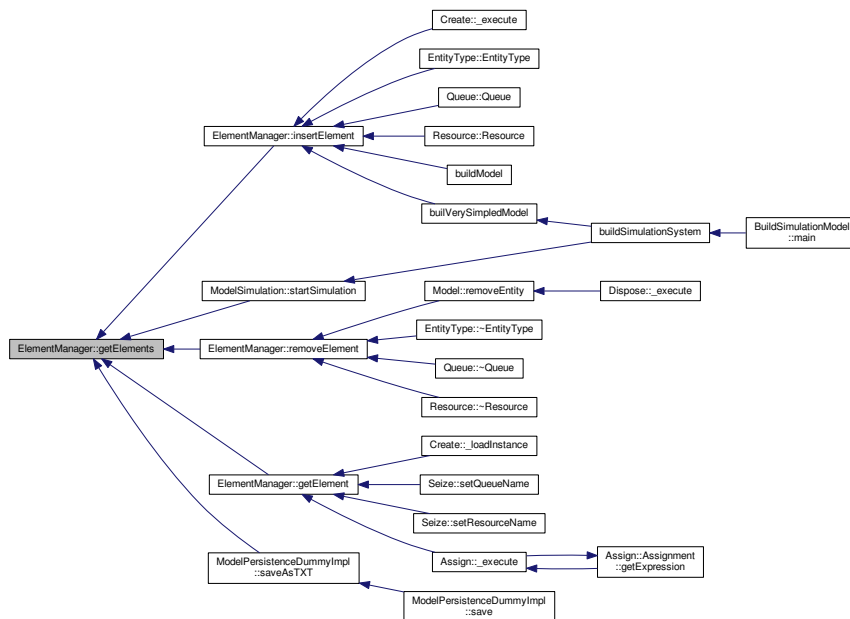


### 5.15.3.3 `List< ModelElement * > *` `ElementManager::getElements ( std::string infraTypename ) const`

Here is the call graph for this function:

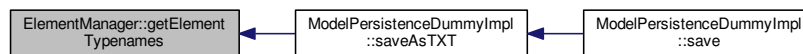


Here is the caller graph for this function:



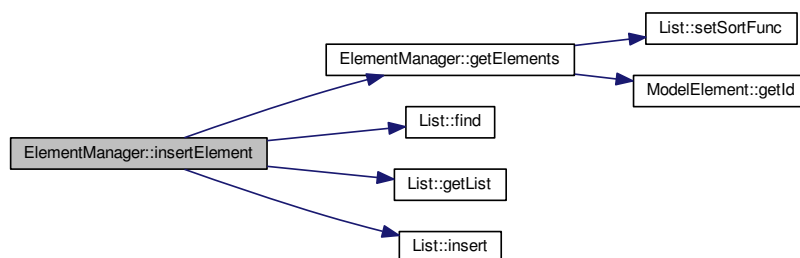
#### 5.15.3.4 `std::list< std::string > * ElementManager::getElementTypenames ( ) const`

Here is the caller graph for this function:



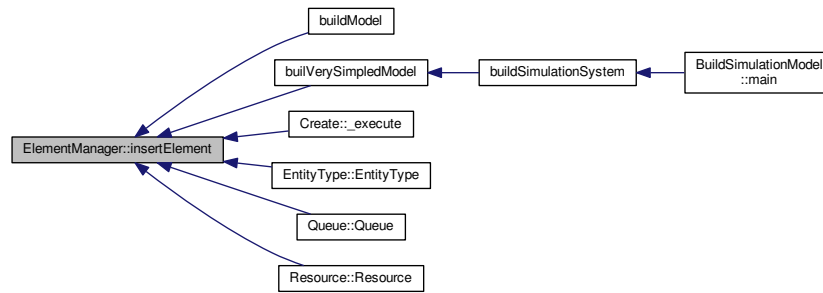
#### 5.15.3.5 `bool ElementManager::insertElement ( std::string infraTypename, ModelElement * infra )`

Here is the call graph for this function:



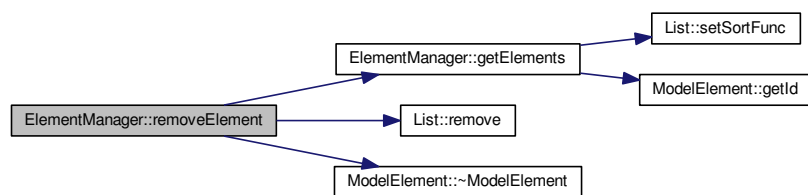


Here is the caller graph for this function:

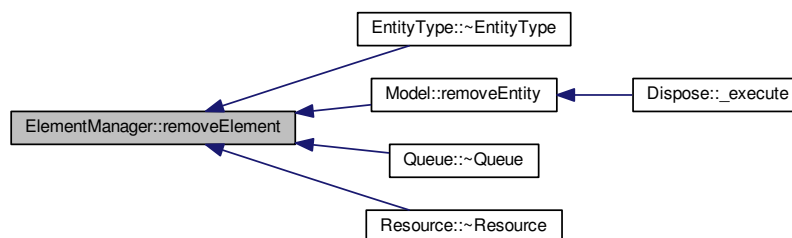


#### 5.15.3.6 void ElementManager::removeElement ( std::string infraTypename, ModelElement \* infra )

Here is the call graph for this function:

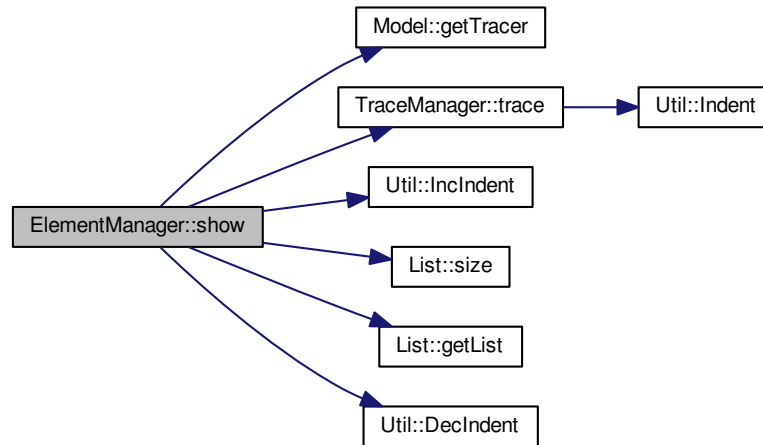


Here is the caller graph for this function:



#### 5.15.3.7 void ElementManager::show ( )

Here is the call graph for this function:



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

- [ElementManager.h](#)
- [ElementManager.cpp](#)

## 5.16 ElementManager\_if Class Reference

```
#include <ElementManager_if.h>
```

### Public Member Functions

- [ElementManager\\_if](#) ( )
- [ElementManager\\_if](#) (const [ElementManager\\_if](#) &orig)
- virtual [~ElementManager\\_if](#) ( )

### 5.16.1 Constructor & Destructor Documentation

5.16.1.1 [ElementManager\\_if::ElementManager\\_if](#) ( )

5.16.1.2 [ElementManager\\_if::ElementManager\\_if](#) ( const [ElementManager\\_if](#) & orig )

5.16.1.3 virtual [ElementManager\\_if::~ElementManager\\_if](#) ( ) [virtual]

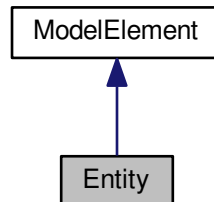
The documentation for this class was generated from the following file:

- [ElementManager\\_if.h](#)

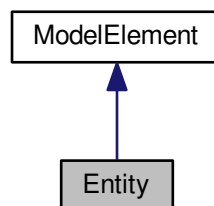
## 5.17 Entity Class Reference

```
#include <Entity.h>
```

Inheritance diagram for Entity:



Collaboration diagram for Entity:



### Public Member Functions

- [Entity](#) ()
- [Entity](#) (const [Entity](#) &orig)
- virtual [~Entity](#) ()
- virtual std::string [show](#) ()
- void [setEntityType](#) ([EntityType](#) \*entityType)
- [EntityType](#) \* [getEntityType](#) () const
- double [getAttributeValue](#) (std::string attributeName)
- void [setAttributeValue](#) (std::string attributeName, double value)

### Protected Member Functions

- virtual void [\\_loadInstance](#) (std::list< std::string > words)
- virtual std::list< std::string > \* [\\_saveInstance](#) ()
- virtual bool [\\_verifySymbols](#) (std::string \*errorMessage)

## Additional Inherited Members

### 5.17.1 Constructor & Destructor Documentation

5.17.1.1 `Entity::Entity ( )`

5.17.1.2 `Entity::Entity ( const Entity & orig )`

5.17.1.3 `Entity::~Entity ( )` `[virtual]`

### 5.17.2 Member Function Documentation

5.17.2.1 `void Entity::_loadInstance ( std::list< std::string > words )` `[protected]`, `[virtual]`

Implements [ModelElement](#).

5.17.2.2 `std::list< std::string > * Entity::_saveInstance ( )` `[protected]`, `[virtual]`

Reimplemented from [ModelElement](#).

Here is the call graph for this function:



5.17.2.3 `bool Entity::_verifySymbols ( std::string * errorMessage )` `[protected]`, `[virtual]`

Implements [ModelElement](#).

5.17.2.4 `double Entity::getAttributeValue ( std::string attributeName )`

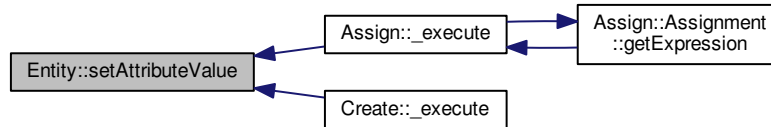
5.17.2.5 `EntityType * Entity::getEntityType ( )` `const`

Here is the caller graph for this function:



#### 5.17.2.6 void Entity::setAttributeValue ( std::string *attributeName*, double *value* )

Here is the caller graph for this function:



#### 5.17.2.7 void Entity::setEntityType ( EntityType \* *entityType* )

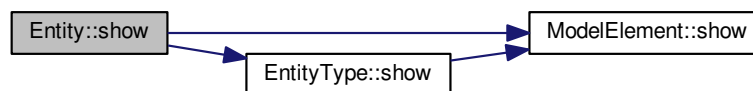
Here is the caller graph for this function:



#### 5.17.2.8 std::string Entity::show ( ) [virtual]

Reimplemented from [ModelElement](#).

Here is the call graph for this function:



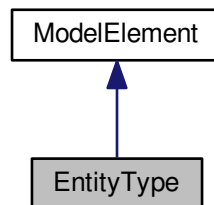
The documentation for this class was generated from the following files:

- [Entity.h](#)
- [Entity.cpp](#)

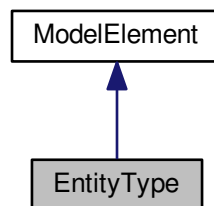
## 5.18 EntityType Class Reference

```
#include <EntityType.h>
```

Inheritance diagram for EntityType:



Collaboration diagram for EntityType:



### Public Member Functions

- [EntityType](#) ([ElementManager](#) \*elemManager)
- [EntityType](#) ([ElementManager](#) \*elemManager, std::string name)
- [EntityType](#) ([ElementManager](#) \*elemManager, std::string name, std::string initialPicture, double initialWaitingCost, double initialVACost, double initialNVACost, double initialOtherCost)
- [EntityType](#) (const [EntityType](#) &orig)
- virtual [~EntityType](#) ()
- virtual std::string [show](#) ()
- void [setInitialWaitingCost](#) (double \_initialWaitingCost)
- double [getInitialWaitingCost](#) () const
- void [setInitialOtherCost](#) (double \_initialOtherCost)
- double [getInitialOtherCost](#) () const
- void [setInitialNVACost](#) (double \_initialNVACost)
- double [getInitialNVACost](#) () const
- void [setInitialVACost](#) (double \_initialVACost)

- double [getInitialVACost](#) () const
- void [setInitialPicture](#) (std::string \_initialPicture)
- std::string [getInitialPicture](#) () const
- [StatisticsCollector](#) \* [getCstatTimeInSystem](#) () const
- [StatisticsCollector](#) \* [getCstatNVATime](#) () const
- [StatisticsCollector](#) \* [getCstatVATime](#) () const
- [StatisticsCollector](#) \* [getCstatOtherTime](#) () const
- [StatisticsCollector](#) \* [getCstatTransferTime](#) () const
- [StatisticsCollector](#) \* [getCstatWaitingTime](#) () const

### Protected Member Functions

- virtual void [\\_loadInstance](#) (std::list< std::string > words)
- virtual std::list< std::string > \* [\\_saveInstance](#) ()
- virtual bool [\\_verifySymbols](#) (std::string \*errorMessage)

### Additional Inherited Members

#### 5.18.1 Constructor & Destructor Documentation

##### 5.18.1.1 EntityType::EntityType ( ElementManager \* *elemManager* )

Here is the caller graph for this function:



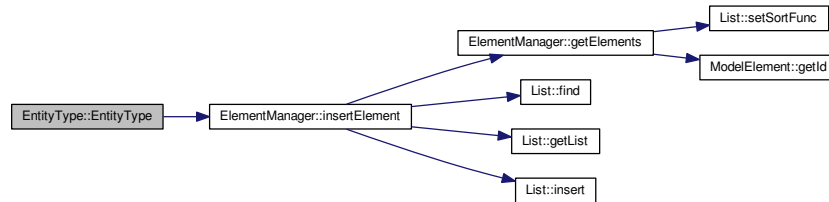
##### 5.18.1.2 EntityType::EntityType ( ElementManager \* *elemManager*, std::string *name* )

Here is the call graph for this function:



5.18.1.3 `EntityType::EntityType ( ElementManager * elemManager, std::string name, std::string initialPicture, double initialWaitingCost, double initialVACost, double initialNVACost, double initialOtherCost )`

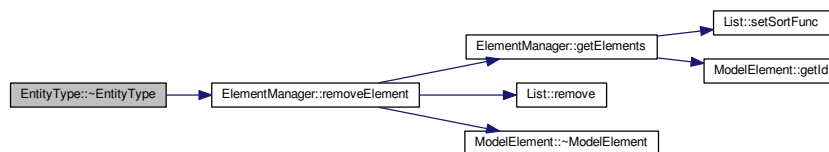
Here is the call graph for this function:



5.18.1.4 `EntityType::EntityType ( const EntityType & orig )`

5.18.1.5 `EntityType::~EntityType ( ) [virtual]`

Here is the call graph for this function:



## 5.18.2 Member Function Documentation

5.18.2.1 `void EntityType::_loadInstance ( std::list< std::string > words ) [protected], [virtual]`

Implements [ModelElement](#).

5.18.2.2 `std::list< std::string > * EntityType::_saveInstance ( ) [protected], [virtual]`

Reimplemented from [ModelElement](#).

Here is the call graph for this function:





5.18.2.3 `bool EntityType::_verifySymbols ( std::string * errorMessage )` `[protected]`, `[virtual]`

Implements [ModelElement](#).

5.18.2.4 `StatisticsCollector * EntityType::getCstatNVATime ( ) const`

5.18.2.5 `StatisticsCollector * EntityType::getCstatOtherTime ( ) const`

5.18.2.6 `StatisticsCollector * EntityType::getCstatTimeInSystem ( ) const`

5.18.2.7 `StatisticsCollector * EntityType::getCstatTransferTime ( ) const`

5.18.2.8 `StatisticsCollector * EntityType::getCstatVATime ( ) const`

5.18.2.9 `StatisticsCollector * EntityType::getCstatWaitingTime ( ) const`

5.18.2.10 `double EntityType::getInitialNVACost ( ) const`

5.18.2.11 `double EntityType::getInitialOtherCost ( ) const`

5.18.2.12 `std::string EntityType::getInitialPicture ( ) const`

5.18.2.13 `double EntityType::getInitialVACost ( ) const`

5.18.2.14 `double EntityType::getInitialWaitingCost ( ) const`

5.18.2.15 `void EntityType::setInitialNVACost ( double _initialNVACost )`

5.18.2.16 `void EntityType::setInitialOtherCost ( double _initialOtherCost )`

5.18.2.17 `void EntityType::setInitialPicture ( std::string _initialPicture )`

5.18.2.18 `void EntityType::setInitialVACost ( double _initialVACost )`

5.18.2.19 `void EntityType::setInitialWaitingCost ( double _initialWaitingCost )`

5.18.2.20 `std::string EntityType::show ( )` `[virtual]`

Reimplemented from [ModelElement](#).

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:

- [EntityType.h](#)
- [EntityType.cpp](#)

## 5.19 Event Class Reference

```
#include <Event.h>
```

### Public Member Functions

- [Event](#) (double *time*, [Entity](#) \**entity*, [ModelComponent](#) \**component*)
- [Event](#) (const [Event](#) &*orig*)
- virtual [~Event](#) ()
- double [getTime](#) () const
- [ModelComponent](#) \* [getComponent](#) () const
- [Entity](#) \* [getEntity](#) () const
- std::string [show](#) ()

### 5.19.1 Constructor & Destructor Documentation

5.19.1.1 `Event::Event ( double time, Entity * entity, ModelComponent * component )`

5.19.1.2 `Event::Event ( const Event & orig )`

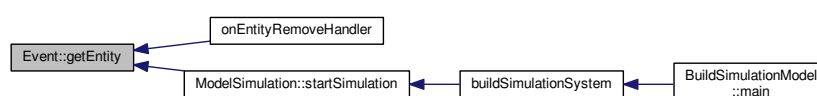
5.19.1.3 `Event::~~Event ( )` [virtual]

### 5.19.2 Member Function Documentation

5.19.2.1 `ModelComponent * Event::getComponent ( ) const`

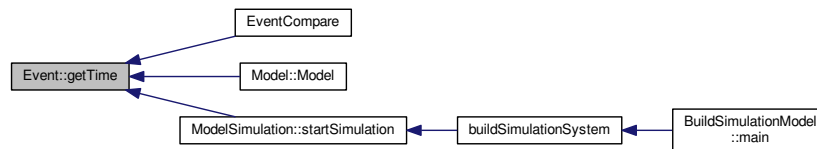
5.19.2.2 `Entity * Event::getEntity ( ) const`

Here is the caller graph for this function:



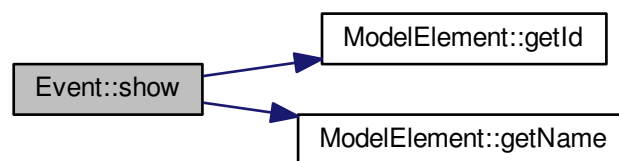
## 5.19.2.3 double Event::getTime ( ) const

Here is the caller graph for this function:

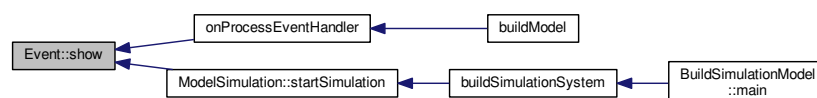


## 5.19.2.4 std::string Event::show ( )

Here is the call graph for this function:



Here is the caller graph for this function:



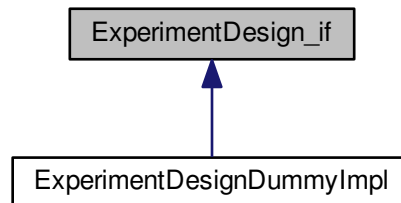
The documentation for this class was generated from the following files:

- [Event.h](#)
- [Event.cpp](#)

## 5.20 ExperimentDesign\_if Class Reference

```
#include <ExperimentDesign_if.h>
```

Inheritance diagram for ExperimentDesign\_if:



### Public Member Functions

- virtual [ProcessAnalyser\\_if](#) \* [getProcessAnalyser](#) () const =0
- virtual bool [generate2krScenarioExperiments](#) ()=0
- virtual bool [calculateContributionAndCoefficients](#) ()=0
- virtual std::list< [FactorOrInteractionContribution](#) \* > \* [getContributions](#) () const =0

### 5.20.1 Detailed Description

It designs a set of experiments ([SimulationScenario](#)) where que level of factors ([SimulationControl](#)) are set automatically to create a  $2^k$  experiment design, and where the contributions of the factors and their interactions (just a set of [SimulationControl](#)) can be obtained.

### 5.20.2 Member Function Documentation

5.20.2.1 virtual bool [ExperimentDesign\\_if::calculateContributionAndCoefficients](#) ( ) [pure virtual]

Implemented in [ExperimentDesignDummyImpl](#).

5.20.2.2 virtual bool [ExperimentDesign\\_if::generate2krScenarioExperiments](#) ( ) [pure virtual]

Implemented in [ExperimentDesignDummyImpl](#).

5.20.2.3 virtual std::list<[FactorOrInteractionContribution](#)\*> \* [ExperimentDesign\\_if::getContributions](#) ( ) const [pure virtual]

Implemented in [ExperimentDesignDummyImpl](#).

5.20.2.4 `virtual ProcessAnalyser_if* ExperimentDesign_if::getProcessAnalyser ( ) const` [pure virtual]

Implemented in [ExperimentDesignDummyImpl](#).

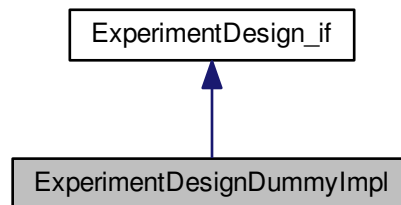
The documentation for this class was generated from the following file:

- [ExperimentDesign\\_if.h](#)

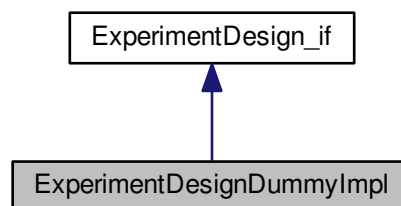
## 5.21 ExperimentDesignDummyImpl Class Reference

```
#include <ExperimentDesignDummyImpl.h>
```

Inheritance diagram for ExperimentDesignDummyImpl:



Collaboration diagram for ExperimentDesignDummyImpl:



### Public Member Functions

- [ExperimentDesignDummyImpl](#) ()
- [ExperimentDesignDummyImpl](#) (const [ExperimentDesignDummyImpl](#) &orig)
- `virtual ~ExperimentDesignDummyImpl` ()
- `ProcessAnalyser_if * getProcessAnalyser` () const
- `bool generate2krScenarioExperiments` ()
- `bool calculateContributionAndCoefficients` ()
- `std::list< FactorOrInteractionContribution * > * getContributions` () const

### 5.21.1 Constructor & Destructor Documentation

5.21.1.1 `ExperimentDesignDummyImpl::ExperimentDesignDummyImpl ( )`

5.21.1.2 `ExperimentDesignDummyImpl::ExperimentDesignDummyImpl ( const ExperimentDesignDummyImpl & orig )`

5.21.1.3 `ExperimentDesignDummyImpl::~~ExperimentDesignDummyImpl ( )` [virtual]

### 5.21.2 Member Function Documentation

5.21.2.1 `bool ExperimentDesignDummyImpl::calculateContributionAndCoefficients ( )` [virtual]

Implements [ExperimentDesign\\_if](#).

5.21.2.2 `bool ExperimentDesignDummyImpl::generate2krScenarioExperiments ( )` [virtual]

Implements [ExperimentDesign\\_if](#).

5.21.2.3 `std::list< FactorOrInteractionContribution * > * ExperimentDesignDummyImpl::getContributions ( ) const` [virtual]

Implements [ExperimentDesign\\_if](#).

5.21.2.4 `ProcessAnalyser_if * ExperimentDesignDummyImpl::getProcessAnalyser ( ) const` [virtual]

Implements [ExperimentDesign\\_if](#).

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

- [ExperimentDesignDummyImpl.h](#)
- [ExperimentDesignDummyImpl.cpp](#)

## 5.22 FactorOrInteractionContribution Class Reference

```
#include <FactorOrInteractionContribution.h>
```

### Public Member Functions

- [FactorOrInteractionContribution](#) (double contribution, double modelCoefficient, std::list< [SimulationControl](#) \* > \*controls)
- [FactorOrInteractionContribution](#) (const [FactorOrInteractionContribution](#) &orig)
- [~FactorOrInteractionContribution](#) ()
- double [getModelCoefficient](#) () const
- std::list< [SimulationControl](#) \* > \* [getControls](#) () const
- double [getContribution](#) () const

### 5.22.1 Detailed Description

This simple class corresponds to a factor when it refers to just one [SimulationControl](#), or to the interaction between two or more factors when it refers to more [SimulationControl](#). It also encapsulates the contribution of the factor or interaction and its coefficient in the full model that estimates one specific [SimulationResponse](#).

### 5.22.2 Constructor & Destructor Documentation

5.22.2.1 `FactorOrInteractionContribution::FactorOrInteractionContribution ( double contribution, double modelCoefficient, std::list< SimulationControl * > * controls )`

5.22.2.2 `FactorOrInteractionContribution::FactorOrInteractionContribution ( const FactorOrInteractionContribution & orig )`

5.22.2.3 `FactorOrInteractionContribution::~~FactorOrInteractionContribution ( )`

### 5.22.3 Member Function Documentation

5.22.3.1 `double FactorOrInteractionContribution::getContribution ( ) const`

5.22.3.2 `std::list< SimulationControl * > * FactorOrInteractionContribution::getControls ( ) const`

5.22.3.3 `double FactorOrInteractionContribution::getModelCoefficient ( ) const`

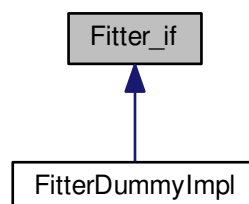
The documentation for this class was generated from the following files:

- [FactorOrInteractionContribution.h](#)
- [FactorOrInteractionContribution.cpp](#)

## 5.23 Fitter\_if Class Reference

```
#include <Fitter_if.h>
```

Inheritance diagram for `Fitter_if`:



## Public Member Functions

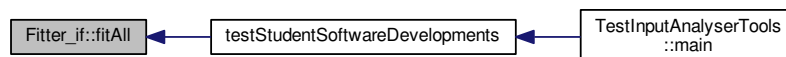
- virtual bool [isNormalDistributed](#) (double confidencelevel)=0
- virtual void [fitUniform](#) (double \*sqerror, double \*min, double \*max)=0
- virtual void [fitTriangular](#) (double \*sqerror, double \*min, double \*mo, double \*max)=0
- virtual void [fitNormal](#) (double \*sqerror, double \*avg, double \*stddev)=0
- virtual void [fitExpo](#) (double \*sqerror, double \*avg1)=0
- virtual void [fitErlang](#) (double \*sqerror, double \*avg, double \*m)=0
- virtual void [fitBeta](#) (double \*sqerror, double \*alpha, double \*beta, double \*infLimit, double \*supLimit)=0
- virtual void [fitWeibull](#) (double \*sqerror, double \*alpha, double \*scale)=0
- virtual void [fitAll](#) (double \*sqerror, std::string \*name)=0
- virtual void [setDataFilename](#) (std::string dataFilename)=0
- virtual std::string [getDataFilename](#) ()=0

### 5.23.1 Member Function Documentation

5.23.1.1 virtual void Fitter\_if::fitAll ( double \* *sqerror*, std::string \* *name* ) [pure virtual]

Implemented in [FitterDummyImpl](#).

Here is the caller graph for this function:



5.23.1.2 virtual void Fitter\_if::fitBeta ( double \* *sqerror*, double \* *alpha*, double \* *beta*, double \* *infLimit*, double \* *supLimit* ) [pure virtual]

Implemented in [FitterDummyImpl](#).

5.23.1.3 virtual void Fitter\_if::fitErlang ( double \* *sqerror*, double \* *avg*, double \* *m* ) [pure virtual]

Implemented in [FitterDummyImpl](#).

5.23.1.4 virtual void Fitter\_if::fitExpo ( double \* *sqerror*, double \* *avg1* ) [pure virtual]

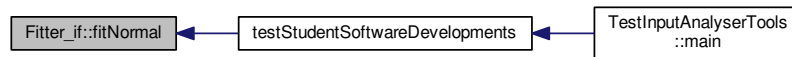
Implemented in [FitterDummyImpl](#).



5.23.1.5 `virtual void Fitter_if::fitNormal ( double * sqerror, double * avg, double * stddev )` [pure virtual]

Implemented in [FitterDummyImpl](#).

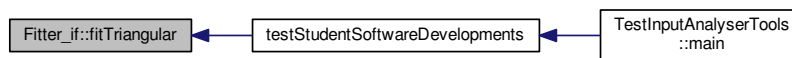
Here is the caller graph for this function:



5.23.1.6 `virtual void Fitter_if::fitTriangular ( double * sqerror, double * min, double * mo, double * max )` [pure virtual]

Implemented in [FitterDummyImpl](#).

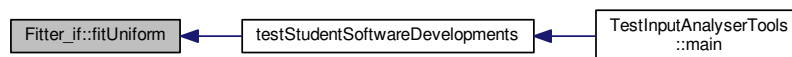
Here is the caller graph for this function:



5.23.1.7 `virtual void Fitter_if::fitUniform ( double * sqerror, double * min, double * max )` [pure virtual]

Implemented in [FitterDummyImpl](#).

Here is the caller graph for this function:



5.23.1.8 `virtual void Fitter_if::fitWeibull ( double * sqerror, double * alpha, double * scale )` [pure virtual]

Implemented in [FitterDummyImpl](#).

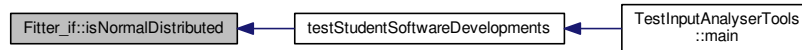
5.23.1.9 `virtual std::string Fitter_if::getDataFilename ( )` [pure virtual]

Implemented in [FitterDummyImpl](#).

5.23.1.10 `virtual bool Fitter_if::isNormalDistributed ( double confidencelevel )` [pure virtual]

Implemented in [FitterDummyImpl](#).

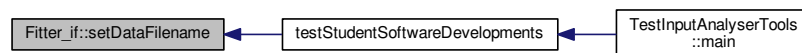
Here is the caller graph for this function:



5.23.1.11 `virtual void Fitter_if::setDataFilename ( std::string dataFilename )` [pure virtual]

Implemented in [FitterDummyImpl](#).

Here is the caller graph for this function:



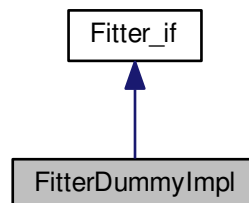
The documentation for this class was generated from the following file:

- [Fitter\\_if.h](#)

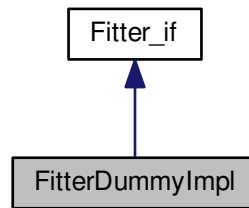
## 5.24 FitterDummyImpl Class Reference

```
#include <FitterDummyImpl.h>
```

Inheritance diagram for FitterDummyImpl:



Collaboration diagram for FitterDummyImpl:



## Public Member Functions

- [FitterDummyImpl](#) ()
- [FitterDummyImpl](#) (const [FitterDummyImpl](#) &orig)
- [~FitterDummyImpl](#) ()
- bool [isNormalDistributed](#) (double confidencelevel)
- void [fitUniform](#) (double \*sqerror, double \*min, double \*max)
- void [fitTriangular](#) (double \*sqerror, double \*min, double \*mo, double \*max)
- void [fitNormal](#) (double \*sqerror, double \*avg, double \*stddev)
- void [fitExpo](#) (double \*sqerror, double \*avg1)
- void [fitErlang](#) (double \*sqerror, double \*avg, double \*m)
- void [fitBeta](#) (double \*sqerror, double \*alpha, double \*beta, double \*infLimit, double \*supLimit)
- void [fitWeibull](#) (double \*sqerror, double \*alpha, double \*scale)
- void [fitAll](#) (double \*sqerror, std::string \*name)
- void [setDataFilename](#) (std::string dataFilename)
- std::string [getDataFilename](#) ()

## 5.24.1 Constructor & Destructor Documentation

5.24.1.1 [FitterDummyImpl::FitterDummyImpl](#) ( )

5.24.1.2 [FitterDummyImpl::FitterDummyImpl](#) ( const [FitterDummyImpl](#) & orig )

5.24.1.3 [FitterDummyImpl::~~FitterDummyImpl](#) ( )

## 5.24.2 Member Function Documentation

5.24.2.1 void [FitterDummyImpl::fitAll](#) ( double \* *sqerror*, std::string \* *name* ) [virtual]

Implements [Fitter\\_if](#).

5.24.2.2 void FitterDummyImpl::fitBeta ( double \* *sqrrerror*, double \* *alpha*, double \* *beta*, double \* *infLimit*, double \* *supLimit* ) [virtual]

Implements [Fitter\\_if](#).

5.24.2.3 void FitterDummyImpl::fitErlang ( double \* *sqrrerror*, double \* *avg*, double \* *m* ) [virtual]

Implements [Fitter\\_if](#).

5.24.2.4 void FitterDummyImpl::fitExpo ( double \* *sqrrerror*, double \* *avg1* ) [virtual]

Implements [Fitter\\_if](#).

5.24.2.5 void FitterDummyImpl::fitNormal ( double \* *sqrrerror*, double \* *avg*, double \* *stddev* ) [virtual]

Implements [Fitter\\_if](#).

5.24.2.6 void FitterDummyImpl::fitTriangular ( double \* *sqrrerror*, double \* *min*, double \* *mo*, double \* *max* ) [virtual]

Implements [Fitter\\_if](#).

5.24.2.7 void FitterDummyImpl::fitUniform ( double \* *sqrrerror*, double \* *min*, double \* *max* ) [virtual]

Implements [Fitter\\_if](#).

5.24.2.8 void FitterDummyImpl::fitWeibull ( double \* *sqrrerror*, double \* *alpha*, double \* *scale* ) [virtual]

Implements [Fitter\\_if](#).

5.24.2.9 std::string FitterDummyImpl::getDataFilename ( ) [virtual]

Implements [Fitter\\_if](#).

5.24.2.10 bool FitterDummyImpl::isNormalDistributed ( double *confidencelevel* ) [virtual]

Implements [Fitter\\_if](#).

5.24.2.11 void FitterDummyImpl::setDataFilename ( std::string *dataFilename* ) [virtual]

Implements [Fitter\\_if](#).

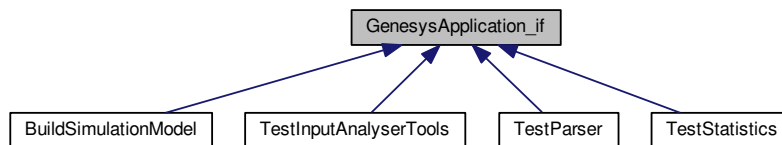
The documentation for this class was generated from the following files:

- [FitterDummyImpl.h](#)
- [FitterDummyImpl.cpp](#)

## 5.25 GenesysApplication\_if Class Reference

```
#include <GenesysApplication_if.h>
```

Inheritance diagram for GenesysApplication\_if:



### Public Member Functions

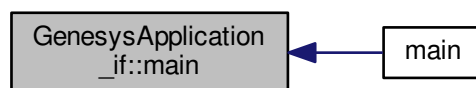
- virtual int [main](#) (int argc, char \*\*argv)=0

#### 5.25.1 Member Function Documentation

5.25.1.1 virtual int GenesysApplication\_if::main ( int *argc*, char \*\* *argv* ) [pure virtual]

Implemented in [TestInputAnalyserTools](#), [TestParser](#), [BuildSimulationModel](#), and [TestStatistics](#).

Here is the caller graph for this function:



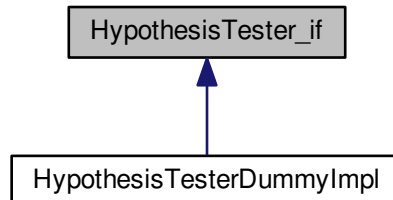
The documentation for this class was generated from the following file:

- [GenesysApplication\\_if.h](#)

## 5.26 HypothesisTester\_if Class Reference

```
#include <HypothesisTester_if.h>
```

Inheritance diagram for HypothesisTester\_if:



### Public Types

- enum [H1Comparition](#) { [DIFFERENT](#) = 1, [LESS\\_THAN](#) = 2, [GREATER\\_THAN](#) = 3 }

### Public Member Functions

- virtual double [testAverage](#) (double confidencelevel, double avg, [H1Comparition](#) comp)=0
- virtual double [testProportion](#) (double confidencelevel, double prop, [H1Comparition](#) comp)=0
- virtual double [testVariance](#) (double confidencelevel, double var, [H1Comparition](#) comp)=0
- virtual double [testAverage](#) (double confidencelevel, std::string secondPopulationDataFilename, [H1Comparition](#) comp)=0
- virtual double [testProportion](#) (double confidencelevel, std::string secondPopulationDataFilename, [H1Comparition](#) comp)=0
- virtual double [testVariance](#) (double confidencelevel, std::string secondPopulationDataFilename, [H1Comparition](#) comp)=0
- virtual void [setDataFilename](#) (std::string dataFilename)=0
- virtual std::string [getDataFilename](#) ()=0

### 5.26.1 Detailed Description

Interface for parametric hypothesis tests based on a datafile.

### 5.26.2 Member Enumeration Documentation

#### 5.26.2.1 enum HypothesisTester\_if::H1Comparition

Enumerator

***DIFFERENT***  
***LESS\_THAN***  
***GREATER\_THAN***

### 5.26.3 Member Function Documentation

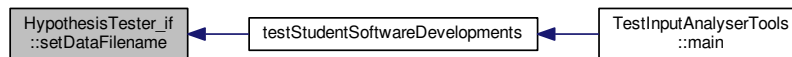
5.26.3.1 `virtual std::string HypothesisTester_if::getDataFilename ( ) [pure virtual]`

Implemented in [HypothesisTesterDummyImpl](#).

5.26.3.2 `virtual void HypothesisTester_if::setDataFilename ( std::string dataFilename ) [pure virtual]`

Implemented in [HypothesisTesterDummyImpl](#).

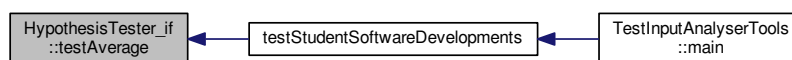
Here is the caller graph for this function:



5.26.3.3 `virtual double HypothesisTester_if::testAverage ( double confidencelevel, double avg, H1Comparition comp ) [pure virtual]`

Implemented in [HypothesisTesterDummyImpl](#).

Here is the caller graph for this function:



5.26.3.4 `virtual double HypothesisTester_if::testAverage ( double confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp ) [pure virtual]`

Implemented in [HypothesisTesterDummyImpl](#).

5.26.3.5 `virtual double HypothesisTester_if::testProportion ( double confidencelevel, double prop, H1Comparition comp ) [pure virtual]`

Implemented in [HypothesisTesterDummyImpl](#).

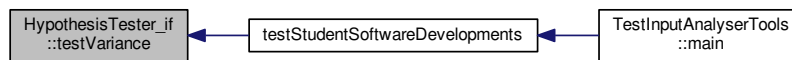
5.26.3.6 virtual double HypothesisTester\_if::testProportion ( double *confidencelevel*, std::string *secondPopulationDataFilename*, H1Comparition *comp* ) [pure virtual]

Implemented in [HypothesisTesterDummyImpl](#).

5.26.3.7 virtual double HypothesisTester\_if::testVariance ( double *confidencelevel*, double *var*, H1Comparition *comp* ) [pure virtual]

Implemented in [HypothesisTesterDummyImpl](#).

Here is the caller graph for this function:



5.26.3.8 virtual double HypothesisTester\_if::testVariance ( double *confidencelevel*, std::string *secondPopulationDataFilename*, H1Comparition *comp* ) [pure virtual]

Implemented in [HypothesisTesterDummyImpl](#).

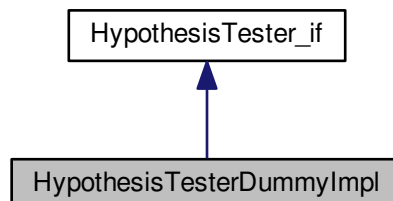
The documentation for this class was generated from the following file:

- [HypothesisTester\\_if.h](#)

## 5.27 HypothesisTesterDummyImpl Class Reference

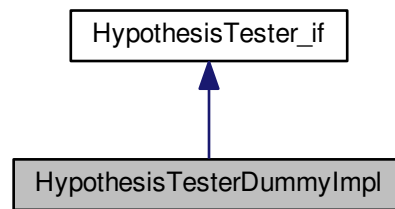
```
#include <HypothesisTesterDummyImpl.h>
```

Inheritance diagram for HypothesisTesterDummyImpl:





Collaboration diagram for HypothesisTesterDummyImpl:



## Public Member Functions

- [HypothesisTesterDummyImpl](#) ()
- [HypothesisTesterDummyImpl](#) (const [HypothesisTesterDummyImpl](#) &orig)
- [~HypothesisTesterDummyImpl](#) ()
- double [testAverage](#) (double confidencelevel, double avg, [H1Comparison](#) comp)
- double [testProportion](#) (double confidencelevel, double prop, [H1Comparison](#) comp)
- double [testVariance](#) (double confidencelevel, double var, [H1Comparison](#) comp)
- double [testAverage](#) (double confidencelevel, std::string secondPopulationDataFilename, [H1Comparison](#) comp)
- double [testProportion](#) (double confidencelevel, std::string secondPopulationDataFilename, [H1Comparison](#) comp)
- double [testVariance](#) (double confidencelevel, std::string secondPopulationDataFilename, [H1Comparison](#) comp)
- void [setDataFilename](#) (std::string dataFilename)
- std::string [getDataFilename](#) ()

## Additional Inherited Members

### 5.27.1 Constructor & Destructor Documentation

5.27.1.1 `HypothesisTesterDummyImpl::HypothesisTesterDummyImpl ( )`

5.27.1.2 `HypothesisTesterDummyImpl::HypothesisTesterDummyImpl ( const HypothesisTesterDummyImpl & orig )`

5.27.1.3 `HypothesisTesterDummyImpl::~~HypothesisTesterDummyImpl ( )`

### 5.27.2 Member Function Documentation

5.27.2.1 `std::string HypothesisTesterDummyImpl::getDataFilename ( )` [virtual]

Implements [HypothesisTester\\_if](#).

5.27.2.2 `void HypothesisTesterDummyImpl::setDataFilename ( std::string dataFilename ) [virtual]`

Implements [HypothesisTester\\_if](#).

5.27.2.3 `double HypothesisTesterDummyImpl::testAverage ( double confidencelevel, double avg, H1Comparition comp ) [virtual]`

Implements [HypothesisTester\\_if](#).

5.27.2.4 `double HypothesisTesterDummyImpl::testAverage ( double confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp ) [virtual]`

Implements [HypothesisTester\\_if](#).

5.27.2.5 `double HypothesisTesterDummyImpl::testProportion ( double confidencelevel, double prop, H1Comparition comp ) [virtual]`

Implements [HypothesisTester\\_if](#).

5.27.2.6 `double HypothesisTesterDummyImpl::testProportion ( double confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp ) [virtual]`

Implements [HypothesisTester\\_if](#).

5.27.2.7 `double HypothesisTesterDummyImpl::testVariance ( double confidencelevel, double var, H1Comparition comp ) [virtual]`

Implements [HypothesisTester\\_if](#).

5.27.2.8 `double HypothesisTesterDummyImpl::testVariance ( double confidencelevel, std::string secondPopulationDataFilename, H1Comparition comp ) [virtual]`

Implements [HypothesisTester\\_if](#).

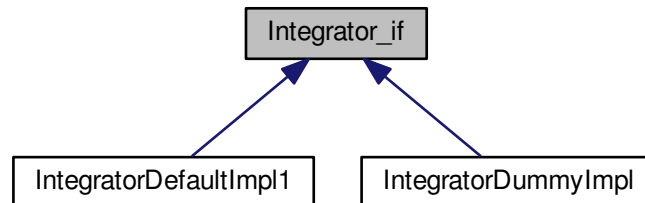
The documentation for this class was generated from the following files:

- [HypothesisTesterDummyImpl.h](#)
- [HypothesisTesterDummyImpl.cpp](#)

## 5.28 Integrator\_if Class Reference

```
#include <Integrator_if.h>
```

Inheritance diagram for Integrator\_if:



### Public Member Functions

- virtual void [setPrecision](#) (double e)=0
- virtual double [getPrecision](#) ()=0
- virtual double [integrate](#) (double min, double max, double(\*f)(double, double), double p2)=0
- virtual double [integrate](#) (double min, double max, double(\*f)(double, double, double), double p2, double p3)=0
- virtual double [integrate](#) (double min, double max, double(\*f)(double, double, double, double), double p2, double p3, double p4)=0
- virtual double [integrate](#) (double min, double max, double(\*f)(double, double, double, double, double), double p2, double p3, double p4, double p5)=0

### 5.28.1 Member Function Documentation

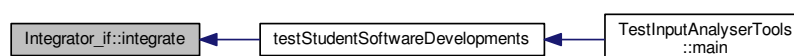
5.28.1.1 virtual double Integrator\_if::getPrecision ( ) [pure virtual]

Implemented in [IntegratorDefaultImpl1](#), and [IntegratorDummyImpl](#).

5.28.1.2 virtual double Integrator\_if::integrate ( double *min*, double *max*, double(\*f)(double, double) *f*, double *p2* ) [pure virtual]

Implemented in [IntegratorDefaultImpl1](#), and [IntegratorDummyImpl](#).

Here is the caller graph for this function:



5.28.1.3 `virtual double Integrator_if::integrate ( double min, double max, double(*) (double, double, double) f, double p2, double p3 ) [pure virtual]`

Implemented in [IntegratorDefaultImpl1](#), and [IntegratorDummyImpl](#).

5.28.1.4 `virtual double Integrator_if::integrate ( double min, double max, double(*) (double, double, double, double) f, double p2, double p3, double p4 ) [pure virtual]`

Implemented in [IntegratorDefaultImpl1](#), and [IntegratorDummyImpl](#).

5.28.1.5 `virtual double Integrator_if::integrate ( double min, double max, double(*) (double, double, double, double, double) f, double p2, double p3, double p4, double p5 ) [pure virtual]`

Implemented in [IntegratorDefaultImpl1](#), and [IntegratorDummyImpl](#).

5.28.1.6 `virtual void Integrator_if::setPrecision ( double e ) [pure virtual]`

Implemented in [IntegratorDefaultImpl1](#), and [IntegratorDummyImpl](#).

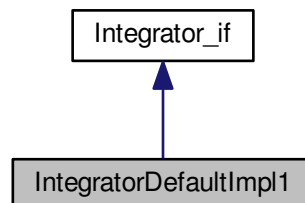
The documentation for this class was generated from the following file:

- [Integrator\\_if.h](#)

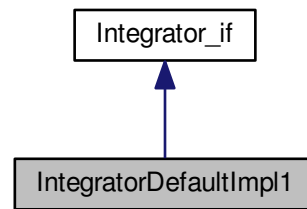
## 5.29 IntegratorDefaultImpl1 Class Reference

```
#include <IntegratorDefaultImpl1.h>
```

Inheritance diagram for IntegratorDefaultImpl1:



Collaboration diagram for IntegratorDefaultImpl1:



## Public Member Functions

- [IntegratorDefaultImpl1](#) ( )
- [IntegratorDefaultImpl1](#) (const [IntegratorDefaultImpl1](#) &orig)
- virtual [~IntegratorDefaultImpl1](#) ( )
- virtual void [setPrecision](#) (double e)
- virtual double [getPrecision](#) ( )
- virtual double [integrate](#) (double min, double max, double(\*f)(double, double), double p2)
- virtual double [integrate](#) (double min, double max, double(\*f)(double, double, double), double p2, double p3)
- virtual double [integrate](#) (double min, double max, double(\*f)(double, double, double, double), double p2, double p3, double p4)
- virtual double [integrate](#) (double min, double max, double(\*f)(double, double, double, double, double), double p2, double p3, double p4, double p5)

## 5.29.1 Constructor & Destructor Documentation

5.29.1.1 [IntegratorDefaultImpl1::IntegratorDefaultImpl1](#) ( )

5.29.1.2 [IntegratorDefaultImpl1::IntegratorDefaultImpl1](#) ( const [IntegratorDefaultImpl1](#) & orig )

5.29.1.3 [IntegratorDefaultImpl1::~~IntegratorDefaultImpl1](#) ( ) [virtual]

## 5.29.2 Member Function Documentation

5.29.2.1 [double IntegratorDefaultImpl1::getPrecision](#) ( ) [virtual]

Implements [Integrator\\_if](#).

5.29.2.2 [double IntegratorDefaultImpl1::integrate](#) ( double *min*, double *max*, double(\*f)(double, double) *f*, double *p2* ) [virtual]

Implements [Integrator\\_if](#).

5.29.2.3 `double IntegratorDefaultImpl1::integrate ( double min, double max, double(*) (double, double, double) f, double p2, double p3 ) [virtual]`

Implements [Integrator\\_if](#).

5.29.2.4 `double IntegratorDefaultImpl1::integrate ( double min, double max, double(*) (double, double, double, double) f, double p2, double p3, double p4 ) [virtual]`

Implements [Integrator\\_if](#).

5.29.2.5 `double IntegratorDefaultImpl1::integrate ( double min, double max, double(*) (double, double, double, double, double) f, double p2, double p3, double p4, double p5 ) [virtual]`

Implements [Integrator\\_if](#).

5.29.2.6 `void IntegratorDefaultImpl1::setPrecision ( double e ) [virtual]`

Implements [Integrator\\_if](#).

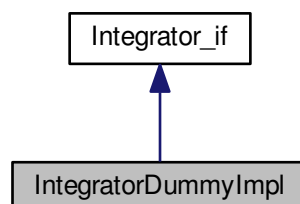
The documentation for this class was generated from the following files:

- [IntegratorDefaultImpl1.h](#)
- [IntegratorDefaultImpl1.cpp](#)

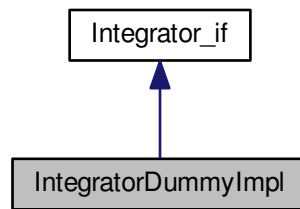
## 5.30 IntegratorDummyImpl Class Reference

```
#include <IntegratorDummyImpl.h>
```

Inheritance diagram for IntegratorDummyImpl:



Collaboration diagram for IntegratorDummyImpl:



## Public Member Functions

- [IntegratorDummyImpl](#) ()
- [IntegratorDummyImpl](#) (const [IntegratorDummyImpl](#) &orig)
- [~IntegratorDummyImpl](#) ()
- void [setPrecision](#) (double e)
- double [getPrecision](#) ()
- double [integrate](#) (double min, double max, double(\*f)(double, double), double p2)
- double [integrate](#) (double min, double max, double(\*f)(double, double, double), double p2, double p3)
- double [integrate](#) (double min, double max, double(\*f)(double, double, double, double), double p2, double p3, double p4)
- double [integrate](#) (double min, double max, double(\*f)(double, double, double, double, double), double p2, double p3, double p4, double p5)

### 5.30.1 Constructor & Destructor Documentation

5.30.1.1 [IntegratorDummyImpl::IntegratorDummyImpl](#) ( )

5.30.1.2 [IntegratorDummyImpl::IntegratorDummyImpl](#) ( const [IntegratorDummyImpl](#) & orig )

5.30.1.3 [IntegratorDummyImpl::~~IntegratorDummyImpl](#) ( )

### 5.30.2 Member Function Documentation

5.30.2.1 double [IntegratorDummyImpl::getPrecision](#) ( ) `[virtual]`

Implements [Integrator\\_if](#).

5.30.2.2 double [IntegratorDummyImpl::integrate](#) ( double *min*, double *max*, double(\*)*f*(double, double) *f*, double *p2* ) `[virtual]`

Implements [Integrator\\_if](#).

5.30.2.3 `double IntegratorDummyImpl::integrate ( double min, double max, double(*) (double, double, double) f, double p2, double p3 ) [virtual]`

Implements [Integrator\\_if](#).

5.30.2.4 `double IntegratorDummyImpl::integrate ( double min, double max, double(*) (double, double, double, double) f, double p2, double p3, double p4 ) [virtual]`

Implements [Integrator\\_if](#).

5.30.2.5 `double IntegratorDummyImpl::integrate ( double min, double max, double(*) (double, double, double, double, double) f, double p2, double p3, double p4, double p5 ) [virtual]`

Implements [Integrator\\_if](#).

5.30.2.6 `void IntegratorDummyImpl::setPrecision ( double e ) [virtual]`

Implements [Integrator\\_if](#).

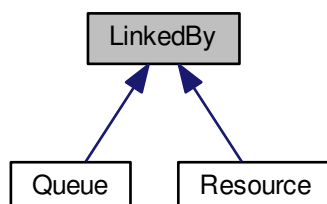
The documentation for this class was generated from the following files:

- [IntegratorDummyImpl.h](#)
- [IntegratorDummyImpl.cpp](#)

## 5.31 LinkBy Class Reference

```
#include <LinkBy.h>
```

Inheritance diagram for LinkBy:





## Public Member Functions

- [LinkedBy](#) ()
- [LinkedBy](#) (const [LinkedBy](#) &orig)
- virtual [~LinkedBy](#) ()
- void [addLink](#) ()
- void [removeLink](#) ()
- bool [isLinked](#) ()

### 5.31.1 Constructor & Destructor Documentation

5.31.1.1 [LinkedBy::LinkedBy](#) ( )

5.31.1.2 [LinkedBy::LinkedBy](#) ( const [LinkedBy](#) & *orig* )

5.31.1.3 [LinkedBy::~~LinkedBy](#) ( ) [[virtual](#)]

### 5.31.2 Member Function Documentation

5.31.2.1 void [LinkedBy::addLink](#) ( )

5.31.2.2 bool [LinkedBy::isLinked](#) ( )

5.31.2.3 void [LinkedBy::removeLink](#) ( )

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

- [LinkedBy.h](#)
- [LinkedBy.cpp](#)

## 5.32 List< T > Class Template Reference

```
#include <List.h>
```

### Public Types

- using [CompFuncT](#) = std::function< bool(const T, const T) >

## Public Member Functions

- [List](#) ()
- [List](#) (const [List](#) &orig)
- virtual [~List](#) ()
- unsigned int [size](#) ()
- bool [empty](#) ()
- void [clear](#) ()
- void [pop\\_front](#) ()
- template<class Compare >  
void [sort](#) (Compare comp)
- std::list< T > \* [getList](#) () const
- T [create](#) ()
- template<typename U >  
T [create](#) (U arg)
- std::string [show](#) ()
- std::list< T >::iterator [find](#) (T element)
- void [insert](#) (T element)
- void [remove](#) (T element)
- T [next](#) ()
- T [first](#) ()
- T [last](#) ()
- T [previous](#) ()
- T [actual](#) ()
- void [setSortFunc](#) (CompFunc\_t \_sortFunc)

### 5.32.1 Member Typedef Documentation

5.32.1.1 `template<typename T> using List< T >::CompFunc_t = std::function<bool(const T, const T) >`

### 5.32.2 Constructor & Destructor Documentation

5.32.2.1 `template<typename T> List< T >::List ( )`

5.32.2.2 `template<typename T> List< T >::List ( const List< T > & orig )`

5.32.2.3 `template<typename T> List< T >::~~List ( )` [virtual]

### 5.32.3 Member Function Documentation

5.32.3.1 `template<typename T> T List< T >::actual ( )`

5.32.3.2 `template<typename T> void List< T >::clear ( )`

Here is the caller graph for this function:

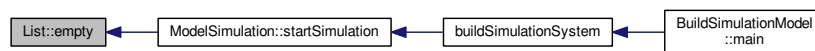


5.32.3.3 `template<typename T> T List< T >::create ( )`

5.32.3.4 `template<typename T> template<typename U> T List< T >::create ( U arg )`

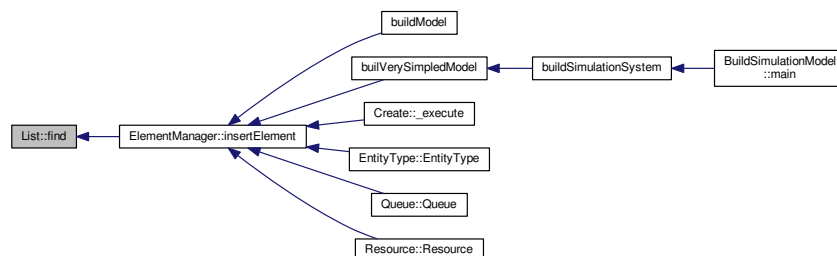
5.32.3.5 `template<typename T> bool List< T >::empty ( )`

Here is the caller graph for this function:



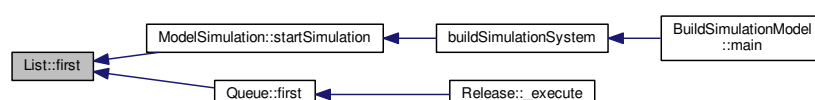
5.32.3.6 `template<typename T> std::list< T >::iterator List< T >::find ( T element )`

Here is the caller graph for this function:

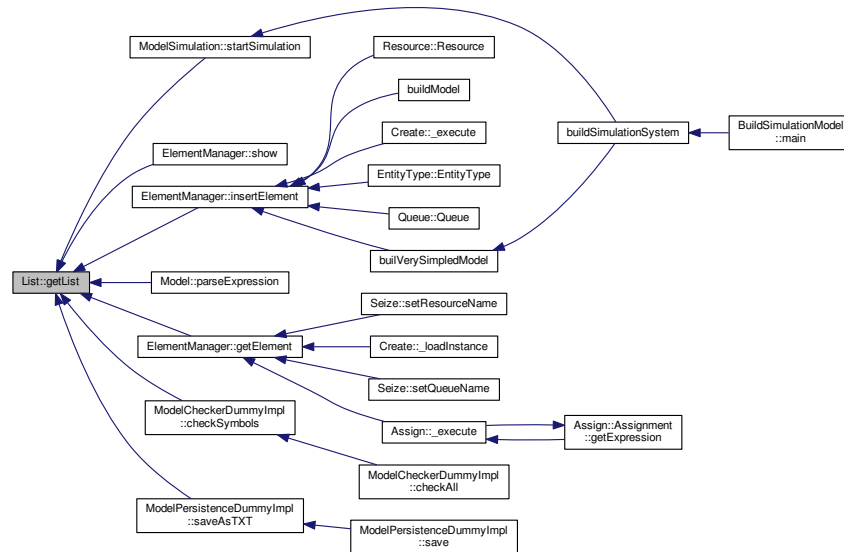


5.32.3.7 `template<typename T> T List< T >::first ( )`

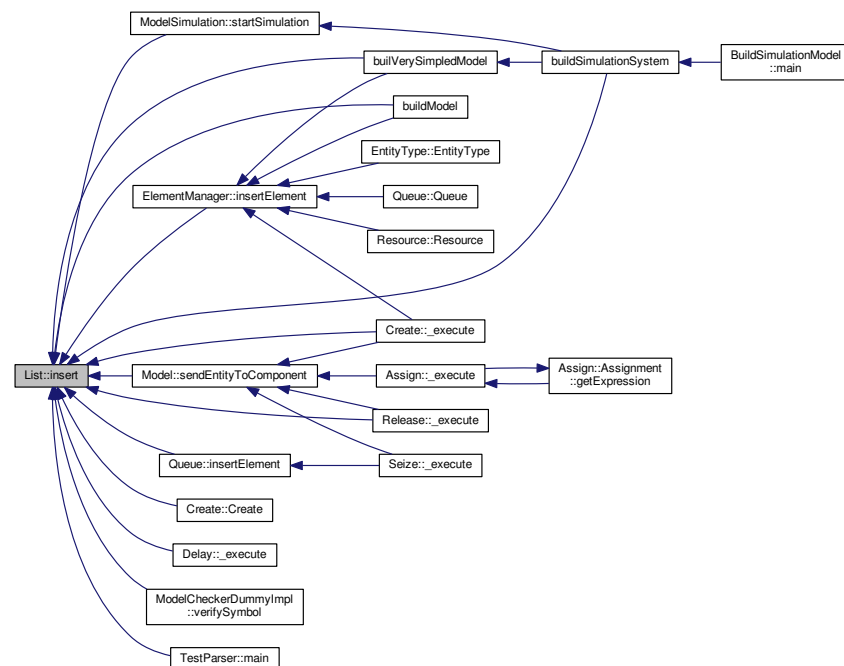
Here is the caller graph for this function:



Here is the caller graph for this function:



Here is the caller graph for this function:

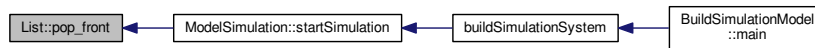


5.32.3.10 `template<typename T> T List< T >::last ( )`

5.32.3.11 `template<typename T> T List< T >::next ( )`

5.32.3.12 `template<typename T> void List< T >::pop_front ( )`

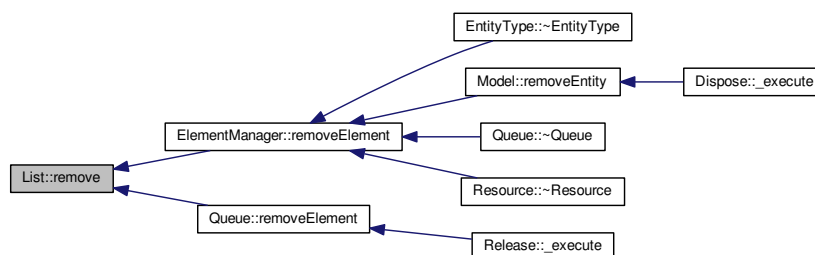
Here is the caller graph for this function:



5.32.3.13 `template<typename T> T List< T >::previous ( )`

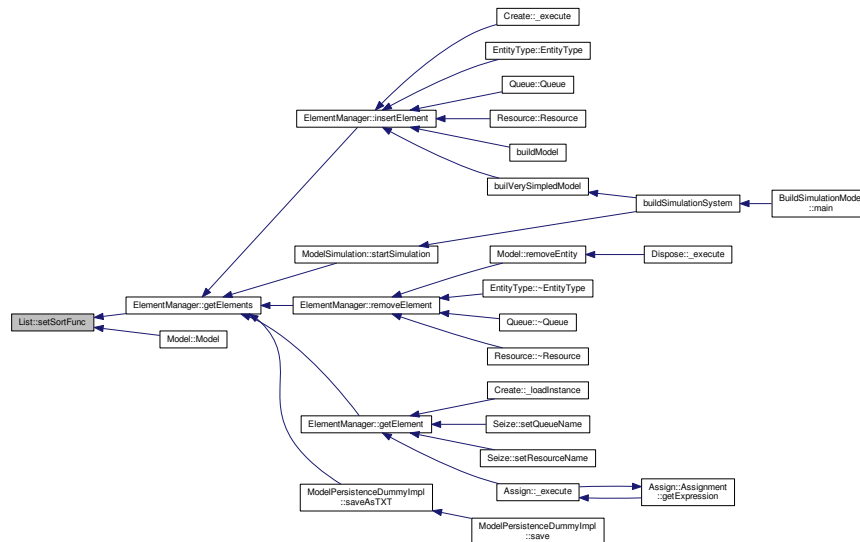
5.32.3.14 `template<typename T> void List< T >::remove ( T element )`

Here is the caller graph for this function:



### 5.32.3.15 `template<typename T> void List< T >::setSortFunc ( CompFuncT _sortFunc )`

Here is the caller graph for this function:



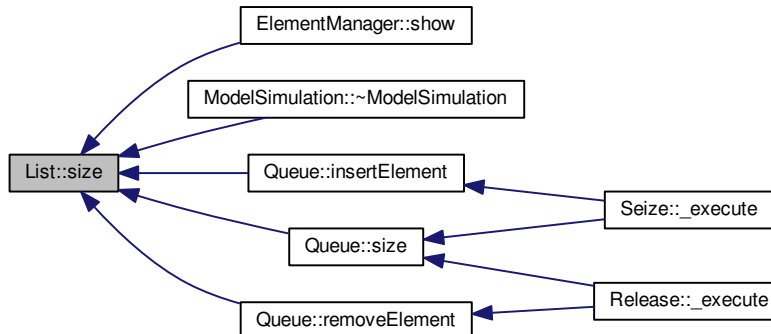
### 5.32.3.16 `template<typename T> std::string List< T >::show ( )`

Here is the caller graph for this function:



5.32.3.17 `template<typename T> unsigned int List< T >::size ( )`

Here is the caller graph for this function:

5.32.3.18 `template<typename T> template<class Compare> void List< T >::sort ( Compare comp )`

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

- [List.h](#)

## 5.33 Model Class Reference

```
#include <Model.h>
```

### Public Member Functions

- [Model](#) ([Simulator](#) \*simulator)
- [Model](#) (const [Model](#) &orig)
- virtual [~Model](#) ()
- void [showReports](#) ()
- bool [saveModel](#) (std::string filename)
- bool [loadModel](#) (std::string filename)
- bool [checkModel](#) ()

*Checks the integrity and consistency of the model, possibly corrects some inconsistencies, and returns if the model is in position to the simulated.*

- bool [verifySymbol](#) (std::string componentName, std::string expressionName, std::string expression, std::string expressionResult, bool mandatory)

*Verifies if a symbol defined in a component ([ModelComponent](#)) or element is syntactically valid and addresses existing components or elements. It's used only by and directed by the component that defines the symbol.*

- void [removeEntity](#) ([Entity](#) \*entity, bool collectStatistics)
- void [sendEntityToComponent](#) ([Entity](#) \*entity, [ModelComponent](#) \*component, double timeDelay)

Used by components ([ModelComponent](#)) to send entities to another specific component, usually the next one connected to it, or used by the model itself, when processing an event ([Event](#)).

- double [parseExpression](#) (const std::string expression)
- double [parseExpression](#) (const std::string expression, bool \*success, std::string \*errorMessage)
- [Util::identification](#) [getId](#) () const
- [List](#)< [SimulationControl](#) \* > \* [getControls](#) () const

Returns a list of values that can be externally controlled (changed). They usually correspond to input parameters in the simulation model that must be changed for an experimental design.

- [List](#)< [SimulationResponse](#) \* > \* [getResponses](#) () const

Returns a list of exits or simulation results that can be read externally. They usually correspond to statistics resulting from the simulation that must be read for an experiment design.

- [TraceManager](#) \* [getTracer](#) () const

Provides access to the class that performs the trace of simulation and replications.

- [OnEventManager](#) \* [getOnEventManager](#) () const
- [ElementManager](#) \* [getElementManager](#) () const

Provides access to the class that manages the most basic elements of the simulation model (such as queues, resources, variables, etc.).

- [ModelInfo](#) \* [getInfos](#) () const
- [Simulator](#) \* [getParent](#) () const
- [ModelSimulation](#) \* [getSimulation](#) () const

Provides access to the class that manages the model simulation.

- [List](#)< [ModelComponent](#) \* > \* [getComponents](#) () const

Returns the list of components (such as [Create](#), [Delay](#), [Dispose](#), etc.) that make up the simulation model.

- [List](#)< [Event](#) \* > \* [getEvents](#) () const

The future events list chronologically sorted; Events are scheduled by components when processing other events, and a replication evolves over time by sequentially processing the very first event in this list. It's initialized with events first described by source components ([SourceComponentModel](#)).

### 5.33.1 Detailed Description

[Model](#) is probably the most important class of Genesys kernel. It represents a discrete event-driven simulation model. Each model is responsible for controlling its own simulation, ie, for sequentially processing events and collecting statistical results. A model is mainly represented by a collection of components ([ModelComponent](#)), adequately configured and connected, and a collection of under layered element ([ModelElement](#)).

### 5.33.2 Constructor & Destructor Documentation

#### 5.33.2.1 [Model::Model](#) ( [Simulator](#) \* *simulator* )

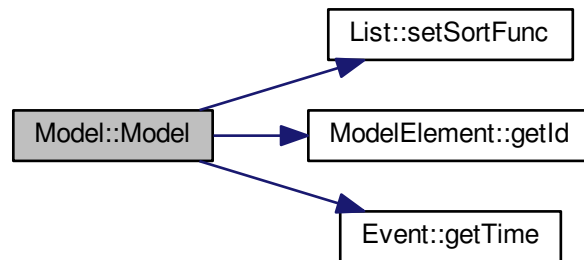
Components are sorted by ID

The future events list must be chronologically sorted

Events are sorted chronologically



Here is the call graph for this function:



5.33.2.2 `Model::Model ( const Model & orig )`

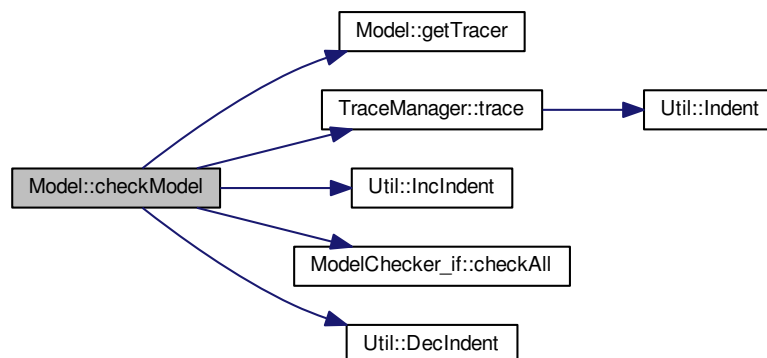
5.33.2.3 `Model::~~Model ( ) [virtual]`

### 5.33.3 Member Function Documentation

5.33.3.1 `bool Model::checkModel ( )`

Checks the integrity and consistency of the model, possibly corrects some inconsistencies, and returns if the model is in position to be simulated.

Here is the call graph for this function:



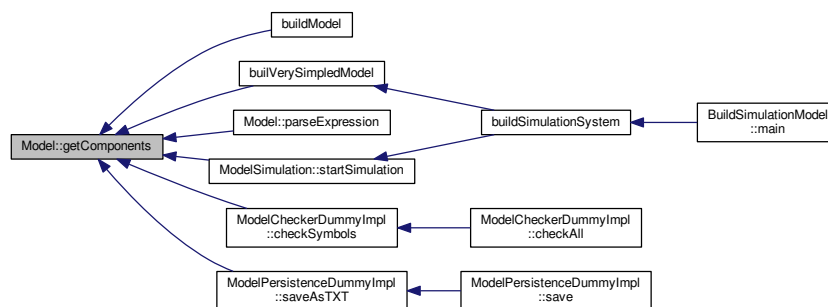
Here is the caller graph for this function:



### 5.33.3.2 `List< ModelComponent * > * Model::getComponents ( ) const`

Returns the list of components (such as [Create](#), [Delay](#), [Dispose](#), etc.) that make up the simulation model.

Here is the caller graph for this function:



### 5.33.3.3 `List< SimulationControl * > * Model::getControls ( ) const`

Returns a list of values that can be externally controlled (changed). They usually correspond to input parameters in the simulation model that must be changed for an experimental design.

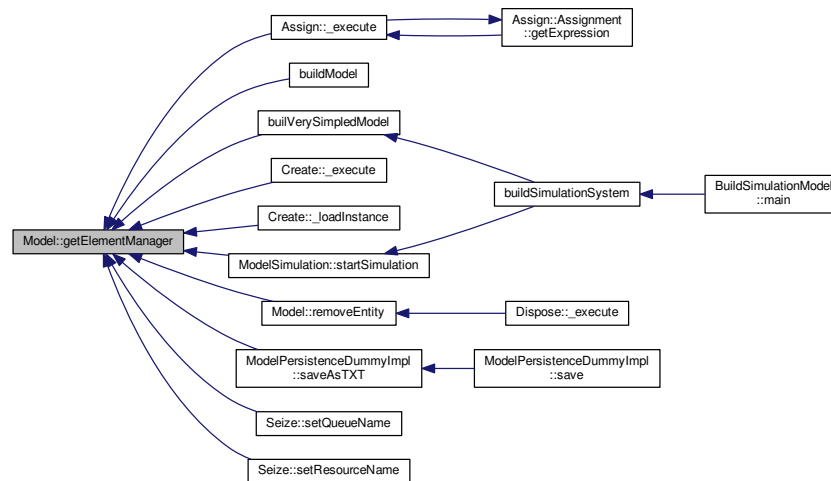
Here is the caller graph for this function:



#### 5.33.3.4 ElementManager \* Model::getElementManager ( ) const

Provides access to the class that manages the most basic elements of the simulation model (such as queues, resources, variables, etc.).

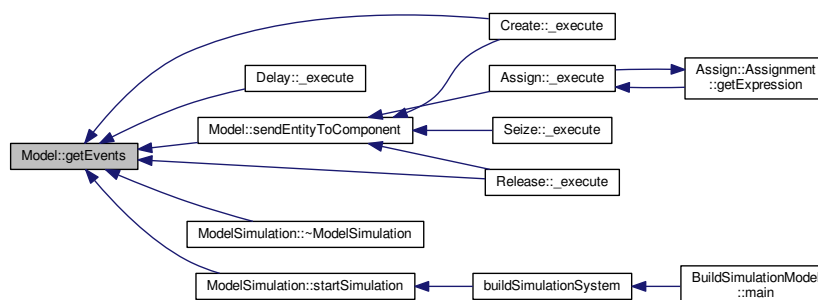
Here is the caller graph for this function:



#### 5.33.3.5 List< Event \* > \* Model::getEvents ( ) const

The future events list chronologically sorted; Events are scheduled by components when processing other events, and a replication evolves over time by sequentially processing the very first event in this list. It's initialized with events first described by source components (SourceComponentModel).

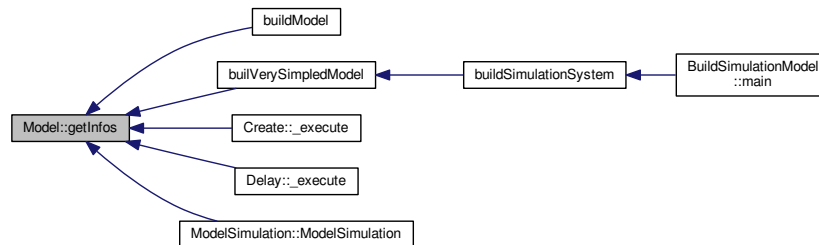
Here is the caller graph for this function:



#### 5.33.3.6 Util::identification Model::getId ( ) const

#### 5.33.3.7 ModelInfo \* Model::getInfos ( ) const

Here is the caller graph for this function:



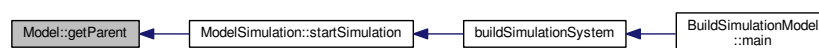
#### 5.33.3.8 OnEventManager \* Model::getOnEventManager ( ) const

Here is the caller graph for this function:



#### 5.33.3.9 Simulator \* Model::getParent ( ) const

Here is the caller graph for this function:



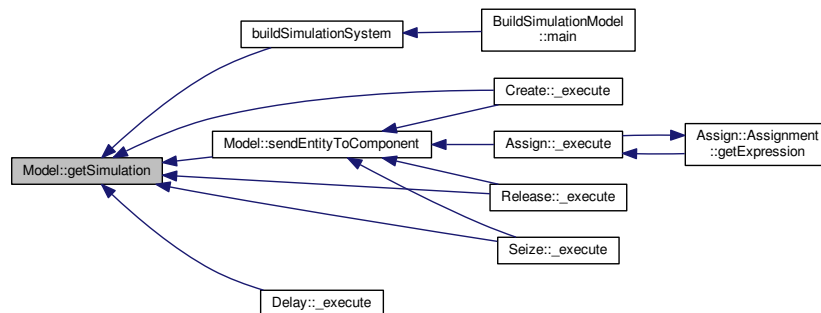
#### 5.33.3.10 List< SimulationResponse \* > \* Model::getResponses ( ) const

Returns a list of exits or simulation results that can be read externally. They usually correspond to statistics resulting from the simulation that must be read for an experiment design.

#### 5.33.3.11 ModelSimulation \* Model::getSimulation ( ) const

Provides access to the class that manages the model simulation.

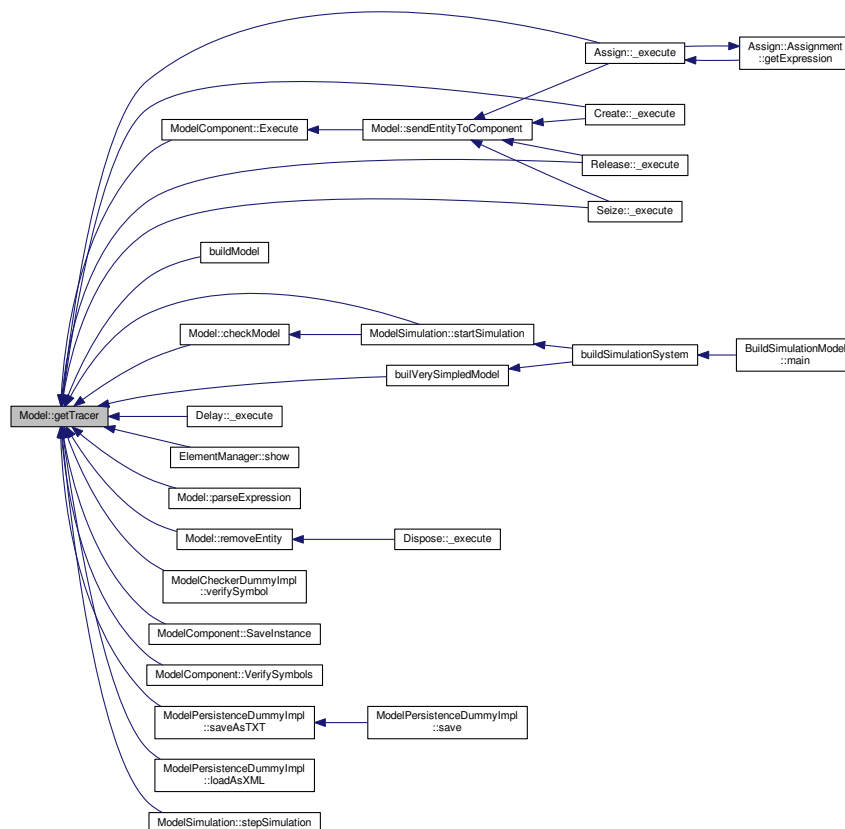
Here is the caller graph for this function:



#### 5.33.3.12 TraceManager \* Model::getTracer ( ) const

Provides access to the class that performs the trace of simulation and replications.

Here is the caller graph for this function:



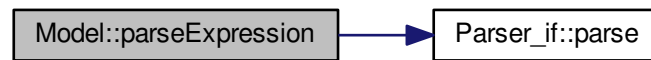
### 5.33.3.13 `bool Model::loadModel ( std::string filename )`

Here is the call graph for this function:

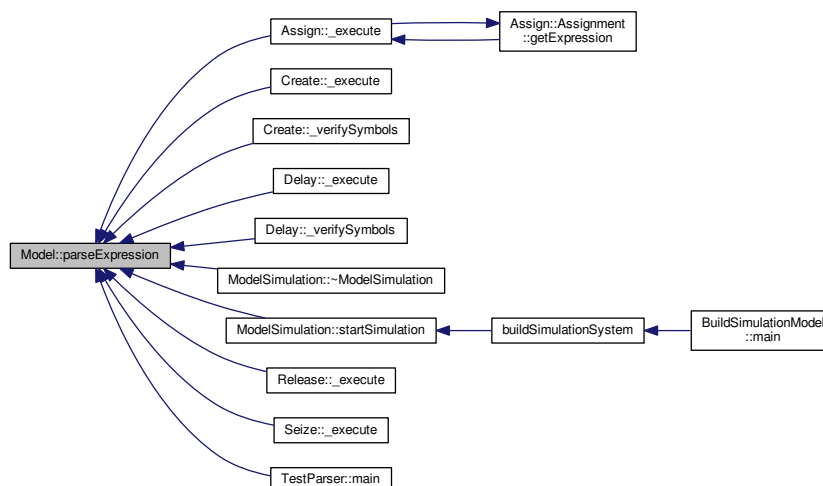


### 5.33.3.14 `double Model::parseExpression ( const std::string expression )`

Here is the call graph for this function:

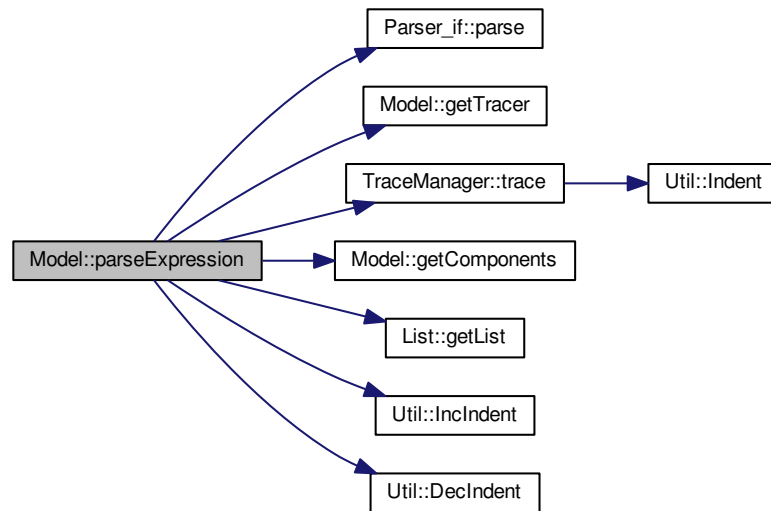


Here is the caller graph for this function:



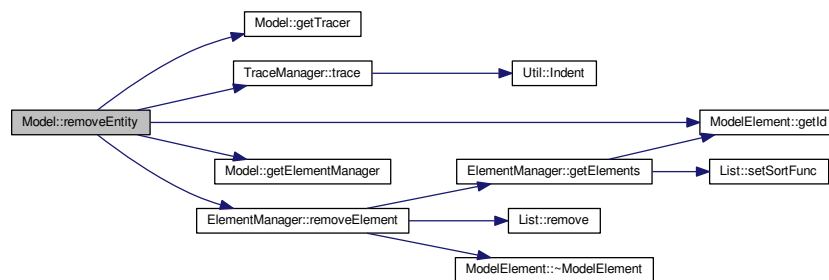
5.33.3.15 `double Model::parseExpression ( const std::string expression, bool * success, std::string * errorMessage )`

Here is the call graph for this function:



5.33.3.16 `void Model::removeEntity ( Entity * entity, bool collectStatistics )`

Here is the call graph for this function:



Here is the caller graph for this function:



### 5.33.3.17 `bool Model::saveModel ( std::string filename )`

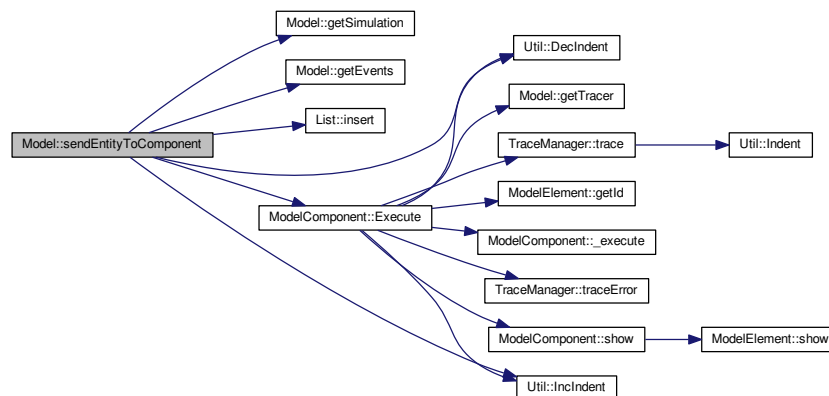
Here is the call graph for this function:



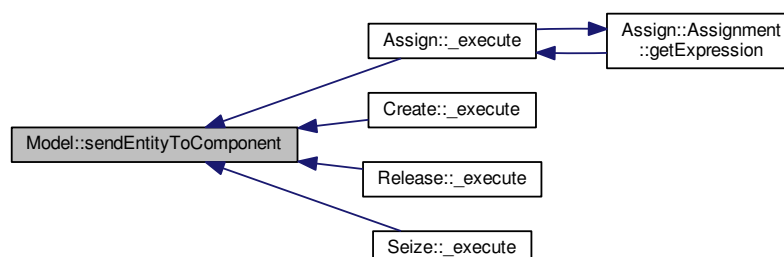
### 5.33.3.18 `void Model::sendEntityToComponent ( Entity * entity, ModelComponent * component, double timeDelay )`

Used by components ([ModelComponent](#)) to send entities to another specific component, usually the next one connected to it, or used by the model itself, when processing an event ([Event](#)).

Here is the call graph for this function:



Here is the caller graph for this function:





5.33.3.19 void Model::showReports ( )

5.33.3.20 bool Model::verifySymbol ( std::string *componentName*, std::string *expressionName*, std::string *expression*, std::string *expressionResult*, bool *mandatory* )

Verifies if a symbol defined in a component ([ModelComponent](#)) or element is syntactically valid and addresses existing components or elements. It's used only by and directed by the component that defines the symbol.

Here is the call graph for this function:



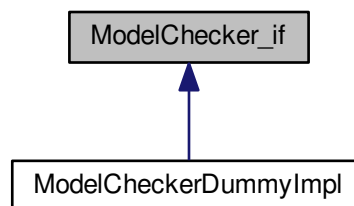
The documentation for this class was generated from the following files:

- [Model.h](#)
- [Model.cpp](#)

## 5.34 ModelChecker\_if Class Reference

```
#include <ModelChecker_if.h>
```

Inheritance diagram for ModelChecker\_if:



### Public Member Functions

- virtual bool [checkAll](#) ()=0
- virtual bool [checkAndAddInternalLiterals](#) ()=0
- virtual bool [checkConnected](#) ()=0
- virtual bool [checkSymbols](#) ()=0
- virtual bool [checkPathway](#) ()=0
- virtual bool [checkActivationCode](#) ()=0
- virtual bool [verifySymbol](#) (std::string *componentName*, std::string *expressionName*, std::string *expression*, std::string *expressionResult*, bool *mandatory*)=0

### 5.34.1 Detailed Description

The ModelChecker is responsible for verifying the model consistency, fixing inconsistencies whenever possible

### 5.34.2 Member Function Documentation

#### 5.34.2.1 `virtual bool ModelChecker_if::checkActivationCode ( ) [pure virtual]`

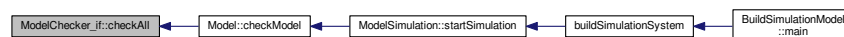
Check if components forms a valid pathway, including logical connections, such as routes, statios and pickups, for example

Implemented in [ModelCheckerDummyImpl](#).

#### 5.34.2.2 `virtual bool ModelChecker_if::checkAll ( ) [pure virtual]`

Implemented in [ModelCheckerDummyImpl](#).

Here is the caller graph for this function:



#### 5.34.2.3 `virtual bool ModelChecker_if::checkAndAddInternalLiterals ( ) [pure virtual]`

Invokes all other checks and returns true only if all of them returned true

Implemented in [ModelCheckerDummyImpl](#).

#### 5.34.2.4 `virtual bool ModelChecker_if::checkConnected ( ) [pure virtual]`

Implemented in [ModelCheckerDummyImpl](#).

#### 5.34.2.5 `virtual bool ModelChecker_if::checkPathway ( ) [pure virtual]`

Checks if user-defined strings for symbols required by components, usually expressions or functions, are valid or references existing and valid elements.

Implemented in [ModelCheckerDummyImpl](#).

#### 5.34.2.6 `virtual bool ModelChecker_if::checkSymbols ( ) [pure virtual]`

Checks if components are consistently connected to other to form a valid process-oriented model, describing how entities proceed to the flow

Implemented in [ModelCheckerDummyImpl](#).

5.34.2.7 virtual bool ModelChecker\_if::verifySymbol ( std::string *componentName*, std::string *expressionName*, std::string *expression*, std::string *expressionResult*, bool *mandatory* ) [pure virtual]

unnecessary

Implemented in [ModelCheckerDummyImpl](#).

Here is the caller graph for this function:



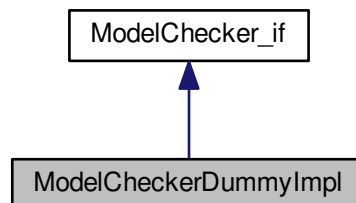
The documentation for this class was generated from the following file:

- [ModelChecker\\_if.h](#)

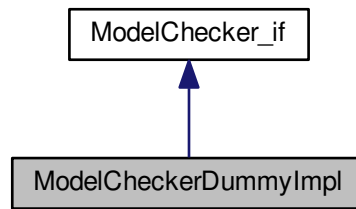
## 5.35 ModelCheckerDummyImpl Class Reference

```
#include <ModelCheckerDummyImpl.h>
```

Inheritance diagram for ModelCheckerDummyImpl:



Collaboration diagram for ModelCheckerDummyImpl:



## Public Member Functions

- [ModelCheckerDummyImpl \(Model \\*model\)](#)
- [ModelCheckerDummyImpl \(const ModelCheckerDummyImpl &orig\)](#)
- [~ModelCheckerDummyImpl \(\)](#)
- [bool checkAll \(\)](#)
- [bool checkAndAddInternalLiterals \(\)](#)
- [bool checkConnected \(\)](#)
- [bool checkSymbols \(\)](#)
- [bool checkPathway \(\)](#)
- [bool checkActivationCode \(\)](#)
- [bool verifySymbol \(std::string componentName, std::string expressionName, std::string expression, std::string expressionResult, bool mandatory\)](#)

### 5.35.1 Detailed Description

Just an example of possible implementation of the ModelChecker interface. Developers can implement their own class

### 5.35.2 Constructor & Destructor Documentation

5.35.2.1 `ModelCheckerDummyImpl::ModelCheckerDummyImpl ( Model * model )`

5.35.2.2 `ModelCheckerDummyImpl::ModelCheckerDummyImpl ( const ModelCheckerDummyImpl & orig )`

5.35.2.3 `ModelCheckerDummyImpl::~~ModelCheckerDummyImpl ( )`

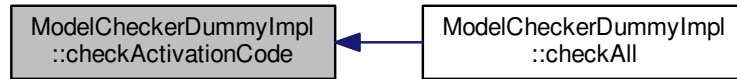
### 5.35.3 Member Function Documentation

5.35.3.1 `bool ModelCheckerDummyImpl::checkActivationCode ( ) [virtual]`

Check if components forms a valid pathway, including logical connections, such as routes, statios and pickups, for example

Implements [ModelChecker\\_if](#).

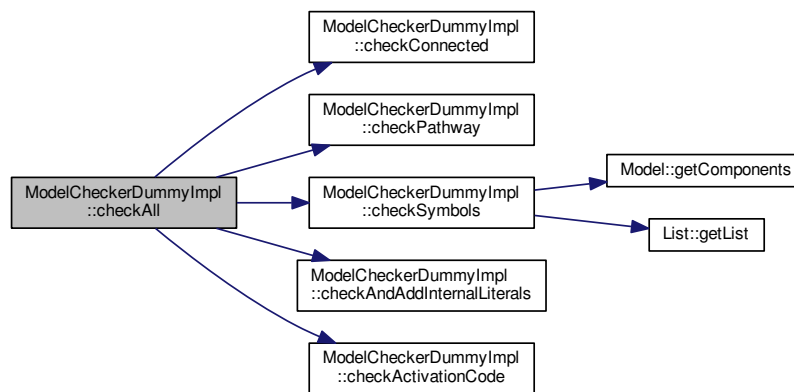
Here is the caller graph for this function:



#### 5.35.3.2 `bool ModelCheckerDummyImpl::checkAll ( ) [virtual]`

Implements [ModelChecker\\_if](#).

Here is the call graph for this function:

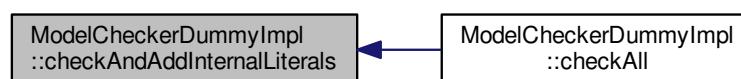


#### 5.35.3.3 `bool ModelCheckerDummyImpl::checkAndAddInternalLiterals ( ) [virtual]`

Invokes all other checks and returns true only if all of them returned true

Implements [ModelChecker\\_if](#).

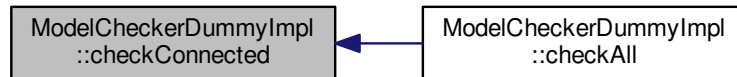
Here is the caller graph for this function:



#### 5.35.3.4 `bool ModelCheckerDummyImpl::checkConnected ( ) [virtual]`

Implements [ModelChecker\\_if](#).

Here is the caller graph for this function:

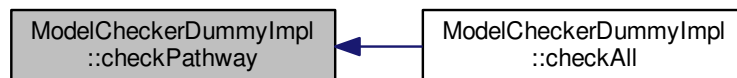


#### 5.35.3.5 `bool ModelCheckerDummyImpl::checkPathway ( ) [virtual]`

Checks if user-defined strings for symbols required by components, usually expressions or functions, are valid or references existing and valid elements.

Implements [ModelChecker\\_if](#).

Here is the caller graph for this function:

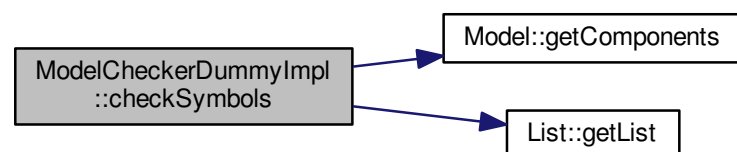


#### 5.35.3.6 `bool ModelCheckerDummyImpl::checkSymbols ( ) [virtual]`

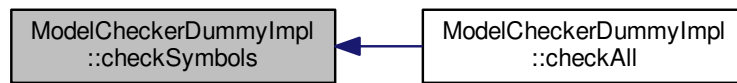
Checks if components are consistently connected to other to form a valid process-oriented model, describing how entities proceed to the flow

Implements [ModelChecker\\_if](#).

Here is the call graph for this function:



Here is the caller graph for this function:

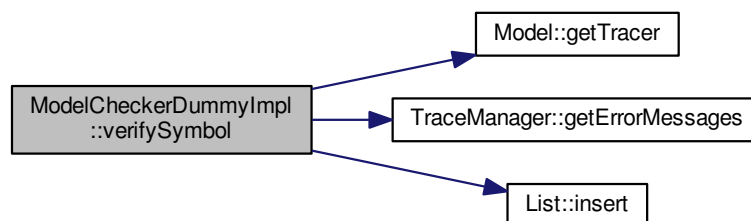


5.35.3.7 `bool ModelCheckerDummyImpl::verifySymbol ( std::string componentName, std::string expressionName, std::string expression, std::string expressionResult, bool mandatory ) [virtual]`

unnecessary

Implements [ModelChecker\\_if](#).

Here is the call graph for this function:



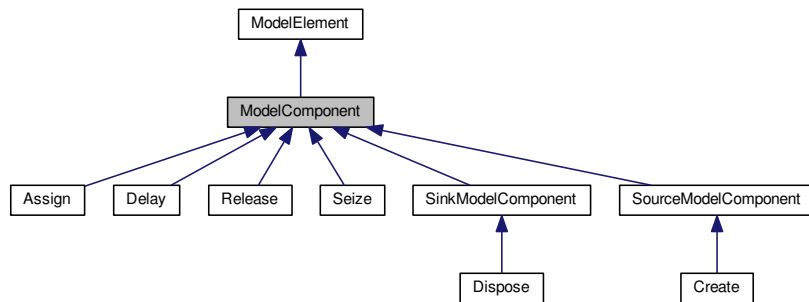
The documentation for this class was generated from the following files:

- [ModelCheckerDummyImpl.h](#)
- [ModelCheckerDummyImpl.cpp](#)

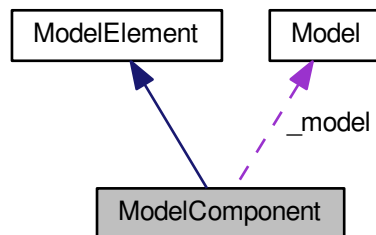
## 5.36 ModelComponent Class Reference

```
#include <ModelComponent.h>
```

Inheritance diagram for ModelComponent:



Collaboration diagram for ModelComponent:



## Public Member Functions

- [ModelComponent](#) ([Model](#) \*model)
- [ModelComponent](#) (const [ModelComponent](#) &orig)
- virtual [~ModelComponent](#) ()
- virtual std::string [show](#) ()
- List< [ModelComponent](#) \* > \* [getNextComponents](#) () const

Returns a list of components directly connected to the output. Usually the components have a single output, but they may have none (such as [Dispose](#)) or more than one (as [Decide](#)).

## Static Public Member Functions

- static void [Execute](#) ([Entity](#) \*entity, [ModelComponent](#) \*component)
- This method triggers the simulation of the behavior of the component. It is invoked when an event (corresponding to this component) is taken from the list of future events or when an entity arrives at this component by connection.
- static bool [VerifySymbols](#) ([ModelComponent](#) \*component, std::string \*errorMessage)
  - static std::list< std::string > \* [SaveInstance](#) ([ModelComponent](#) \*component)



## Protected Member Functions

- virtual void `_execute` (`Entity *entity`)=0
- virtual `std::list< std::string > * _saveInstance` ()
- virtual `std::list< std::string > * _saveInstance` (`std::string type`)

## Protected Attributes

- `Model * _model`

### 5.36.1 Detailed Description

Um componente do modelo é um bloco que representa um comportamento específico a ser simulado. O comportamento é disparado quando uma entidade chega ao componente, o que corresponde à ocorrência de um evento. Um modelo de simulação corresponde a um conjunto de componentes interconectados para formar o processo pelo qual a entidade é submetida.

#### Parameters

<i>model</i>	The model this component belongs to
--------------	-------------------------------------

### 5.36.2 Constructor & Destructor Documentation

5.36.2.1 `ModelComponent::ModelComponent ( Model * model )`

5.36.2.2 `ModelComponent::ModelComponent ( const ModelComponent & orig )`

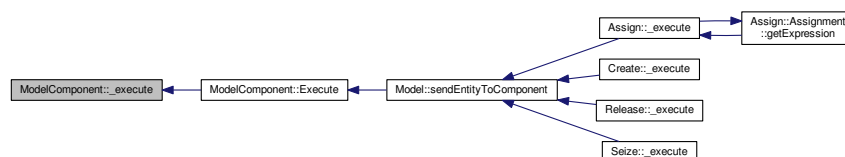
5.36.2.3 `ModelComponent::~ModelComponent ( )` [virtual]

### 5.36.3 Member Function Documentation

5.36.3.1 `virtual void ModelComponent::_execute ( Entity * entity )` [protected],[pure virtual]

Implemented in [Assign](#), [Seize](#), [Release](#), [Create](#), [Delay](#), and [Dispose](#).

Here is the caller graph for this function:



5.36.3.2 `std::list< std::string > * ModelComponent::_saveInstance ( )` [protected],[virtual]

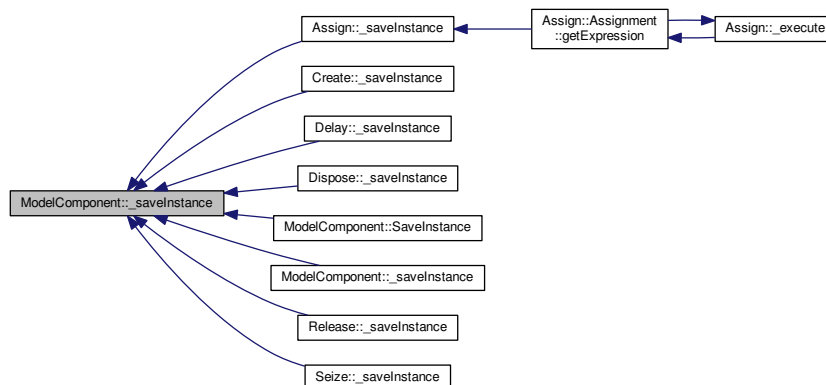
Reimplemented from [ModelElement](#).

Reimplemented in [Assign](#), [Seize](#), [Release](#), [Create](#), [Delay](#), and [Dispose](#).

Here is the call graph for this function:



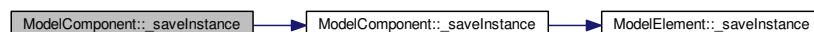
Here is the caller graph for this function:



5.36.3.3 `std::list< std::string > * ModelComponent::_saveInstance ( std::string type )` [protected],[virtual]

Reimplemented from [ModelElement](#).

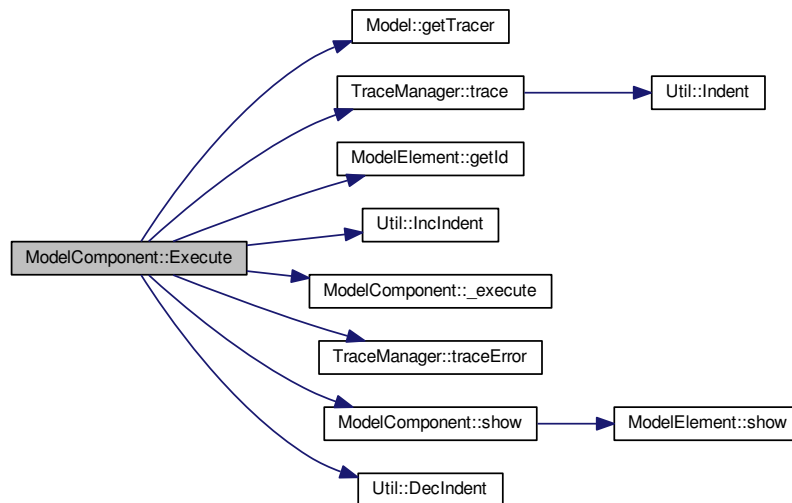
Here is the call graph for this function:



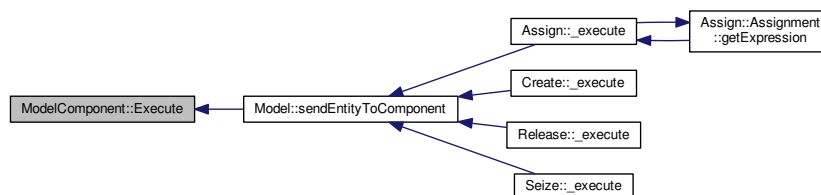
#### 5.36.3.4 void ModelComponent::Execute ( Entity \* entity, ModelComponent \* component ) [static]

This method triggers the simulation of the behavior of the component. It is invoked when an event (corresponding to this component) is taken from the list of future events or when an entity arrives at this component by connection.

Here is the call graph for this function:



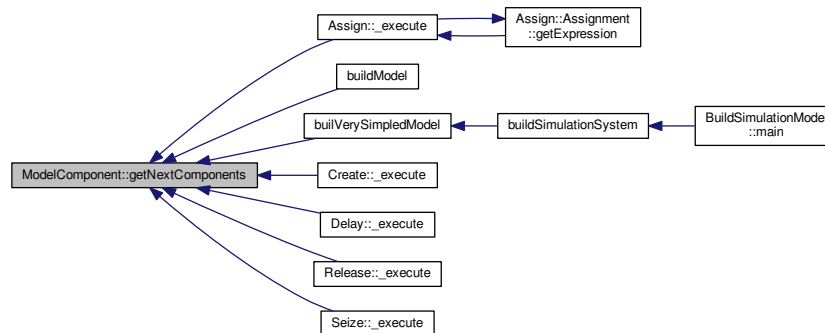
Here is the caller graph for this function:



#### 5.36.3.5 List< ModelComponent \* > \* ModelComponent::getNextComponents ( ) const

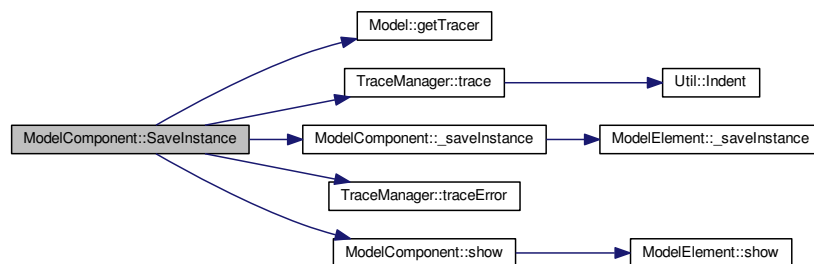
Returns a list of components directly connected to the output. Usually the components have a single output, but they may have none (such as [Dispose](#)) or more than one (as [Decide](#)).

Here is the caller graph for this function:



#### 5.36.3.6 `std::list< std::string > * ModelComponent::SaveInstance ( ModelComponent * component ) [static]`

Here is the call graph for this function:



#### 5.36.3.7 `std::string ModelComponent::show ( ) [virtual]`

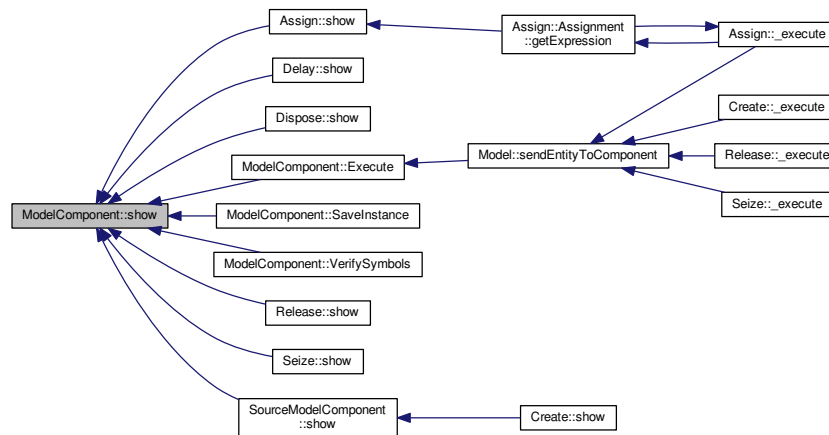
Reimplemented from [ModelElement](#).

Reimplemented in [Assign](#), [SourceModelComponent](#), [Seize](#), [Create](#), [Delay](#), [Release](#), and [Dispose](#).

Here is the call graph for this function:

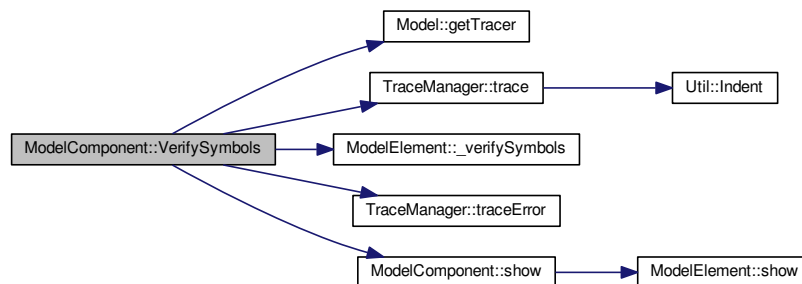


Here is the caller graph for this function:



**5.36.3.8** `bool ModelComponent::VerifySymbols ( ModelComponent * component, std::string * errorMessage )`  
`[static]`

Here is the call graph for this function:



## 5.36.4 Member Data Documentation

**5.36.4.1** `Model* ModelComponent::_model` `[protected]`

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

- [ModelComponent.h](#)
- [ModelComponent.cpp](#)

## 5.37 ModelComponentManager\_if Class Reference

```
#include <ModelComponentManager_if.h>
```

## Public Member Functions

- [ModelComponentManager\\_if](#) ()
- [ModelComponentManager\\_if](#) (const [ModelComponentManager\\_if](#) &orig)
- virtual [~ModelComponentManager\\_if](#) ()

### 5.37.1 Constructor & Destructor Documentation

5.37.1.1 [ModelComponentManager\\_if::ModelComponentManager\\_if](#) ( )

5.37.1.2 [ModelComponentManager\\_if::ModelComponentManager\\_if](#) ( const [ModelComponentManager\\_if](#) & orig )

5.37.1.3 virtual [ModelComponentManager\\_if::~ModelComponentManager\\_if](#) ( ) [virtual]

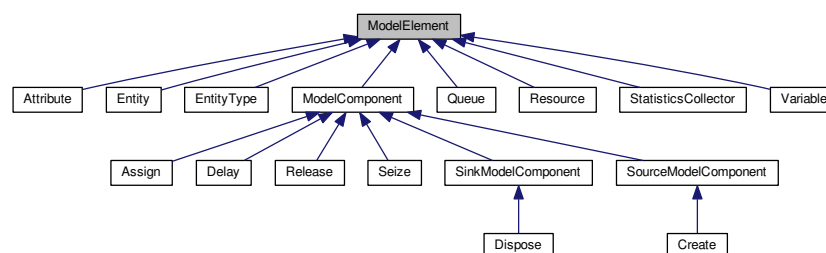
The documentation for this class was generated from the following file:

- [ModelComponentManager\\_if.h](#)

## 5.38 ModelElement Class Reference

```
#include <ModelElement.h>
```

Inheritance diagram for ModelElement:



## Public Member Functions

- [ModelElement](#) (std::string elementTypename)
- [ModelElement](#) (const [ModelElement](#) &orig)
- virtual [~ModelElement](#) ()
- virtual std::string [show](#) ()
- [Util::identification getId](#) () const
- void [setName](#) (std::string \_name)
- std::string [getName](#) () const

### Static Public Member Functions

- static void [LoadInstance](#) (std::list< std::string > words)
- static std::list< std::string > \* [SaveInstance](#) ([ModelElement](#) \*element)
- static bool [VerifySymbols](#) ([ModelElement](#) \*element, std::string \*errorMessage)

### Protected Member Functions

- virtual void [\\_loadInstance](#) (std::list< std::string > words)=0
- virtual std::list< std::string > \* [\\_saveInstance](#) ()
- virtual std::list< std::string > \* [\\_saveInstance](#) (std::string type)
- virtual bool [\\_verifySymbols](#) (std::string \*errorMessage)=0

### Protected Attributes

- [Util::identification\\_id](#)
- std::string [\\_name](#)

#### 5.38.1 Detailed Description

This class is the basis for any element of the model (such as [Queue](#), [Resource](#), [Variable](#), etc.) and also for any component of the model. It has the infrastructure to read and write on file and to verify symbols.

#### 5.38.2 Constructor & Destructor Documentation

##### 5.38.2.1 [ModelElement::ModelElement](#) ( std::string *elementTypename* )

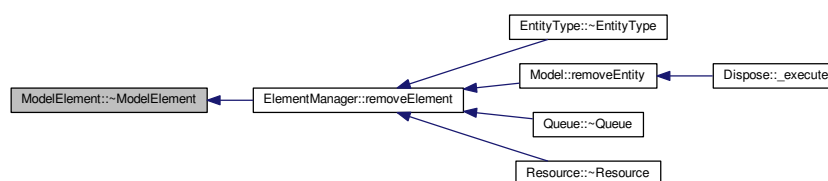
Here is the call graph for this function:



##### 5.38.2.2 [ModelElement::ModelElement](#) ( const [ModelElement](#) & *orig* )

##### 5.38.2.3 [ModelElement::~~ModelElement](#) ( ) [virtual]

Here is the caller graph for this function:



### 5.38.3 Member Function Documentation

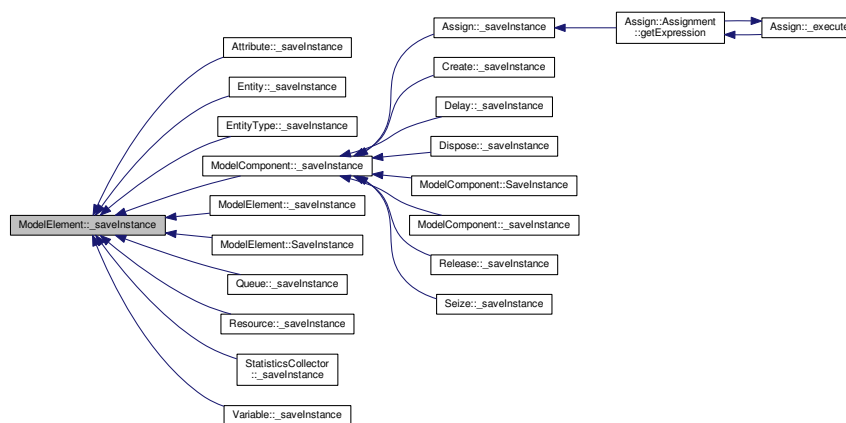
5.38.3.1 `virtual void ModelElement::_loadInstance ( std::list< std::string > words )` [protected], [pure virtual]

Implemented in [Assign](#), [Resource](#), [Seize](#), [Queue](#), [EntityType](#), [Release](#), [Entity](#), [Variable](#), [Create](#), [Delay](#), [StatisticsCollector](#), [Attribute](#), and [Dispose](#).

5.38.3.2 `std::list< std::string > * ModelElement::_saveInstance ( )` [protected], [virtual]

Reimplemented in [Assign](#), [Resource](#), [ModelComponent](#), [Seize](#), [Queue](#), [EntityType](#), [Release](#), [Entity](#), [Variable](#), [Create](#), [Delay](#), [StatisticsCollector](#), [Attribute](#), and [Dispose](#).

Here is the caller graph for this function:



5.38.3.3 `std::list< std::string > * ModelElement::_saveInstance ( std::string type )` [protected], [virtual]

Reimplemented in [ModelComponent](#).

Here is the call graph for this function:





```
5.38.3.4 virtual bool ModelElement::_verifySymbols ( std::string * errorMessage ) [protected], [pure virtual]
```

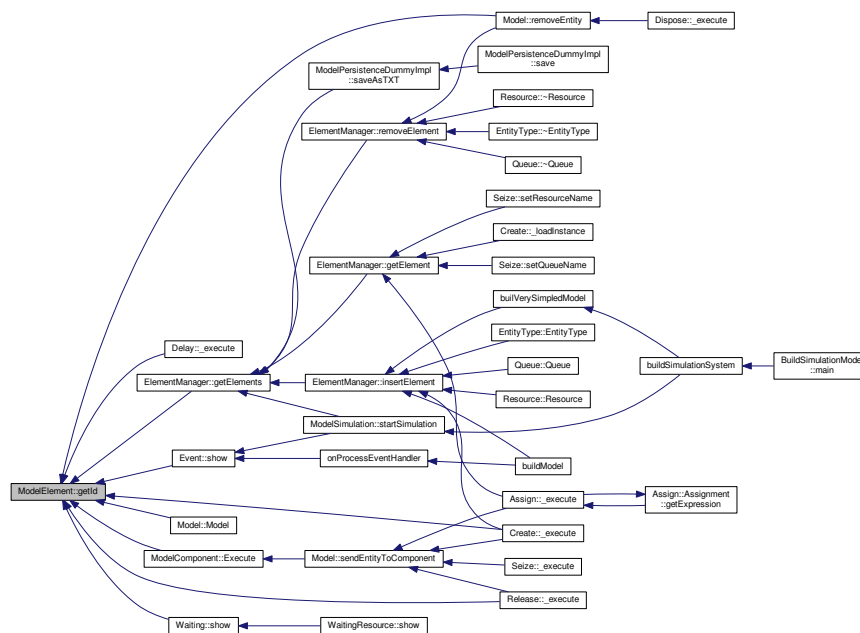
Implemented in [Assign](#), [Resource](#), [Seize](#), [Queue](#), [EntityType](#), [Release](#), [Entity](#), [Variable](#), [Create](#), [Delay](#), [Statistics](#)↵  
[Collector](#), [Attribute](#), and [Dispose](#).

Here is the caller graph for this function:



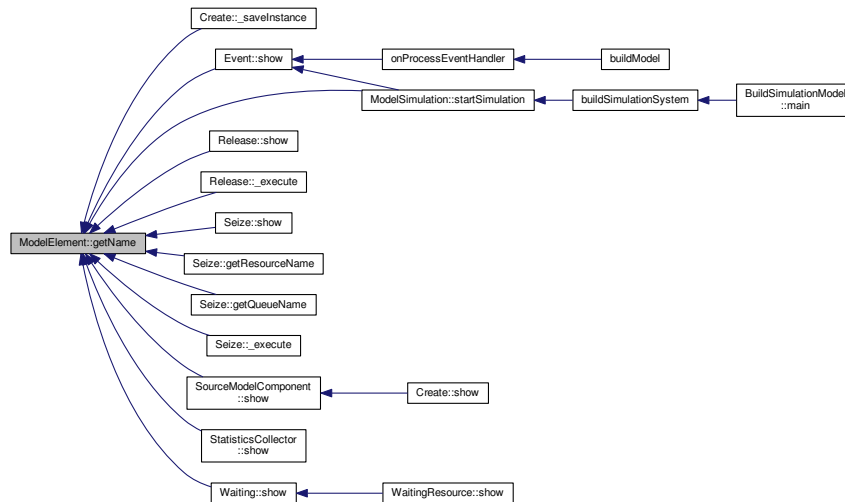
### 5.38.3.5 Util::identitification ModelElement::getId ( ) const

Here is the caller graph for this function:



### 5.38.3.6 `std::string ModelElement::getName ( ) const`

Here is the caller graph for this function:



### 5.38.3.7 `static void ModelElement::LoadInstance ( std::list< std::string > words ) [static]`

### 5.38.3.8 `std::list< std::string > * ModelElement::SaveInstance ( ModelElement * element ) [static]`

Here is the call graph for this function:

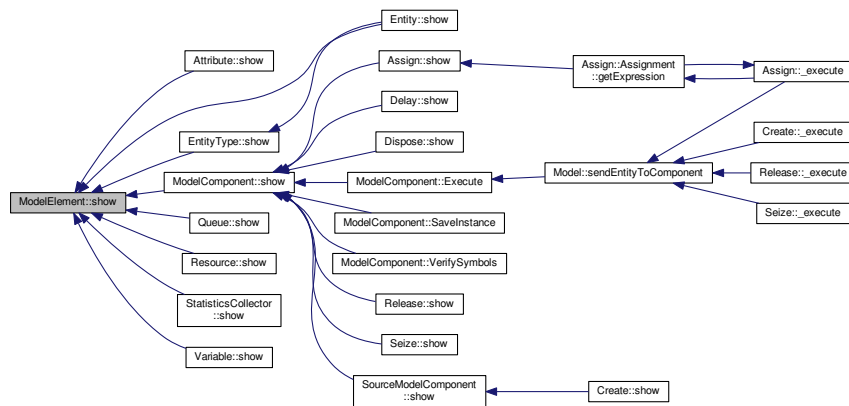


### 5.38.3.9 `void ModelElement::setName ( std::string _name )`

### 5.38.3.10 `std::string ModelElement::show ( ) [virtual]`

Reimplemented in [Assign](#), [SourceModelComponent](#), [ModelComponent](#), [Resource](#), [Queue](#), [EntityType](#), [Seize](#), [Entity](#), [Create](#), [Delay](#), [Attribute](#), [Release](#), [StatisticsCollector](#), [Variable](#), and [Dispose](#).

Here is the caller graph for this function:



5.38.3.11 static bool ModelElement::VerifySymbols ( ModelElement \* *element*, std::string \* *errorMessage* ) [static]

## 5.38.4 Member Data Documentation

5.38.4.1 Util::identification ModelElement::\_id [protected]

5.38.4.2 std::string ModelElement::\_name [protected]

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

- [ModelElement.h](#)
- [ModelElement.cpp](#)

## 5.39 ModelInfo Class Reference

```
#include <ModelInfo.h>
```

### Public Member Functions

- [ModelInfo](#) ()
- [ModelInfo](#) (const [ModelInfo](#) &orig)
- virtual [~ModelInfo](#) ()
- void [setName](#) (std::string \_name)
- std::string [getName](#) () const
- void [setAnalystName](#) (std::string \_analystName)
- std::string [getAnalystName](#) () const
- void [setDescription](#) (std::string \_description)
- std::string [getDescription](#) () const
- void [setProjectTitle](#) (std::string \_projectTitle)

- `std::string getTitle () const`
- `void setVersion (std::string _version)`
- `std::string getVersion () const`
- `void setNumberOfReplications (unsigned int _numberOfReplications)`
- `unsigned int getNumberOfReplications () const`
- `void setReplicationLength (double _replicationLength)`
- `double getReplicationLength () const`
- `void setReplicationLengthTimeUnit (Util::TimeUnit _replicationLengthTimeUnit)`
- `Util::TimeUnit getReplicationLengthTimeUnit () const`
- `void setWarmUpPeriod (double _warmUpPeriod)`
- `double getWarmUpPeriod () const`
- `void setWarmUpPeriodTimeUnit (Util::TimeUnit _warmUpPeriodTimeUnit)`
- `Util::TimeUnit getWarmUpPeriodTimeUnit () const`
- `void setTerminatingCondition (std::string _terminatingCondition)`
- `std::string getTerminatingCondition () const`

### 5.39.1 Detailed Description

`ModellInfo` stores basic model project information.

### 5.39.2 Constructor & Destructor Documentation

5.39.2.1 `ModellInfo::ModellInfo ( )`

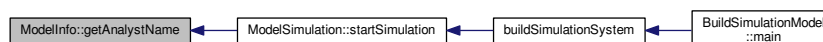
5.39.2.2 `ModellInfo::ModellInfo ( const ModellInfo & orig )`

5.39.2.3 `ModellInfo::~ModellInfo ( )` [virtual]

### 5.39.3 Member Function Documentation

5.39.3.1 `std::string ModellInfo::getAnalystName ( ) const`

Here is the caller graph for this function:



5.39.3.2 `std::string ModellInfo::getDescription ( ) const`

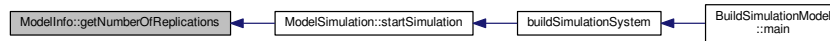
5.39.3.3 `std::string ModellInfo::getName ( ) const`

Here is the caller graph for this function:



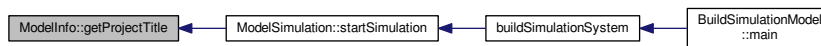
#### 5.39.3.4 unsigned int ModelInfo::getNumberOfReplications ( ) const

Here is the caller graph for this function:



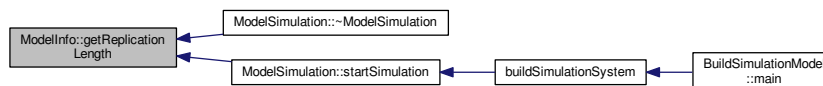
#### 5.39.3.5 std::string ModelInfo::getProjectTitle ( ) const

Here is the caller graph for this function:



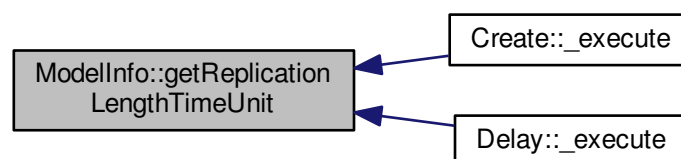
#### 5.39.3.6 double ModelInfo::getReplicationLength ( ) const

Here is the caller graph for this function:



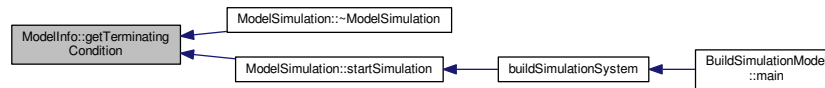
#### 5.39.3.7 Util::TimeUnit ModelInfo::getReplicationLengthTimeUnit ( ) const

Here is the caller graph for this function:



#### 5.39.3.8 `std::string ModelInfo::getTerminatingCondition ( ) const`

Here is the caller graph for this function:



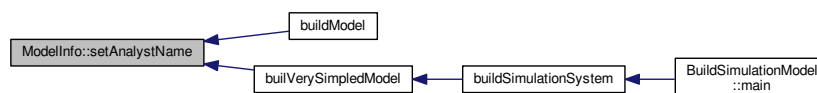
#### 5.39.3.9 `std::string ModelInfo::getVersion ( ) const`

#### 5.39.3.10 `double ModelInfo::getWarmUpPeriod ( ) const`

#### 5.39.3.11 `Util::TimeUnit ModelInfo::getWarmUpPeriodTimeUnit ( ) const`

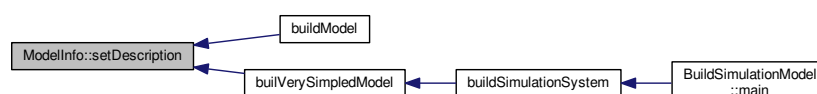
#### 5.39.3.12 `void ModelInfo::setAnalystName ( std::string _analystName )`

Here is the caller graph for this function:



#### 5.39.3.13 `void ModelInfo::setDescription ( std::string _description )`

Here is the caller graph for this function:



5.39.3.14 void ModelInfo::setName ( std::string *\_name* )

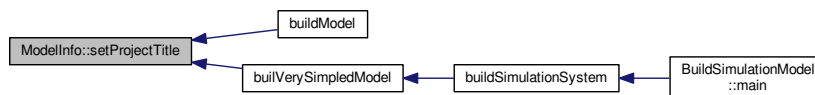
5.39.3.15 void ModelInfo::setNumberOfReplications ( unsigned int *\_numberOfReplications* )

Here is the caller graph for this function:



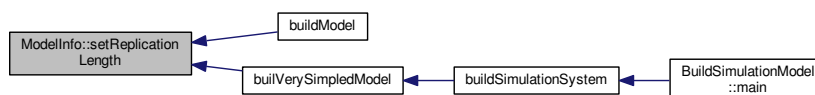
5.39.3.16 void ModelInfo::setProjectTitle ( std::string *\_projectTitle* )

Here is the caller graph for this function:



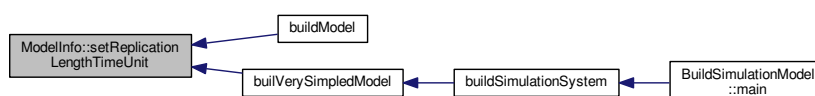
5.39.3.17 void ModelInfo::setReplicationLength ( double *\_replicationLength* )

Here is the caller graph for this function:



5.39.3.18 void ModelInfo::setReplicationLengthTimeUnit ( Util::TimeUnit *\_replicationLengthTimeUnit* )

Here is the caller graph for this function:



5.39.3.19 void ModelInfo::setTerminatingCondition ( std::string *\_terminatingCondition* )

5.39.3.20 void ModelInfo::setVersion ( std::string *\_version* )

5.39.3.21 void ModelInfo::setWarmUpPeriod ( double *\_warmUpPeriod* )

5.39.3.22 void ModelInfo::setWarmUpPeriodTimeUnit ( Util::TimeUnit *\_warmUpPeriodTimeUnit* )

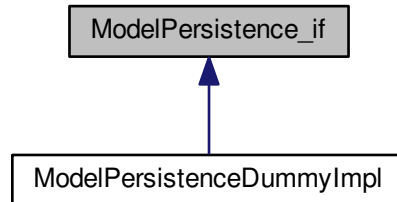
The documentation for this class was generated from the following files:

- [ModelInfo.h](#)
- [ModelInfo.cpp](#)

## 5.40 ModelPersistence\_if Class Reference

```
#include <ModelPersistence_if.h>
```

Inheritance diagram for ModelPersistence\_if:



### Public Member Functions

- virtual bool [saveAsTXT](#) (std::string filename)=0
- virtual bool [loadAsTXT](#) (std::string filename)=0
- virtual bool [saveAsXML](#) (std::string filename)=0
- virtual bool [loadAsXML](#) (std::string filename)=0
- virtual bool [save](#) (std::string filename)=0
- virtual bool [load](#) (std::string filename)=0
- virtual bool [isSaved](#) ()=0

#### 5.40.1 Detailed Description

First and inadequate interface for model persistence. It should use the best pattern for the DAO approach



## 5.40.2 Member Function Documentation

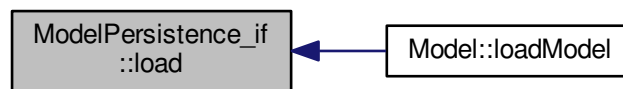
5.40.2.1 `virtual bool ModelPersistence_if::isSaved ( )` [pure virtual]

Implemented in [ModelPersistenceDummyImpl](#).

5.40.2.2 `virtual bool ModelPersistence_if::load ( std::string filename )` [pure virtual]

Implemented in [ModelPersistenceDummyImpl](#).

Here is the caller graph for this function:



5.40.2.3 `virtual bool ModelPersistence_if::loadAsTXT ( std::string filename )` [pure virtual]

Implemented in [ModelPersistenceDummyImpl](#).

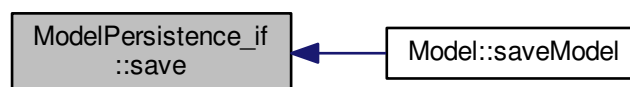
5.40.2.4 `virtual bool ModelPersistence_if::loadAsXML ( std::string filename )` [pure virtual]

Implemented in [ModelPersistenceDummyImpl](#).

5.40.2.5 `virtual bool ModelPersistence_if::save ( std::string filename )` [pure virtual]

Implemented in [ModelPersistenceDummyImpl](#).

Here is the caller graph for this function:



5.40.2.6 `virtual bool ModelPersistence_if::saveAsTXT ( std::string filename ) [pure virtual]`

Implemented in [ModelPersistenceDummyImpl](#).

5.40.2.7 `virtual bool ModelPersistence_if::saveAsXML ( std::string filename ) [pure virtual]`

Implemented in [ModelPersistenceDummyImpl](#).

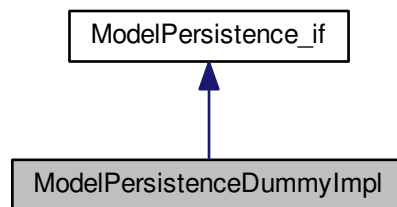
The documentation for this class was generated from the following file:

- [ModelPersistence\\_if.h](#)

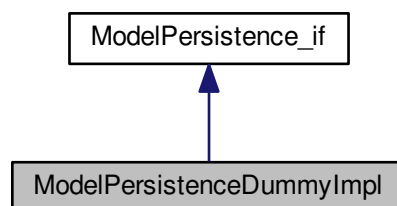
## 5.41 ModelPersistenceDummyImpl Class Reference

```
#include <ModelPersistenceDummyImpl.h>
```

Inheritance diagram for ModelPersistenceDummyImpl:



Collaboration diagram for ModelPersistenceDummyImpl:



## Public Member Functions

- [ModelPersistenceDummyImpl](#) ([Model](#) \*model)
- [ModelPersistenceDummyImpl](#) (const [ModelPersistenceDummyImpl](#) &orig)
- [~ModelPersistenceDummyImpl](#) ()
- virtual bool [saveAsTXT](#) (std::string filename)
- virtual bool [loadAsTXT](#) (std::string filename)
- virtual bool [saveAsXML](#) (std::string filename)
- virtual bool [loadAsXML](#) (std::string filename)
- virtual bool [save](#) (std::string filename)
- virtual bool [load](#) (std::string filename)
- virtual bool [isSaved](#) ()

## 5.41.1 Constructor &amp; Destructor Documentation

5.41.1.1 [ModelPersistenceDummyImpl::ModelPersistenceDummyImpl](#) ( [Model](#) \* *model* )

5.41.1.2 [ModelPersistenceDummyImpl::ModelPersistenceDummyImpl](#) ( const [ModelPersistenceDummyImpl](#) & *orig* )

5.41.1.3 [ModelPersistenceDummyImpl::~~ModelPersistenceDummyImpl](#) ( )

## 5.41.2 Member Function Documentation

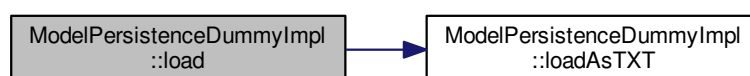
5.41.2.1 bool [ModelPersistenceDummyImpl::isSaved](#) ( ) [virtual]

Implements [ModelPersistence\\_if](#).

5.41.2.2 bool [ModelPersistenceDummyImpl::load](#) ( std::string *filename* ) [virtual]

Implements [ModelPersistence\\_if](#).

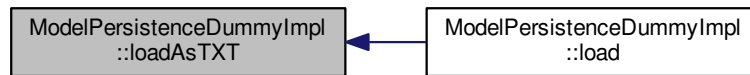
Here is the call graph for this function:



#### 5.41.2.3 bool ModelPersistenceDummyImpl::loadAsTXT ( std::string *filename* ) [virtual]

Implements [ModelPersistence\\_if](#).

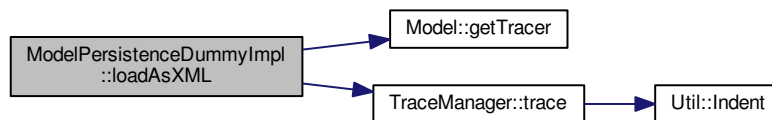
Here is the caller graph for this function:



#### 5.41.2.4 bool ModelPersistenceDummyImpl::loadAsXML ( std::string *filename* ) [virtual]

Implements [ModelPersistence\\_if](#).

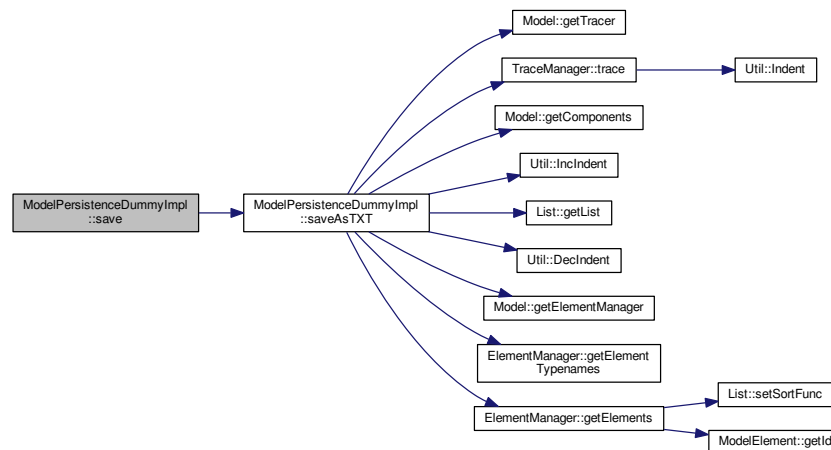
Here is the call graph for this function:



#### 5.41.2.5 bool ModelPersistenceDummyImpl::save ( std::string *filename* ) [virtual]

Implements [ModelPersistence\\_if](#).

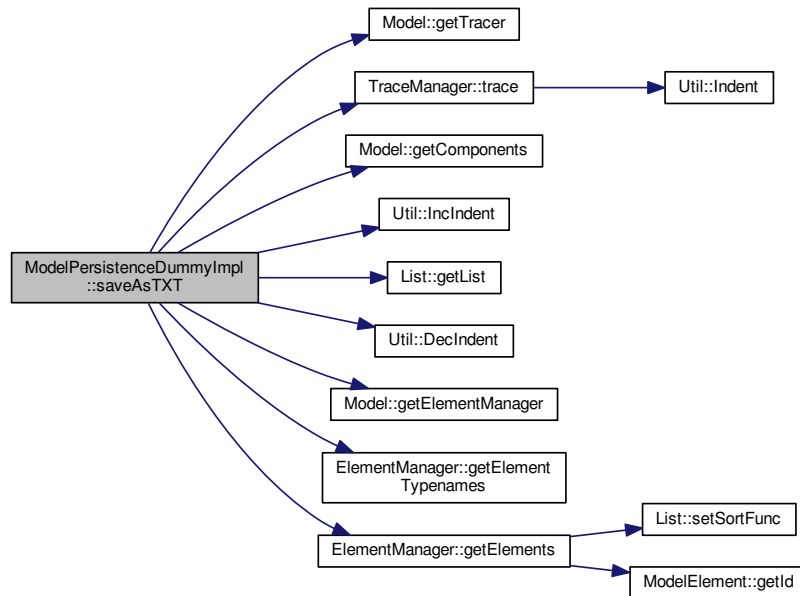
Here is the call graph for this function:



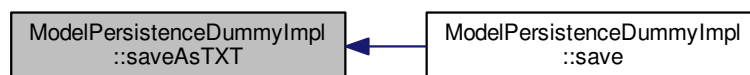
5.41.2.6 `bool ModelPersistenceDummyImpl::saveAsTXT ( std::string filename )` `[virtual]`

Implements [ModelPersistence\\_if](#).

Here is the call graph for this function:



Here is the caller graph for this function:



5.41.2.7 `bool ModelPersistenceDummyImpl::saveAsXML ( std::string filename )` `[virtual]`

Implements [ModelPersistence\\_if](#).

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

- [ModelPersistenceDummyImpl.h](#)
- [ModelPersistenceDummyImpl.cpp](#)

## 5.42 ModelSimulation Class Reference

```
#include <ModelSimulation.h>
```

### Public Member Functions

- [ModelSimulation](#) ([Model](#) \*model)
- [ModelSimulation](#) (const [ModelSimulation](#) &orig)
- virtual [~ModelSimulation](#) ()
- void [startSimulation](#) ()
  - Starts a sequential execution of a simulation, ie, a set of replications of this model.*
- void [pauseSimulation](#) ()
- void [stepSimulation](#) ()
  - Executes the processing of a single event, the next one in the future events list.*
- void [stopSimulation](#) ()
- void [restartSimulation](#) ()
- void [setPauseOnEvent](#) (bool \_pauseOnEvent)
- bool [isPauseOnEvent](#) () const
- void [setStepByStep](#) (bool \_stepByStep)
- bool [isStepByStep](#) () const
- void [setInitializeStatistics](#) (bool \_initializeStatistics)
- bool [isInitializeStatistics](#) () const
- void [setInitializeSystem](#) (bool \_initializeSystem)
- bool [isInitializeSystem](#) () const
- void [setPauseOnReplication](#) (bool \_pauseBetweenReplications)
- bool [isPauseOnReplication](#) () const
- double [getSimulatedTime](#) () const
- bool [isRunning](#) () const
- unsigned int [getCurrentReplicationNumber](#) () const
- [ModelComponent](#) \* [getCurrentComponent](#) () const
- [Entity](#) \* [getCurrentEntity](#) () const

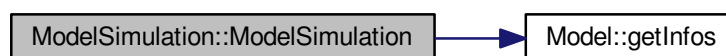
### 5.42.1 Detailed Description

The [ModelSimulation](#) controls the simulation of a model, allowing to start, pause, resume e stop a simulation, composed by a set of replications.

### 5.42.2 Constructor & Destructor Documentation

#### 5.42.2.1 [ModelSimulation::ModelSimulation](#) ( [Model](#) \* model )

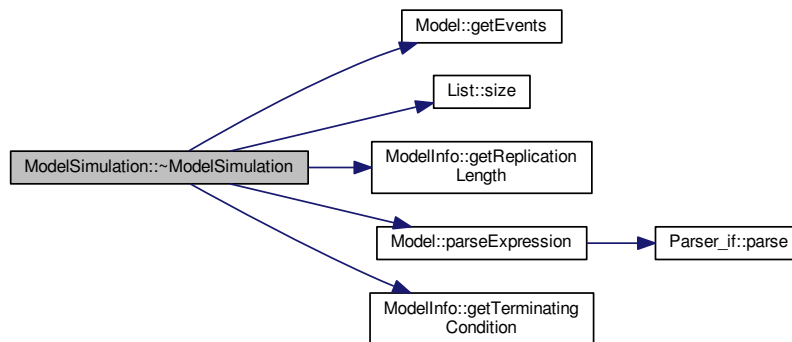
Here is the call graph for this function:



5.42.2.2 `ModelSimulation::ModelSimulation ( const ModelSimulation & orig )`

5.42.2.3 `ModelSimulation::~~ModelSimulation ( ) [virtual]`

Here is the call graph for this function:



### 5.42.3 Member Function Documentation

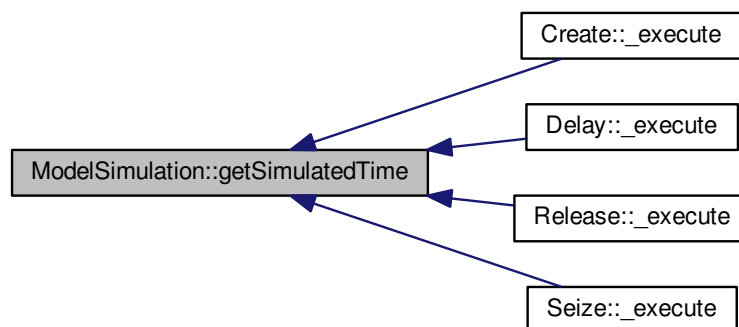
5.42.3.1 `ModelComponent * ModelSimulation::getCurrentComponent ( ) const`

5.42.3.2 `Entity * ModelSimulation::getCurrentEntity ( ) const`

5.42.3.3 `unsigned int ModelSimulation::getCurrentReplicationNumber ( ) const`

5.42.3.4 `double ModelSimulation::getSimulatedTime ( ) const`

Here is the caller graph for this function:



5.42.3.5 `bool ModelSimulation::isInitializeStatistics ( ) const`

5.42.3.6 `bool ModelSimulation::isInitializeSystem ( ) const`

5.42.3.7 `bool ModelSimulation::isPauseOnEvent ( ) const`

5.42.3.8 `bool ModelSimulation::isPauseOnReplication ( ) const`

5.42.3.9 `bool ModelSimulation::isRunning ( ) const`

The current time in the model being simulated, i.e., the instant when the current event was triggered

5.42.3.10 `bool ModelSimulation::isStepByStep ( ) const`

5.42.3.11 `void ModelSimulation::pauseSimulation ( )`

5.42.3.12 `void ModelSimulation::restartSimulation ( )`

5.42.3.13 `void ModelSimulation::setInitializeStatistics ( bool _initializeStatistics )`

5.42.3.14 `void ModelSimulation::setInitializeSystem ( bool _initializeSystem )`

5.42.3.15 `void ModelSimulation::setPauseOnEvent ( bool _pauseOnEvent )`

5.42.3.16 `void ModelSimulation::setPauseOnReplication ( bool _pauseBetweenReplications )`

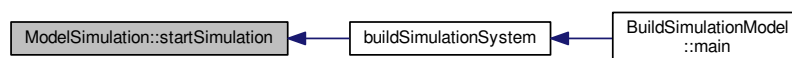
5.42.3.17 `void ModelSimulation::setStepByStep ( bool _stepByStep )`

5.42.3.18 `void ModelSimulation::startSimulation ( )`

Starts a sequential execution of a simulation, ie, a set of replications of this model.

Checks the model and if ok then initialize the simulation, execute repeatedly each replication and then show simulation statistics

Here is the caller graph for this function:

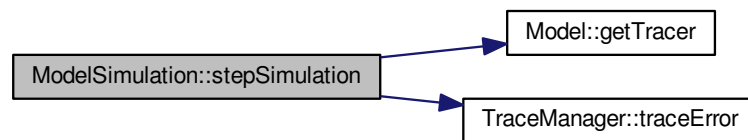




## 5.42.3.19 void ModelSimulation::stepSimulation ( )

Executes the processing of a single event, the next one in the future events list.

Here is the call graph for this function:



## 5.42.3.20 void ModelSimulation::stopSimulation ( )

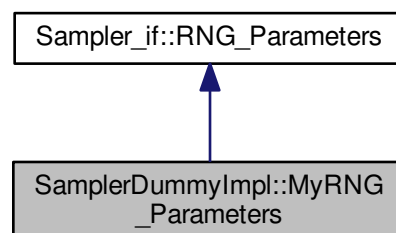
The documentation for this class was generated from the following files:

- [ModelSimulation.h](#)
- [ModelSimulation.cpp](#)

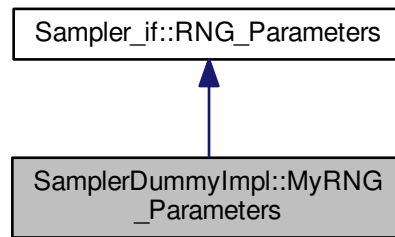
## 5.43 SamplerDummyImpl::MyRNG\_Parameters Class Reference

```
#include <SamplerDummyImpl.h>
```

Inheritance diagram for `SamplerDummyImpl::MyRNG_Parameters`:



Collaboration diagram for `SamplerDummyImpl::MyRNG_Parameters`:



### Public Attributes

- unsigned int [seed](#)
- unsigned int [module](#)
- unsigned int [multiplier](#)

### 5.43.1 Member Data Documentation

5.43.1.1 unsigned int `SamplerDummyImpl::MyRNG_Parameters::module`

5.43.1.2 unsigned int `SamplerDummyImpl::MyRNG_Parameters::multiplier`

5.43.1.3 unsigned int `SamplerDummyImpl::MyRNG_Parameters::seed`

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

- [SamplerDummyImpl.h](#)

## 5.44 OnEventManager Class Reference

```
#include <OnEventManager.h>
```

### Public Member Functions

- [OnEventManager](#) ()
- [OnEventManager](#) (const [OnEventManager](#) &orig)
- virtual [~OnEventManager](#) ()
- void [addOnReplicationStartHandler](#) ([simulationEventHandler](#) EventHandler)
- void [addOnReplicationStepHandler](#) ([simulationEventHandler](#) EventHandler)
- void [addOnReplicationEndHandler](#) ([simulationEventHandler](#) EventHandler)
- void [addOnProcessEventHandler](#) ([simulationEventHandler](#) EventHandler)
- void [addOnSimulationStartHandler](#) ([simulationEventHandler](#) EventHandler)
- void [addOnSimulationEndHandler](#) ([simulationEventHandler](#) EventHandler)
- void [NotifyReplicationStartHandlers](#) ([SimulationEvent](#) \*se)
- void [NotifyReplicationStepHandlers](#) ([SimulationEvent](#) \*se)
- void [NotifyReplicationEndHandlers](#) ([SimulationEvent](#) \*se)
- void [NotifyProcessEventHandlers](#) ([SimulationEvent](#) \*se)
- void [NotifySimulationStartHandlers](#) ([SimulationEvent](#) \*se)
- void [NotifySimulationEndHandlers](#) ([SimulationEvent](#) \*se)

### 5.44.1 Detailed Description

[OnEventManager](#) allows external methods to hook interval simulation events as listeners (or observers) of specific events. All methods added as listeners of an event will be invoked when that event is triggered.

### 5.44.2 Constructor & Destructor Documentation

5.44.2.1 `OnEventManager::OnEventManager ( )`

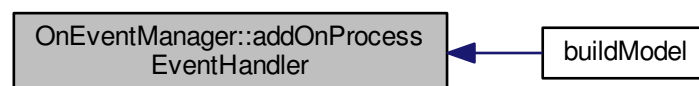
5.44.2.2 `OnEventManager::OnEventManager ( const OnEventManager & orig )`

5.44.2.3 `OnEventManager::~OnEventManager ( ) [virtual]`

### 5.44.3 Member Function Documentation

5.44.3.1 `void OnEventManager::addOnProcessEventHandler ( simulationEventHandler EventHandler )`

Here is the caller graph for this function:



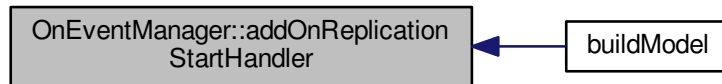
5.44.3.2 `void OnEventManager::addOnReplicationEndHandler ( simulationEventHandler EventHandler )`

Here is the caller graph for this function:



5.44.3.3 void OnEventManager::addOnReplicationStartHandler ( simulationEventHandler *EventHandler* )

Here is the caller graph for this function:

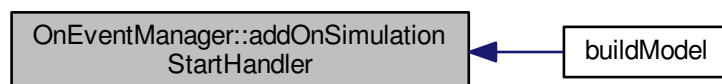


5.44.3.4 void OnEventManager::addOnReplicationStepHandler ( simulationEventHandler *EventHandler* )

5.44.3.5 void OnEventManager::addOnSimulationEndHandler ( simulationEventHandler *EventHandler* )

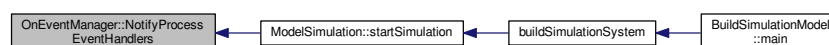
5.44.3.6 void OnEventManager::addOnSimulationStartHandler ( simulationEventHandler *EventHandler* )

Here is the caller graph for this function:



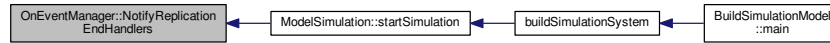
5.44.3.7 void OnEventManager::NotifyProcessEventHandlers ( SimulationEvent \* *se* )

Here is the caller graph for this function:



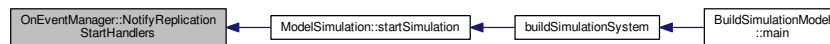
#### 5.44.3.8 void OnEventManager::NotifyReplicationEndHandlers ( SimulationEvent \* se )

Here is the caller graph for this function:



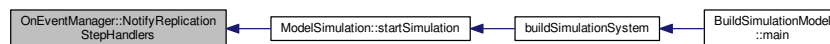
#### 5.44.3.9 void OnEventManager::NotifyReplicationStartHandlers ( SimulationEvent \* se )

Here is the caller graph for this function:



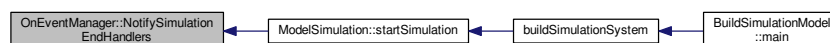
#### 5.44.3.10 void OnEventManager::NotifyReplicationStepHandlers ( SimulationEvent \* se )

Here is the caller graph for this function:



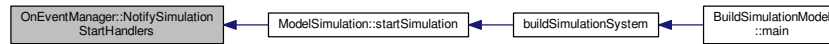
#### 5.44.3.11 void OnEventManager::NotifySimulationEndHandlers ( SimulationEvent \* se )

Here is the caller graph for this function:



#### 5.44.3.12 void OnEventManager::NotifySimulationStartHandlers ( SimulationEvent \* se )

Here is the caller graph for this function:



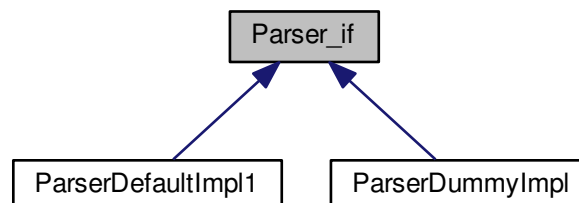
The documentation for this class was generated from the following files:

- [OnEventManager.h](#)
- [OnEventManager.cpp](#)

## 5.45 Parser\_if Class Reference

```
#include <Parser_if.h>
```

Inheritance diagram for Parser\_if:



### Public Member Functions

- virtual double [parse](#) (const std::string expression)=0
- virtual double [parse](#) (const std::string expression, bool \*success, std::string \*errorMessage)=0
- virtual std::string \* [getErrorMessage](#) ()=0

### 5.45.1 Member Function Documentation

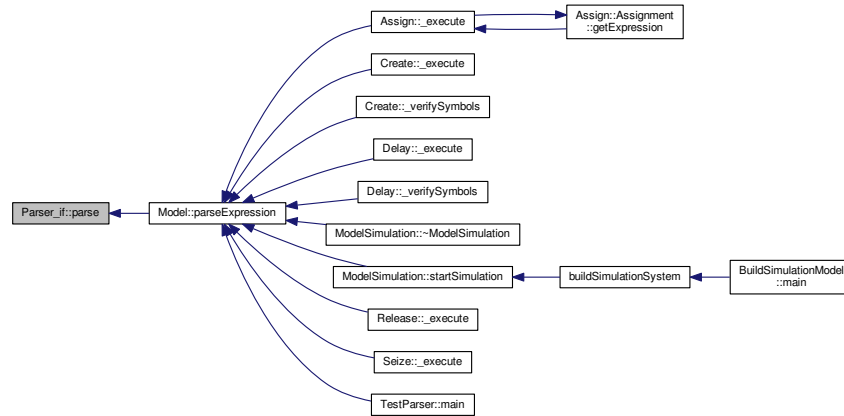
#### 5.45.1.1 virtual std::string\* Parser\_if::getErrorMessage ( ) [pure virtual]

Implemented in [ParserDummyImpl](#), and [ParserDefaultImpl1](#).

5.45.1.2 `virtual double Parser_if::parse ( const std::string expression )` [pure virtual]

Implemented in [ParserDummyImpl](#), and [ParserDefaultImpl1](#).

Here is the caller graph for this function:



5.45.1.3 `virtual double Parser_if::parse ( const std::string expression, bool * success, std::string * errorMessage )` [pure virtual]

Implemented in [ParserDummyImpl](#), and [ParserDefaultImpl1](#).

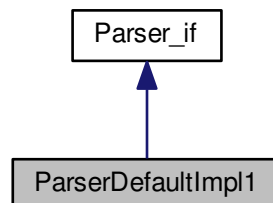
The documentation for this class was generated from the following file:

- [Parser\\_if.h](#)

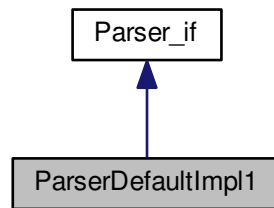
## 5.46 ParserDefaultImpl1 Class Reference

```
#include <ParserDefaultImpl1.h>
```

Inheritance diagram for `ParserDefaultImpl1`:



Collaboration diagram for ParserDefaultImpl1:



## Public Member Functions

- `ParserDefaultImpl1` (`Model *model`)
- `ParserDefaultImpl1` (`const ParserDefaultImpl1 &orig`)
- `virtual ~ParserDefaultImpl1` ()
- `double parse` (`const std::string expression`)
- `double parse` (`const std::string expression, bool *success, std::string *errorMessage`)
- `std::string * getErrorMessage` ()

### 5.46.1 Constructor & Destructor Documentation

5.46.1.1 `ParserDefaultImpl1::ParserDefaultImpl1 ( Model * model )`

5.46.1.2 `ParserDefaultImpl1::ParserDefaultImpl1 ( const ParserDefaultImpl1 & orig )`

5.46.1.3 `ParserDefaultImpl1::~~ParserDefaultImpl1 ( )` [virtual]

### 5.46.2 Member Function Documentation

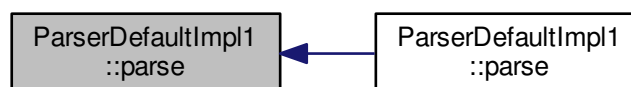
5.46.2.1 `std::string * ParserDefaultImpl1::getErrorMessage ( )` [virtual]

Implements `Parser_if`.

5.46.2.2 `double ParserDefaultImpl1::parse ( const std::string expression )` [virtual]

Implements `Parser_if`.

Here is the caller graph for this function:

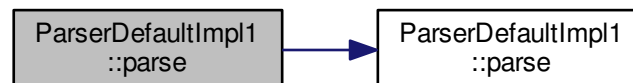




5.46.2.3 `double ParserDefaultImpl1::parse ( const std::string expression, bool * success, std::string * errorMessage )`  
[virtual]

Implements [Parser\\_if](#).

Here is the call graph for this function:



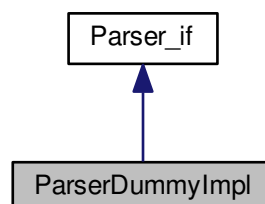
The documentation for this class was generated from the following files:

- [ParserDefaultImpl1.h](#)
- [ParserDefaultImpl1.cpp](#)

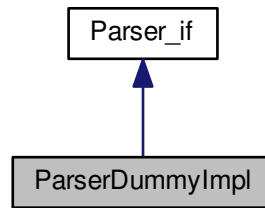
## 5.47 ParserDummyImpl Class Reference

```
#include <ParserDummyImpl.h>
```

Inheritance diagram for ParserDummyImpl:



Collaboration diagram for ParserDummyImpl:



## Public Member Functions

- [ParserDummyImpl](#) ([Model](#) \*model)
- [ParserDummyImpl](#) (const [ParserDummyImpl](#) &orig)
- virtual [~ParserDummyImpl](#) ()
- double [parse](#) (const std::string expression)
- double [parse](#) (const std::string expression, bool \*success, std::string \*errorMessage)
- std::string \* [getErrorMessage](#) ()

### 5.47.1 Constructor & Destructor Documentation

5.47.1.1 `ParserDummyImpl::ParserDummyImpl ( Model * model )`

5.47.1.2 `ParserDummyImpl::ParserDummyImpl ( const ParserDummyImpl & orig )`

5.47.1.3 `ParserDummyImpl::~~ParserDummyImpl ( )` [virtual]

### 5.47.2 Member Function Documentation

5.47.2.1 `std::string * ParserDummyImpl::getErrorMessage ( )` [virtual]

Implements [Parser\\_if](#).

5.47.2.2 `double ParserDummyImpl::parse ( const std::string expression )` [virtual]

Implements [Parser\\_if](#).

Here is the caller graph for this function:



5.47.2.3 `double ParserDummyImpl::parse ( const std::string expression, bool * success, std::string * errorMessage )`  
`[virtual]`

Implements [Parser\\_if](#).

Here is the call graph for this function:



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

- [ParserDummyImpl.h](#)
- [ParserDummyImpl.cpp](#)

## 5.48 Plugin Class Reference

```
#include <Plugin.h>
```

### Public Member Functions

- [Plugin](#) (std::string name, bool source, bool drain)
- [Plugin](#) (const [Plugin](#) &orig)
- virtual [~Plugin](#) ()
- bool [isDrain](#) () const
- bool [isSource](#) () const

### 5.48.1 Detailed Description

A [Plugin](#) represents a dynamically linked component class ([ModelComponent](#)) or element class ([ModelElement](#)); It gives access to a [ModelComponent](#) so it can be used by the model. Classes like [Create](#), [Delay](#), and [Dispose](#) are examples of Plugins. It corresponds directly to the "Expansible" part (the capitalized 'E') of the GenESyS acronymous Plugins are NOT implemented yet

## 5.48.2 Constructor & Destructor Documentation

5.48.2.1 `Plugin::Plugin ( std::string name, bool source, bool drain )`

5.48.2.2 `Plugin::Plugin ( const Plugin & orig )`

5.48.2.3 `Plugin::~~Plugin ( )` `[virtual]`

## 5.48.3 Member Function Documentation

5.48.3.1 `bool Plugin::isDrain ( ) const`

5.48.3.2 `bool Plugin::isSource ( ) const`

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

- [Plugin.h](#)
- [Plugin.cpp](#)

## 5.49 ProbDistrib Class Reference

```
#include <ProbDistrib.h>
```

### Static Public Member Functions

- static double [uniform](#) (double x, double min, double max)
- static double [exponential](#) (double x, double mean)
- static double [erlang](#) (double x, double mean, double M)
- static double [normal](#) (double x, double mean, double stddev)
- static double [gamma](#) (double x, double mean, double alpha)
- static double [beta](#) (double x, double alpha, double beta)
- static double [weibull](#) (double x, double alpha, double scale)
- static double [logNormal](#) (double x, double mean, double stddev)
- static double [triangular](#) (double x, double min, double mode, double max)

### 5.49.1 Member Function Documentation

5.49.1.1 `double ProbDistrib::beta ( double x, double alpha, double beta )` [static]

5.49.1.2 `double ProbDistrib::erlang ( double x, double mean, double M )` [static]

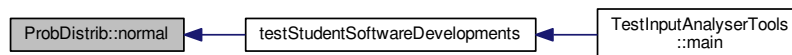
5.49.1.3 `double ProbDistrib::exponential ( double x, double mean )` [static]

5.49.1.4 `double ProbDistrib::gamma ( double x, double mean, double alpha )` [static]

5.49.1.5 `double ProbDistrib::logNormal ( double x, double mean, double stddev )` [static]

5.49.1.6 `double ProbDistrib::normal ( double x, double mean, double stddev )` [static]

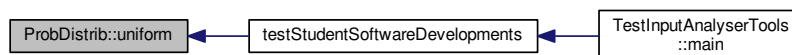
Here is the caller graph for this function:



5.49.1.7 `double ProbDistrib::triangular ( double x, double min, double mode, double max )` [static]

5.49.1.8 `double ProbDistrib::uniform ( double x, double min, double max )` [static]

Here is the caller graph for this function:



5.49.1.9 `double ProbDistrib::weibull ( double x, double alpha, double scale )` [static]

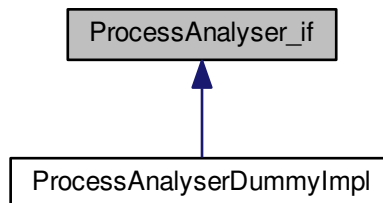
The documentation for this class was generated from the following files:

- [ProbDistrib.h](#)
- [ProbDistrib.cpp](#)

## 5.50 ProcessAnalyser\_if Class Reference

```
#include <ProcessAnalyser_if.h>
```

Inheritance diagram for ProcessAnalyser\_if:



### Public Member Functions

- virtual [List< SimulationScenario \\* >](#) \* [getScenarios](#) () const =0
- virtual [List< SimulationControl \\* >](#) \* [getControls](#) () const =0
- virtual [List< SimulationResponse \\* >](#) \* [getResponses](#) () const =0
- virtual [List< SimulationControl \\* >](#) \* [extractControlsFromModel](#) (std::string modelFilename) const =0
- virtual [List< SimulationResponse \\* >](#) \* [extractResponsesFromModel](#) (std::string modelFilename) const =0
- virtual void [startSimulationOfScenario](#) (SimulationScenario \*scenario)=0
- virtual void [startSimulation](#) ()=0
- virtual void [stopSimulation](#) ()=0
- virtual void [addTraceSimulationHandler](#) (traceSimulationProcessListener traceSimulationProcessListener)=0

### 5.50.1 Detailed Description

The process analyser allows to extract controls and responses from a model, include some of then as controls and responses for a set of scenarios to be simulated

### 5.50.2 Member Function Documentation

**5.50.2.1** virtual void `ProcessAnalyser_if::addTraceSimulationHandler` ( `traceSimulationProcessListener` `traceSimulationProcessListener` ) `[pure virtual]`

Implemented in [ProcessAnalyserDummyImpl](#).

**5.50.2.2** virtual `List<SimulationControl*>` \* `ProcessAnalyser_if::extractControlsFromModel` ( `std::string modelFilename` ) `const` `[pure virtual]`

Implemented in [ProcessAnalyserDummyImpl](#).

5.50.2.3 `virtual List<SimulationResponse*>* ProcessAnalyser_if::extractResponsesFromModel ( std::string  
modelFilename ) const [pure virtual]`

Implemented in [ProcessAnalyserDummyImpl](#).

5.50.2.4 `virtual List<SimulationControl*>* ProcessAnalyser_if::getControls ( ) const [pure virtual]`

Implemented in [ProcessAnalyserDummyImpl](#).

5.50.2.5 `virtual List<SimulationResponse*>* ProcessAnalyser_if::getResponses ( ) const [pure virtual]`

Implemented in [ProcessAnalyserDummyImpl](#).

5.50.2.6 `virtual List<SimulationScenario*>* ProcessAnalyser_if::getScenarios ( ) const [pure virtual]`

Implemented in [ProcessAnalyserDummyImpl](#).

5.50.2.7 `virtual void ProcessAnalyser_if::startSimulation ( ) [pure virtual]`

Implemented in [ProcessAnalyserDummyImpl](#).

5.50.2.8 `virtual void ProcessAnalyser_if::startSimulationOfScenario ( SimulationScenario * scenario ) [pure  
virtual]`

Implemented in [ProcessAnalyserDummyImpl](#).

5.50.2.9 `virtual void ProcessAnalyser_if::stopSimulation ( ) [pure virtual]`

Implemented in [ProcessAnalyserDummyImpl](#).

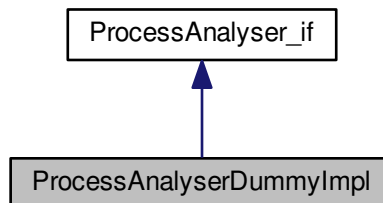
The documentation for this class was generated from the following file:

- [ProcessAnalyser\\_if.h](#)

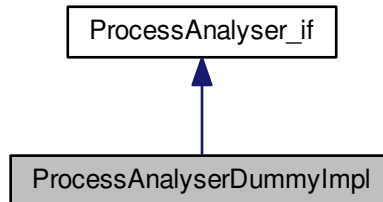
## 5.51 ProcessAnalyserDummyImpl Class Reference

```
#include <ProcessAnalyserDummyImpl.h>
```

Inheritance diagram for ProcessAnalyserDummyImpl:



Collaboration diagram for ProcessAnalyserDummyImpl:



### Public Member Functions

- [ProcessAnalyserDummyImpl \(\)](#)
- [ProcessAnalyserDummyImpl \(const \[ProcessAnalyserDummyImpl\]\(#\) &orig\)](#)
- [~ProcessAnalyserDummyImpl \(\)](#)
- [List< \[SimulationScenario\]\(#\) \\* > \\* \[getScenarios\]\(#\) \(\) const](#)
- [List< \[SimulationControl\]\(#\) \\* > \\* \[getControls\]\(#\) \(\) const](#)
- [List< \[SimulationResponse\]\(#\) \\* > \\* \[getResponses\]\(#\) \(\) const](#)
- [List< \[SimulationControl\]\(#\) \\* > \\* \[extractControlsFromModel\]\(#\) \(std::string modelFilename\) const](#)
- [List< \[SimulationResponse\]\(#\) \\* > \\* \[extractResponsesFromModel\]\(#\) \(std::string modelFilename\) const](#)
- void [startSimulationOfScenario](#) ([SimulationScenario](#) \*scenario)
- void [startSimulation](#) ()
- void [stopSimulation](#) ()
- void [addTraceSimulationHandler](#) ([traceSimulationProcessListener](#) [traceSimulationProcessListener](#))



### 5.51.1 Constructor & Destructor Documentation

5.51.1.1 `ProcessAnalyserDummyImpl::ProcessAnalyserDummyImpl ( )`

5.51.1.2 `ProcessAnalyserDummyImpl::ProcessAnalyserDummyImpl ( const ProcessAnalyserDummyImpl & orig )`

5.51.1.3 `ProcessAnalyserDummyImpl::~~ProcessAnalyserDummyImpl ( )`

### 5.51.2 Member Function Documentation

5.51.2.1 `void ProcessAnalyserDummyImpl::addTraceSimulationHandler ( traceSimulationProcessListener  
traceSimulationProcessListener ) [virtual]`

Implements [ProcessAnalyser\\_if](#).

5.51.2.2 `List< SimulationControl * > * ProcessAnalyserDummyImpl::extractControlsFromModel ( std::string  
modelName ) const [virtual]`

Implements [ProcessAnalyser\\_if](#).

5.51.2.3 `List< SimulationResponse * > * ProcessAnalyserDummyImpl::extractResponsesFromModel ( std::string  
modelName ) const [virtual]`

Implements [ProcessAnalyser\\_if](#).

5.51.2.4 `List< SimulationControl * > * ProcessAnalyserDummyImpl::getControls ( ) const [virtual]`

Implements [ProcessAnalyser\\_if](#).

5.51.2.5 `List< SimulationResponse * > * ProcessAnalyserDummyImpl::getResponses ( ) const [virtual]`

Implements [ProcessAnalyser\\_if](#).

5.51.2.6 `List< SimulationScenario * > * ProcessAnalyserDummyImpl::getScenarios ( ) const [virtual]`

Implements [ProcessAnalyser\\_if](#).

5.51.2.7 `void ProcessAnalyserDummyImpl::startSimulation ( ) [virtual]`

Implements [ProcessAnalyser\\_if](#).

5.51.2.8 `void ProcessAnalyserDummyImpl::startSimulationOfScenario ( SimulationScenario * scenario )` [virtual]

Implements [ProcessAnalyser\\_if](#).

5.51.2.9 `void ProcessAnalyserDummyImpl::stopSimulation ( )` [virtual]

Implements [ProcessAnalyser\\_if](#).

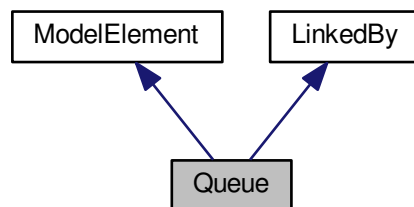
The documentation for this class was generated from the following files:

- [ProcessAnalyserDummyImpl.h](#)
- [ProcessAnalyserDummyImpl.cpp](#)

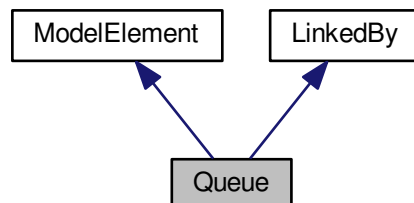
## 5.52 Queue Class Reference

```
#include <Queue.h>
```

Inheritance diagram for Queue:



Collaboration diagram for Queue:



## Public Types

- enum `OrderRule` : int { `OrderRule::FIFO` = 1, `OrderRule::LIFO` = 2, `OrderRule::HIGHESTVALUE` = 3, `OrderRule::SMALLESTVALUE` = 4 }

## Public Member Functions

- `Queue` (`ElementManager` \*elems)
- `Queue` (`ElementManager` \*elems, std::string name)
- `Queue` (const `Queue` &orig)
- virtual `~Queue` ()
- virtual std::string `show` ()
- void `insertElement` (`Waiting` \*element)
- void `removeElement` (`Waiting` \*element, double tnow)
- unsigned int `size` ()
- `Waiting` \* `first` ()
- void `setAttributeName` (std::string \_attributeName)
- std::string `getAttributeName` () const
- void `setOrderRule` (`OrderRule` \_orderRule)
- `Queue::OrderRule` `getOrderRule` () const

## Protected Member Functions

- virtual void `_loadInstance` (std::list< std::string > words)
- virtual std::list< std::string > \* `_saveInstance` ()
- virtual bool `_verifySymbols` (std::string \*errorMessage)

## Additional Inherited Members

### 5.52.1 Member Enumeration Documentation

5.52.1.1 enum `Queue::OrderRule` : int [strong]

Enumerator

***FIFO***

***LIFO***

***HIGHESTVALUE***

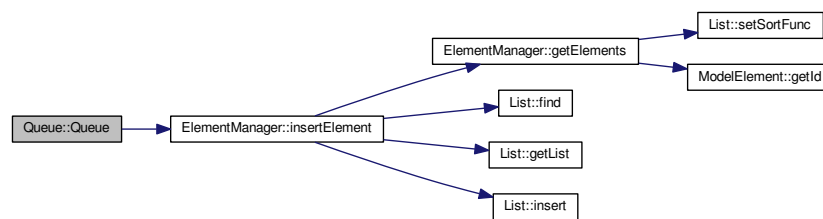
***SMALLESTVALUE***

## 5.52.2 Constructor & Destructor Documentation

### 5.52.2.1 Queue::Queue ( ElementManager \* *elems* )

### 5.52.2.2 Queue::Queue ( ElementManager \* *elems*, std::string *name* )

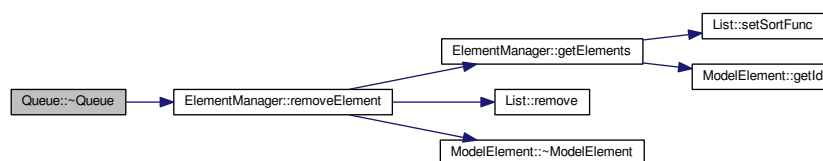
Here is the call graph for this function:



### 5.52.2.3 Queue::Queue ( const Queue & *orig* )

### 5.52.2.4 Queue::~~Queue ( ) [virtual]

Here is the call graph for this function:



## 5.52.3 Member Function Documentation

### 5.52.3.1 void Queue::\_loadInstance ( std::list< std::string > *words* ) [protected], [virtual]

Implements [ModelElement](#).

5.52.3.2 `std::list< std::string > * Queue::_saveInstance ( )` [protected],[virtual]

Reimplemented from [ModelElement](#).

Here is the call graph for this function:



5.52.3.3 `bool Queue::_verifySymbols ( std::string * errorMessage )` [protected],[virtual]

Implements [ModelElement](#).

5.52.3.4 `Waiting * Queue::first ( )`

Here is the call graph for this function:



Here is the caller graph for this function:

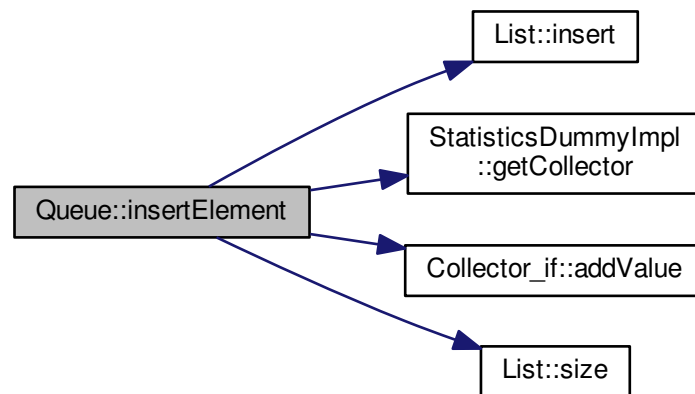


5.52.3.5 `std::string Queue::getAttributeName ( ) const`

5.52.3.6 `Queue::OrderRule Queue::getOrderRule ( ) const`

5.52.3.7 `void Queue::insertElement ( Waiting * element )`

Here is the call graph for this function:

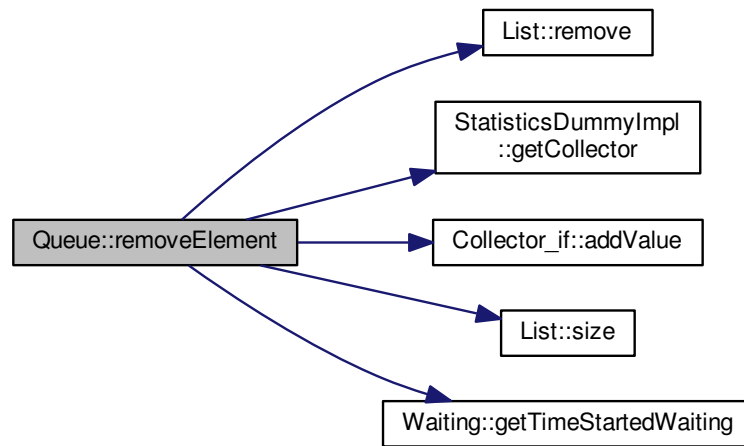


Here is the caller graph for this function:



5.52.3.8 void Queue::removeElement ( Waiting \* *element*, double *tnow* )

Here is the call graph for this function:



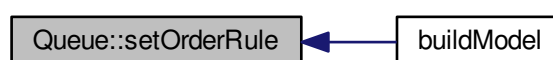
Here is the caller graph for this function:



5.52.3.9 void Queue::setAttributeName ( std::string *\_attributeName* )

5.52.3.10 void Queue::setOrderRule ( OrderRule *\_orderRule* )

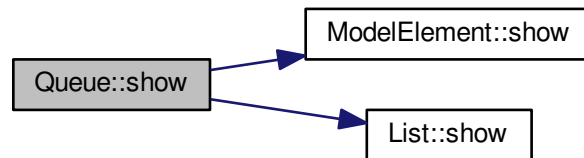
Here is the caller graph for this function:



#### 5.52.3.11 `std::string Queue::show ( )` [virtual]

Reimplemented from [ModelElement](#).

Here is the call graph for this function:

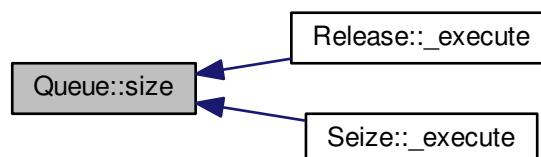


#### 5.52.3.12 `unsigned int Queue::size ( )`

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:

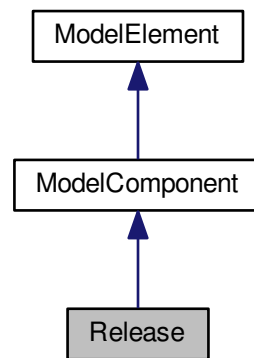
- [Queue.h](#)
- [Queue.cpp](#)



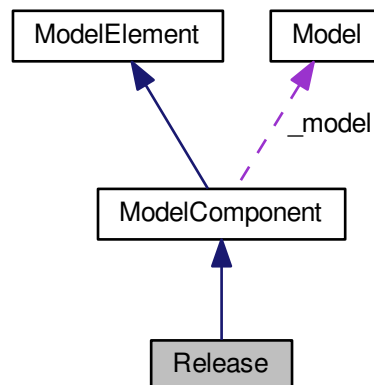
## 5.53 Release Class Reference

```
#include <Release.h>
```

Inheritance diagram for Release:



Collaboration diagram for Release:



### Public Member Functions

- [Release](#) ([Model](#) \*model)
- [Release](#) (const [Release](#) &orig)
- virtual [~Release](#) ()
- virtual std::string [show](#) ()

- void [setPriority](#) (unsigned short \_priority)
- unsigned short [getPriority](#) () const
- void [setResourceType](#) ([Resource::ResourceType](#) \_resourceType)
- [Resource::ResourceType](#) [getResourceType](#) () const
- void [setQuantity](#) (std::string \_quantity)
- std::string [getQuantity](#) () const
- void [setRule](#) ([Resource::ResourceRule](#) \_rule)
- [Resource::ResourceRule](#) [getRule](#) () const
- void [setSaveAttribute](#) (std::string \_saveAttribute)
- std::string [getSaveAttribute](#) () const
- void [setResource](#) ([Resource](#) \*\_resource)
- [Resource](#) \* [getResource](#) () const

### Protected Member Functions

- virtual void [\\_execute](#) ([Entity](#) \*entity)
- virtual void [\\_loadInstance](#) (std::list< std::string > words)
- virtual std::list< std::string > \* [\\_saveInstance](#) ()
- virtual bool [\\_verifySymbols](#) (std::string \*errorMessage)

### Additional Inherited Members

#### 5.53.1 Constructor & Destructor Documentation

5.53.1.1 [Release::Release](#) ( [Model](#) \* *model* )

5.53.1.2 [Release::Release](#) ( const [Release](#) & *orig* )

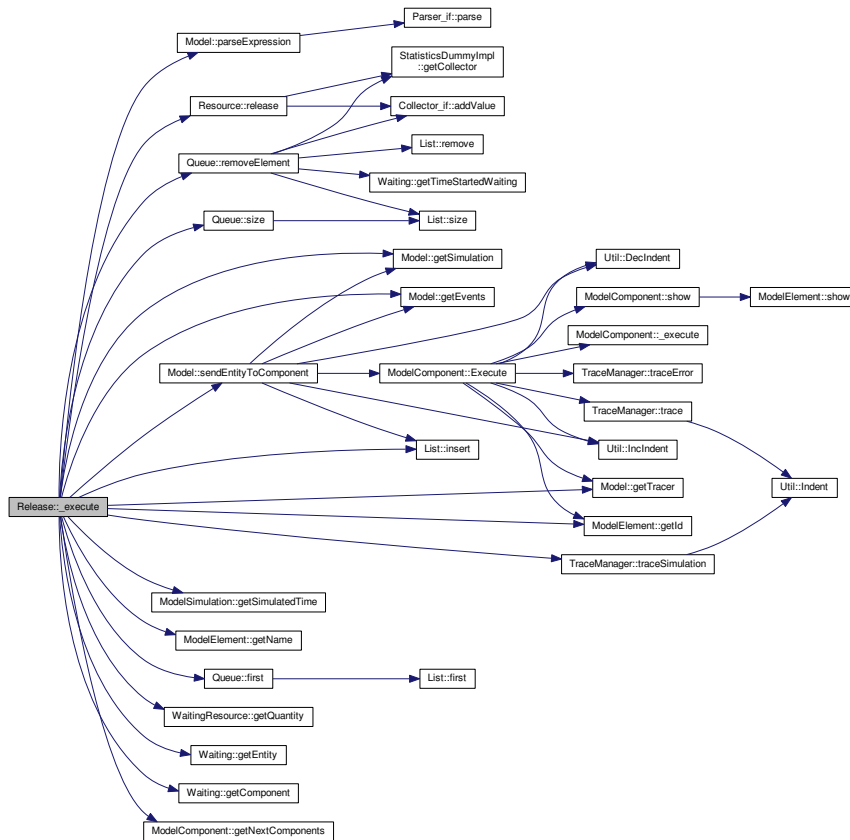
5.53.1.3 [Release::~~Release](#) ( ) [virtual]

#### 5.53.2 Member Function Documentation

5.53.2.1 void [Release::\\_execute](#) ( [Entity](#) \* *entity* ) [protected],[virtual]

Implements [ModelComponent](#).

Here is the call graph for this function:



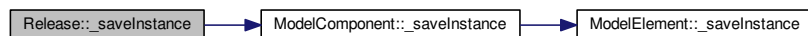
**5.53.2.2** `void Release::_loadInstance ( std::list< std::string > words )` [protected],[virtual]

Implements [ModelElement](#).

**5.53.2.3** `std::list< std::string > * Release::_saveInstance ( )` [protected],[virtual]

Reimplemented from [ModelComponent](#).

Here is the call graph for this function:



**5.53.2.4** `bool Release::_verifySymbols ( std::string * errorMessage )` [protected],[virtual]

Implements [ModelElement](#).

5.53.2.5 unsigned short Release::getPriority ( ) const

5.53.2.6 std::string Release::getQuantity ( ) const

5.53.2.7 Resource \* Release::getResource ( ) const

5.53.2.8 Resource::ResourceType Release::getResourceType ( ) const

5.53.2.9 Resource::ResourceRule Release::getRule ( ) const

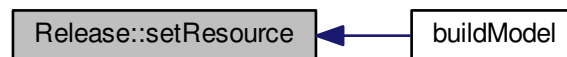
5.53.2.10 std::string Release::getSaveAttribute ( ) const

5.53.2.11 void Release::setPriority ( unsigned short *\_priority* )

5.53.2.12 void Release::setQuantity ( std::string *\_quantity* )

5.53.2.13 void Release::setResource ( Resource \* *\_resource* )

Here is the caller graph for this function:



5.53.2.14 void Release::setResourceType ( Resource::ResourceType *\_resourceType* )

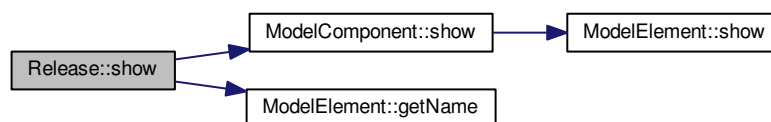
5.53.2.15 void Release::setRule ( Resource::ResourceRule *\_rule* )

5.53.2.16 void Release::setSaveAttribute ( std::string *\_saveAttribute* )

5.53.2.17 std::string Release::show ( ) [virtual]

Reimplemented from [ModelComponent](#).

Here is the call graph for this function:



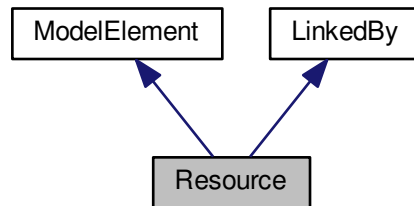
The documentation for this class was generated from the following files:

- [Release.h](#)
- [Release.cpp](#)

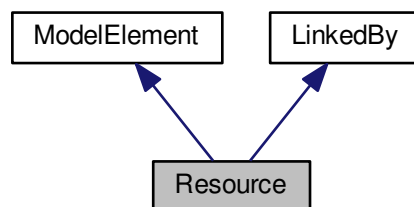
## 5.54 Resource Class Reference

```
#include <Resource.h>
```

Inheritance diagram for Resource:



Collaboration diagram for Resource:



### Public Types

- enum **ResourceType** : int { **ResourceType::SET** = 1, **ResourceType::RESOURCE** = 2 }
- enum **ResourceRule** : int { **ResourceRule::RANDOM** = 1, **ResourceRule::CICLICAL** = 2, **ResourceRule::ESPECIFIC** = 3, **ResourceRule::SMALLESTBUSY** = 4, **ResourceRule::LARGESTREMAININGCAPACITY** = 5 }
- enum **ResourceState** : int { **ResourceState::IDLE** = 1, **ResourceState::BUSY** = 2, **ResourceState::FAILED** = 3, **ResourceState::INACTIVE** = 4, **ResourceState::OTHER** = 5 }

## Public Member Functions

- [Resource](#) ([ElementManager](#) \*elems)
- [Resource](#) ([ElementManager](#) \*elems, std::string name)
- [Resource](#) (const [Resource](#) &orig)
- virtual [~Resource](#) ()
- virtual std::string [show](#) ()
- void [seize](#) (unsigned int quantity, double tnow)
- void [release](#) (unsigned int quantity, double tnow)
- void [setResourceState](#) ([ResourceState](#) \_resourceState)
- [Resource::ResourceState](#) [getResourceState](#) () const
- void [setCapacity](#) (unsigned int \_capacity)
- unsigned int [getCapacity](#) () const
- void [setCostBusyHour](#) (double \_costBusyHour)
- double [getCostBusyHour](#) () const
- void [setCostIdleHour](#) (double \_costIdleHour)
- double [getCostIdleHour](#) () const
- void [setCostPerUse](#) (double \_costPerUse)
- double [getCostPerUse](#) () const
- unsigned int [getNumberBusy](#) () const
- unsigned int [getNumberOut](#) () const

## Protected Member Functions

- virtual void [\\_loadInstance](#) (std::list< std::string > words)
- virtual std::list< std::string > \* [\\_saveInstance](#) ()
- virtual bool [\\_verifySymbols](#) (std::string \*errorMessage)

## Additional Inherited Members

### 5.54.1 Member Enumeration Documentation

5.54.1.1 enum [Resource::ResourceRule](#) : int [strong]

Enumerator

***RANDOM***  
***CICLICAL***  
***ESPECIFIC***  
***SMALLESTBUSY***  
***LARGESTREMAININGCAPACITY***

5.54.1.2 enum [Resource::ResourceState](#) : int [strong]

Enumerator

***IDLE***  
***BUSY***  
***FAILED***  
***INACTIVE***  
***OTHER***

5.54.1.3 `enum Resource::ResourceType : int [strong]`

Enumerator

**SET**

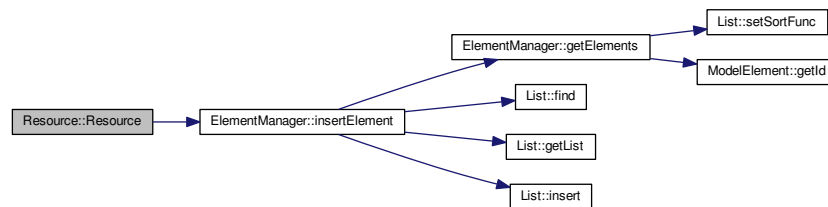
**RESOURCE**

## 5.54.2 Constructor & Destructor Documentation

5.54.2.1 `Resource::Resource ( ElementManager * elems )`

5.54.2.2 `Resource::Resource ( ElementManager * elems, std::string name )`

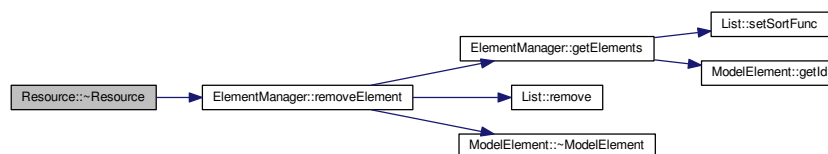
Here is the call graph for this function:



5.54.2.3 `Resource::Resource ( const Resource & orig )`

5.54.2.4 `Resource::~~Resource ( ) [virtual]`

Here is the call graph for this function:



## 5.54.3 Member Function Documentation

5.54.3.1 `void Resource::_loadInstance ( std::list< std::string > words ) [protected], [virtual]`

Implements [ModelElement](#).

5.54.3.2 `std::list< std::string > * Resource::_saveInstance ( )` [protected],[virtual]

Reimplemented from [ModelElement](#).

Here is the call graph for this function:



5.54.3.3 `bool Resource::_verifySymbols ( std::string * errorMessage )` [protected],[virtual]

Implements [ModelElement](#).

5.54.3.4 `unsigned int Resource::getCapacity ( ) const`

Here is the caller graph for this function:



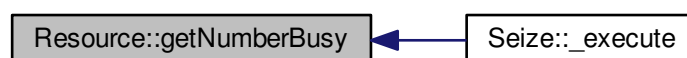
5.54.3.5 `double Resource::getCostBusyHour ( ) const`

5.54.3.6 `double Resource::getCostIdleHour ( ) const`

5.54.3.7 `double Resource::getCostPerUse ( ) const`

5.54.3.8 `unsigned int Resource::getNumberBusy ( ) const`

Here is the caller graph for this function:



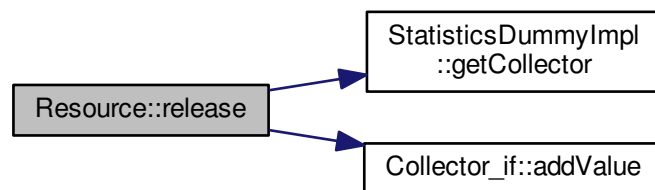


5.54.3.9 unsigned int Resource::getNumberOut ( ) const

5.54.3.10 Resource::ResourceState Resource::getResourceState ( ) const

5.54.3.11 void Resource::release ( unsigned int *quantity*, double *tnow* )

Here is the call graph for this function:



Here is the caller graph for this function:



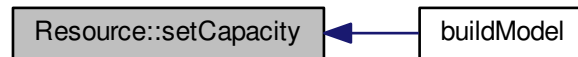
5.54.3.12 void Resource::seize ( unsigned int *quantity*, double *tnow* )

Here is the caller graph for this function:



5.54.3.13 `void Resource::setCapacity ( unsigned int _capacity )`

Here is the caller graph for this function:



5.54.3.14 `void Resource::setCostBusyHour ( double _costBusyHour )`

5.54.3.15 `void Resource::setCostIdleHour ( double _costIdleHour )`

5.54.3.16 `void Resource::setCostPerUse ( double _costPerUse )`

5.54.3.17 `void Resource::setResourceState ( ResourceState _resourceState )`

5.54.3.18 `std::string Resource::show ( ) [virtual]`

Reimplemented from [ModelElement](#).

Here is the call graph for this function:



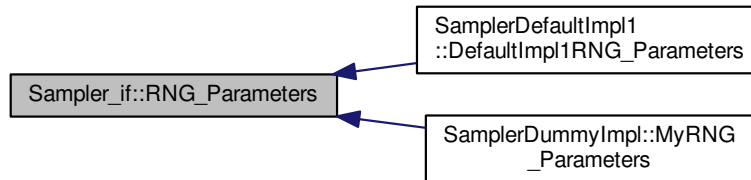
The documentation for this class was generated from the following files:

- [Resource.h](#)
- [Resource.cpp](#)

## 5.55 `Sampler_if::RNG_Parameters` Class Reference

```
#include <Sampler_if.h>
```

Inheritance diagram for `Sampler_if::RNG_Parameters`:



### 5.55.1 Detailed Description

class that encapsulates attributes required to generate random numbers, which depends on the generation method used.

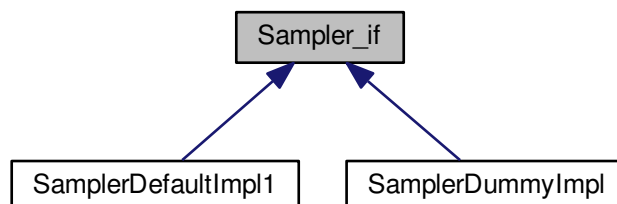
The documentation for this class was generated from the following file:

- [Sampler\\_if.h](#)

## 5.56 `Sampler_if` Class Reference

```
#include <Sampler_if.h>
```

Inheritance diagram for `Sampler_if`:



### Classes

- class [RNG\\_Parameters](#)

## Public Member Functions

- virtual double [random](#) ()=0
- virtual double [sampleUniform](#) (double min, double max)=0
- virtual double [sampleExponential](#) (double mean)=0
- virtual double [sampleErlang](#) (double mean, int M)=0
- virtual double [sampleNormal](#) (double mean, double stddev)=0
- virtual double [sampleGamma](#) (double mean, double alpha)=0
- virtual double [sampleBeta](#) (double alpha, double beta, double infLimit, double supLimit)=0
- virtual double [sampleWeibull](#) (double alpha, double scale)=0
- virtual double [sampleLogNormal](#) (double mean, double stddev)=0
- virtual double [sampleTriangular](#) (double min, double mode, double max)=0
- virtual double [sampleDiscrete](#) (double value, double acumProb,...)=0
- virtual void [setRNGparameters](#) ([RNG\\_Parameters](#) \*param)=0
- virtual [RNG\\_Parameters](#) \* [getRNGparameters](#) () const =0

### 5.56.1 Detailed Description

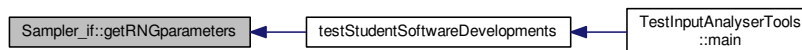
Interface that describes the methods to be implemented by classes that generate random values that follow a specific probability distribution.

### 5.56.2 Member Function Documentation

#### 5.56.2.1 virtual [RNG\\_Parameters](#)\* [Sampler\\_if::getRNGparameters](#) ( ) const [pure virtual]

Implemented in [SamplerDefaultImpl1](#), and [SamplerDummyImpl](#).

Here is the caller graph for this function:



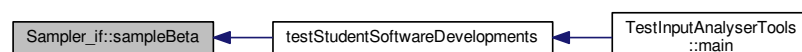
#### 5.56.2.2 virtual double [Sampler\\_if::random](#) ( ) [pure virtual]

Implemented in [SamplerDefaultImpl1](#), and [SamplerDummyImpl](#).

#### 5.56.2.3 virtual double [Sampler\\_if::sampleBeta](#) ( double *alpha*, double *beta*, double *infLimit*, double *supLimit* ) [pure virtual]

Implemented in [SamplerDefaultImpl1](#), and [SamplerDummyImpl](#).

Here is the caller graph for this function:



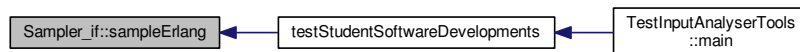
5.56.2.4 virtual double Sampler\_if::sampleDiscrete ( double *value*, double *acumProb*, ... ) [pure virtual]

Implemented in [SamplerDefaultImpl1](#), and [SamplerDummyImpl](#).

5.56.2.5 virtual double Sampler\_if::sampleErlang ( double *mean*, int *M* ) [pure virtual]

Implemented in [SamplerDefaultImpl1](#), and [SamplerDummyImpl](#).

Here is the caller graph for this function:



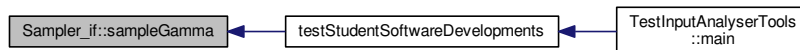
5.56.2.6 virtual double Sampler\_if::sampleExponential ( double *mean* ) [pure virtual]

Implemented in [SamplerDefaultImpl1](#), and [SamplerDummyImpl](#).

5.56.2.7 virtual double Sampler\_if::sampleGamma ( double *mean*, double *alpha* ) [pure virtual]

Implemented in [SamplerDefaultImpl1](#), and [SamplerDummyImpl](#).

Here is the caller graph for this function:



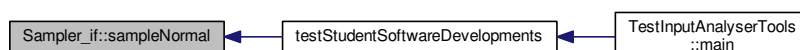
5.56.2.8 virtual double Sampler\_if::sampleLogNormal ( double *mean*, double *stddev* ) [pure virtual]

Implemented in [SamplerDefaultImpl1](#), and [SamplerDummyImpl](#).

5.56.2.9 virtual double Sampler\_if::sampleNormal ( double *mean*, double *stddev* ) [pure virtual]

Implemented in [SamplerDefaultImpl1](#), and [SamplerDummyImpl](#).

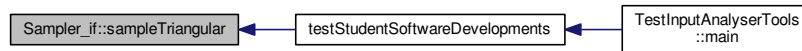
Here is the caller graph for this function:



5.56.2.10 `virtual double Sampler_if::sampleTriangular ( double min, double mode, double max )` [pure virtual]

Implemented in [SamplerDefaultImpl1](#), and [SamplerDummyImpl](#).

Here is the caller graph for this function:



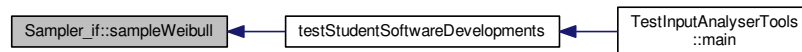
5.56.2.11 `virtual double Sampler_if::sampleUniform ( double min, double max )` [pure virtual]

Implemented in [SamplerDefaultImpl1](#), and [SamplerDummyImpl](#).

5.56.2.12 `virtual double Sampler_if::sampleWeibull ( double alpha, double scale )` [pure virtual]

Implemented in [SamplerDefaultImpl1](#), and [SamplerDummyImpl](#).

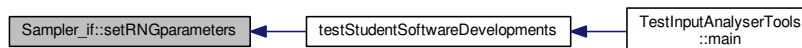
Here is the caller graph for this function:



5.56.2.13 `virtual void Sampler_if::setRNGparameters ( RNG_Parameters * param )` [pure virtual]

Implemented in [SamplerDefaultImpl1](#), and [SamplerDummyImpl](#).

Here is the caller graph for this function:



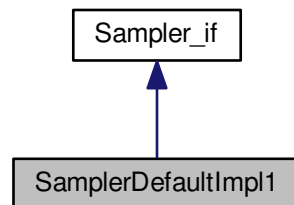
The documentation for this class was generated from the following file:

- [Sampler\\_if.h](#)

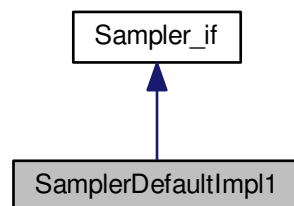
## 5.57 `SamplerDefaultImpl1` Class Reference

```
#include <SamplerDefaultImpl1.h>
```

Inheritance diagram for `SamplerDefaultImpl1`:



Collaboration diagram for `SamplerDefaultImpl1`:



### Classes

- class [DefaultImpl1RNG\\_Parameters](#)

### Public Member Functions

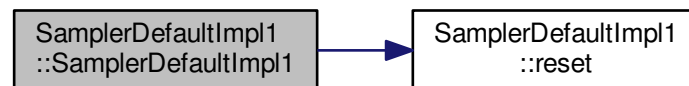
- [SamplerDefaultImpl1](#) ()
- [SamplerDefaultImpl1](#) (const [SamplerDefaultImpl1](#) &orig)
- virtual [~SamplerDefaultImpl1](#) ()
- double [random](#) ()
- double [sampleUniform](#) (double min, double max)
- double [sampleExponential](#) (double mean)
- double [sampleErlang](#) (double mean, int M)
- double [sampleNormal](#) (double mean, double stddev)
- double [sampleGamma](#) (double mean, double alpha)

- double [sampleBeta](#) (double alpha, double beta, double infLimit, double supLimit)
- double [sampleWeibull](#) (double alpha, double scale)
- double [sampleLogNormal](#) (double mean, double stddev)
- double [sampleTriangular](#) (double min, double mode, double max)
- double [sampleDiscrete](#) (double value, double acumProb,...)
- void [reset](#) ()  
*reinitialize seed and other parameters so (pseudo) random number sequence will be generated again.*
- void [setRNGparameters](#) ([RNG\\_Parameters](#) \*param)
- [RNG\\_Parameters](#) \* [getRNGparameters](#) () const

## 5.57.1 Constructor & Destructor Documentation

### 5.57.1.1 [SamplerDefaultImpl1::SamplerDefaultImpl1](#) ( )

Here is the call graph for this function:



### 5.57.1.2 [SamplerDefaultImpl1::SamplerDefaultImpl1](#) ( const [SamplerDefaultImpl1](#) & orig )

### 5.57.1.3 [SamplerDefaultImpl1::~~SamplerDefaultImpl1](#) ( ) [virtual]

## 5.57.2 Member Function Documentation

### 5.57.2.1 [Sampler\\_if::RNG\\_Parameters](#) \* [SamplerDefaultImpl1::getRNGparameters](#) ( ) const [virtual]

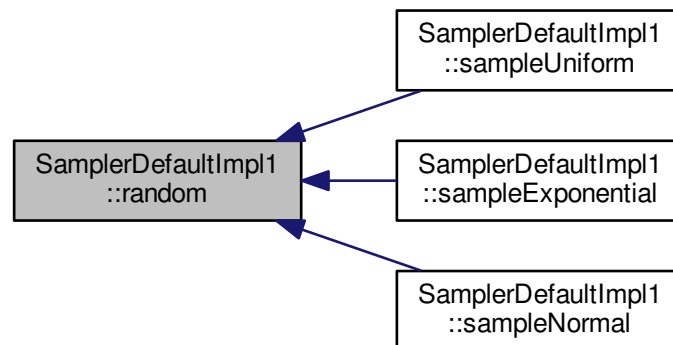
Implements [Sampler\\_if](#).

### 5.57.2.2 double [SamplerDefaultImpl1::random](#) ( ) [virtual]

Implements [Sampler\\_if](#).



Here is the caller graph for this function:



#### 5.57.2.3 void SamplerDefaultImpl1::reset ( )

reinitialize seed and other parameters so (pseudo) random number sequence will be generated again.

Here is the caller graph for this function:



#### 5.57.2.4 double SamplerDefaultImpl1::sampleBeta ( double *alpha*, double *beta*, double *infLimit*, double *supLimit* ) [virtual]

Implements [Sampler\\_if](#).

#### 5.57.2.5 double SamplerDefaultImpl1::sampleDiscrete ( double *value*, double *acumProb*, ... ) [virtual]

Implements [Sampler\\_if](#).

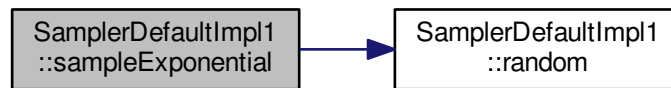
#### 5.57.2.6 double SamplerDefaultImpl1::sampleErlang ( double *mean*, int *M* ) [virtual]

Implements [Sampler\\_if](#).

5.57.2.7 `double SamplerDefaultImpl1::sampleExponential ( double mean ) [virtual]`

Implements [Sampler\\_if](#).

Here is the call graph for this function:



5.57.2.8 `double SamplerDefaultImpl1::sampleGamma ( double mean, double alpha ) [virtual]`

Implements [Sampler\\_if](#).

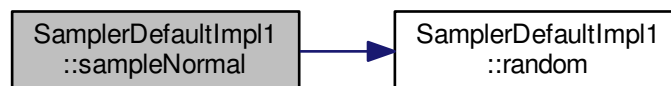
5.57.2.9 `double SamplerDefaultImpl1::sampleLogNormal ( double mean, double stddev ) [virtual]`

Implements [Sampler\\_if](#).

5.57.2.10 `double SamplerDefaultImpl1::sampleNormal ( double mean, double stddev ) [virtual]`

Implements [Sampler\\_if](#).

Here is the call graph for this function:



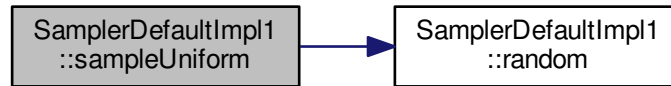
5.57.2.11 `double SamplerDefaultImpl1::sampleTriangular ( double min, double mode, double max ) [virtual]`

Implements [Sampler\\_if](#).

5.57.2.12 `double SamplerDefaultImpl1::sampleUniform ( double min, double max )` [virtual]

Implements [Sampler\\_if](#).

Here is the call graph for this function:



5.57.2.13 `double SamplerDefaultImpl1::sampleWeibull ( double alpha, double scale )` [virtual]

Implements [Sampler\\_if](#).

5.57.2.14 `void SamplerDefaultImpl1::setRNGparameters ( Sampler_if::RNG_Parameters * param )` [virtual]

Implements [Sampler\\_if](#).

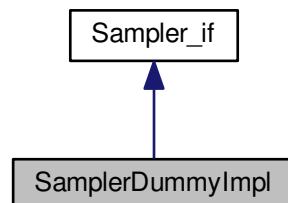
The documentation for this class was generated from the following files:

- [SamplerDefaultImpl1.h](#)
- [SamplerDefaultImpl1.cpp](#)

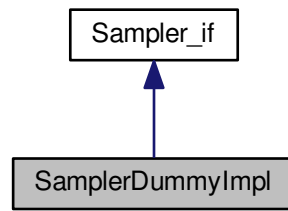
## 5.58 SamplerDummyImpl Class Reference

```
#include <SamplerDummyImpl.h>
```

Inheritance diagram for `SamplerDummyImpl`:



Collaboration diagram for `SamplerDummyImpl`:



## Classes

- class [MyRNG\\_Parameters](#)

## Public Member Functions

- [SamplerDummyImpl](#) ()
- [SamplerDummyImpl](#) (const [SamplerDummyImpl](#) &orig)
- [~SamplerDummyImpl](#) ()
- double [random](#) ()
- double [sampleUniform](#) (double min, double max)
- double [sampleExponential](#) (double mean)
- double [sampleErlang](#) (double mean, int M)
- double [sampleNormal](#) (double mean, double stddev)
- double [sampleGamma](#) (double mean, double alpha)
- double [sampleBeta](#) (double alpha, double beta, double infLimit, double supLimit)
- double [sampleWeibull](#) (double alpha, double scale)
- double [sampleLogNormal](#) (double mean, double stddev)
- double [sampleTriangular](#) (double min, double mode, double max)
- double [sampleDiscrete](#) (double value, double acumProb,...)
- void [setRNGparameters](#) ([RNG\\_Parameters](#) \*param)
- [RNG\\_Parameters](#) \* [getRNGparameters](#) () const

### 5.58.1 Constructor & Destructor Documentation

5.58.1.1 `SamplerDummyImpl::SamplerDummyImpl ( )`

5.58.1.2 `SamplerDummyImpl::SamplerDummyImpl ( const SamplerDummyImpl & orig )`

5.58.1.3 `SamplerDummyImpl::~~SamplerDummyImpl ( )`

### 5.58.2 Member Function Documentation

5.58.2.1 `Sampler\_if::RNG\_Parameters * SamplerDummyImpl::getRNGparameters ( ) const` `[virtual]`

Implements [Sampler\\_if](#).

5.58.2.2 `double SamplerDummyImpl::random ( )` [virtual]

Implements [Sampler\\_if](#).

5.58.2.3 `double SamplerDummyImpl::sampleBeta ( double alpha, double beta, double infLimit, double supLimit )` [virtual]

Implements [Sampler\\_if](#).

5.58.2.4 `double SamplerDummyImpl::sampleDiscrete ( double value, double acumProb, ... )` [virtual]

Implements [Sampler\\_if](#).

5.58.2.5 `double SamplerDummyImpl::sampleErlang ( double mean, int M )` [virtual]

Implements [Sampler\\_if](#).

5.58.2.6 `double SamplerDummyImpl::sampleExponential ( double mean )` [virtual]

Implements [Sampler\\_if](#).

5.58.2.7 `double SamplerDummyImpl::sampleGamma ( double mean, double alpha )` [virtual]

Implements [Sampler\\_if](#).

5.58.2.8 `double SamplerDummyImpl::sampleLogNormal ( double mean, double stddev )` [virtual]

Implements [Sampler\\_if](#).

5.58.2.9 `double SamplerDummyImpl::sampleNormal ( double mean, double stddev )` [virtual]

Implements [Sampler\\_if](#).

5.58.2.10 `double SamplerDummyImpl::sampleTriangular ( double min, double mode, double max )` [virtual]

Implements [Sampler\\_if](#).

5.58.2.11 `double SamplerDummyImpl::sampleUniform ( double min, double max )` [virtual]

Implements [Sampler\\_if](#).

5.58.2.12 `double SamplerDummyImpl::sampleWeibull ( double alpha, double scale )` [virtual]

Implements [Sampler\\_if](#).

5.58.2.13 `void SamplerDummyImpl::setRNGparameters ( Sampler_if::RNG_Parameters * param )` [virtual]

Implements [Sampler\\_if](#).

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

- [SamplerDummyImpl.h](#)
- [SamplerDummyImpl.cpp](#)

## 5.59 ScenarioExperiment\_if Class Reference

```
#include <ScenarioExperiment_if.h>
```

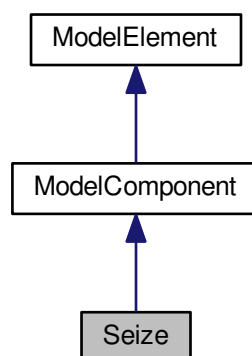
The documentation for this class was generated from the following file:

- [ScenarioExperiment\\_if.h](#)

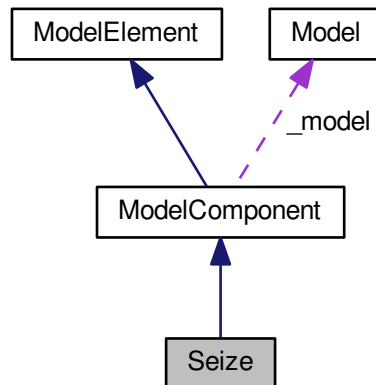
## 5.60 Seize Class Reference

```
#include <Seize.h>
```

Inheritance diagram for Seize:



Collaboration diagram for Seize:



## Public Member Functions

- [Seize](#) ([Model](#) \*model)
- [Seize](#) (const [Seize](#) &orig)
- virtual [~Seize](#) ()
- virtual std::string [show](#) ()
- void [setLastMemberSeized](#) (unsigned int \_lastMemberSeized)
- unsigned int [getLastMemberSeized](#) () const
- void [setSaveAttribute](#) (std::string \_saveAttribute)
- std::string [getSaveAttribute](#) () const
- void [setRule](#) ([Resource::ResourceRule](#) \_rule)
- [Resource::ResourceRule](#) [getRule](#) () const
- void [setQuantity](#) (std::string \_quantity)
- std::string [getQuantity](#) () const
- void [setResourceType](#) ([Resource::ResourceType](#) \_resourceType)
- [Resource::ResourceType](#) [getResourceType](#) () const
- void [setPriority](#) (unsigned short \_priority)
- unsigned short [getPriority](#) () const
- void [setAllocationType](#) (unsigned int \_allocationType)
- unsigned int [getAllocationType](#) () const
- void [setResourceName](#) (std::string \_resourceName) throw ()
- std::string [getResourceName](#) () const
- void [setQueueName](#) (std::string queueName) throw ()
- std::string [getQueueName](#) () const
- void [setResource](#) ([Resource](#) \*resource)
- [Resource](#) \* [getResource](#) () const
- void [setQueue](#) ([Queue](#) \*queue)
- [Queue](#) \* [getQueue](#) () const

## Protected Member Functions

- virtual void `_execute` (`Entity *entity`)
- virtual void `_loadInstance` (`std::list< std::string > words`)
- virtual `std::list< std::string > * _saveInstance` ()
- virtual bool `_verifySymbols` (`std::string *errorMessage`)

## Additional Inherited Members

### 5.60.1 Detailed Description

`Seize` tries to allocate a certain amount of a resource

### 5.60.2 Constructor & Destructor Documentation

5.60.2.1 `Seize::Seize ( Model * model )`

5.60.2.2 `Seize::Seize ( const Seize & orig )`

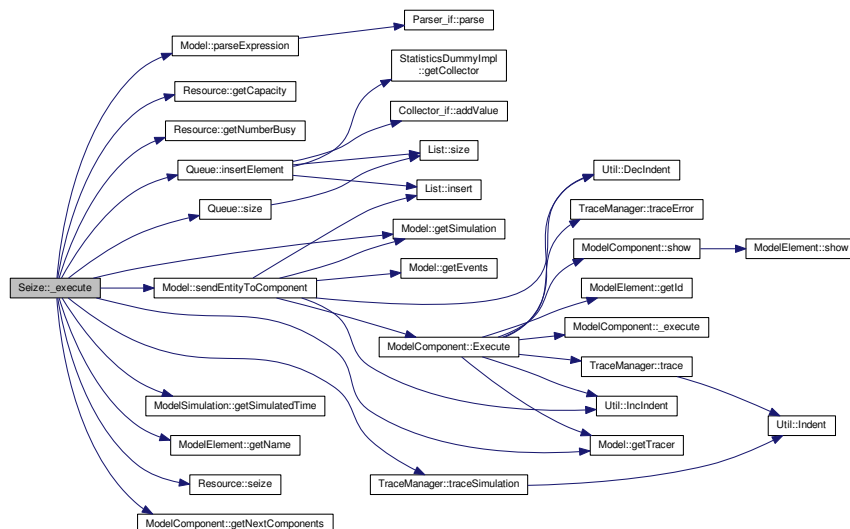
5.60.2.3 `Seize::~Seize ( )` `[virtual]`

### 5.60.3 Member Function Documentation

5.60.3.1 `void Seize::_execute ( Entity * entity )` `[protected]`, `[virtual]`

Implements `ModelComponent`.

Here is the call graph for this function:





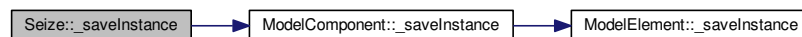
5.60.3.2 `void Seize::_loadInstance ( std::list< std::string > words )` [protected],[virtual]

Implements [ModelElement](#).

5.60.3.3 `std::list< std::string > * Seize::_saveInstance ( )` [protected],[virtual]

Reimplemented from [ModelComponent](#).

Here is the call graph for this function:



5.60.3.4 `bool Seize::_verifySymbols ( std::string * errorMessage )` [protected],[virtual]

Implements [ModelElement](#).

5.60.3.5 `unsigned int Seize::getAllocationType ( ) const`

5.60.3.6 `unsigned int Seize::getLastMemberSeized ( ) const`

5.60.3.7 `unsigned short Seize::getPriority ( ) const`

5.60.3.8 `std::string Seize::getQuantity ( ) const`

5.60.3.9 `Queue * Seize::getQueue ( ) const`

5.60.3.10 `std::string Seize::getQueueName ( ) const`

Here is the call graph for this function:



5.60.3.11 **Resource \* Seize::getResource ( ) const**

5.60.3.12 **std::string Seize::getResourceName ( ) const**

Here is the call graph for this function:



5.60.3.13 **Resource::ResourceType Seize::getResourceType ( ) const**

5.60.3.14 **Resource::ResourceRule Seize::getRule ( ) const**

5.60.3.15 **std::string Seize::getSaveAttribute ( ) const**

5.60.3.16 **void Seize::setAllocationType ( unsigned int *\_allocationType* )**

5.60.3.17 **void Seize::setLastMemberSeized ( unsigned int *\_lastMemberSeized* )**

5.60.3.18 **void Seize::setPriority ( unsigned short *\_priority* )**

5.60.3.19 **void Seize::setQuantity ( std::string *\_quantity* )**

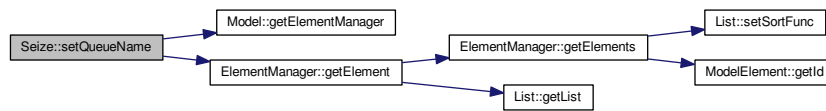
5.60.3.20 **void Seize::setQueue ( Queue \* *queue* )**

Here is the caller graph for this function:



#### 5.60.3.21 void Seize::setQueueName ( std::string *queueName* ) throw )

Here is the call graph for this function:



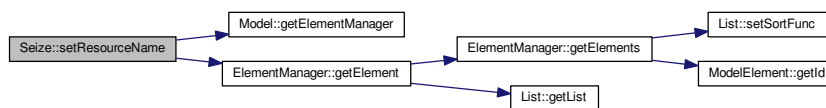
#### 5.60.3.22 void Seize::setResource ( Resource \* *resource* )

Here is the caller graph for this function:



#### 5.60.3.23 void Seize::setResourceName ( std::string *\_resourceName* ) throw )

Here is the call graph for this function:



#### 5.60.3.24 void Seize::setResourceType ( Resource::ResourceType *\_resourceType* )

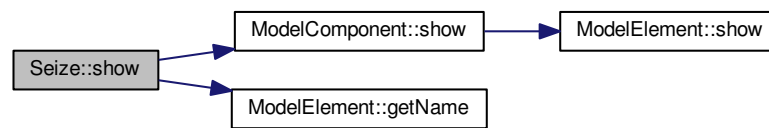
#### 5.60.3.25 void Seize::setRule ( Resource::ResourceRule *\_rule* )

#### 5.60.3.26 void Seize::setSaveAttribute ( std::string *\_saveAttribute* )

#### 5.60.3.27 std::string Seize::show ( ) [virtual]

Reimplemented from [ModelComponent](#).

Here is the call graph for this function:



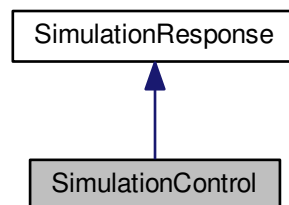
The documentation for this class was generated from the following files:

- [Seize.h](#)
- [Seize.cpp](#)

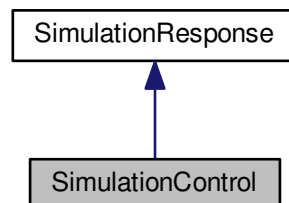
## 5.61 SimulationControl Class Reference

```
#include <SimulationControl.h>
```

Inheritance diagram for SimulationControl:



Collaboration diagram for SimulationControl:



## Public Member Functions

- [SimulationControl](#) (std::string type, std::string name, [GetterMember](#) getterMember, [SetterMember](#) setter↔ Member)
- [SimulationControl](#) (const [SimulationControl](#) &orig)
- virtual [~SimulationControl](#) ()
- void [setValue](#) (double value)

## Additional Inherited Members

### 5.61.1 Detailed Description

Represents any possible parameter or control for a simulation. Any element or event the model can declare one of its own attribute as a simulation control. It just have to create a [SimulationControl](#) object, passing the access to the methods that gets and sets the control value and including this [SimulationControl](#) in the corresponding list of the model

### 5.61.2 Constructor & Destructor Documentation

5.61.2.1 [SimulationControl::SimulationControl](#) ( std::string *type*, std::string *name*, [GetterMember](#) *getterMember*, [SetterMember](#) *setterMember* )

5.61.2.2 [SimulationControl::SimulationControl](#) ( const [SimulationControl](#) & *orig* )

5.61.2.3 [SimulationControl::~~SimulationControl](#) ( ) [virtual]

### 5.61.3 Member Function Documentation

5.61.3.1 void [SimulationControl::setValue](#) ( double *value* )

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

- [SimulationControl.h](#)
- [SimulationControl.cpp](#)

## 5.62 SimulationEvent Class Reference

```
#include <OnEventManager.h>
```

## Public Member Functions

- [SimulationEvent](#) (unsigned int replicationNumber, [Event](#) \*event)
- unsigned int [getReplicationNumber](#) () const
- [Event](#) \* [getEventProcessed](#) () const

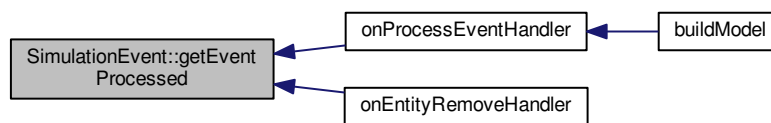
### 5.62.1 Constructor & Destructor Documentation

5.62.1.1 `SimulationEvent::SimulationEvent ( unsigned int replicationNumber, Event * event )` `[inline]`

### 5.62.2 Member Function Documentation

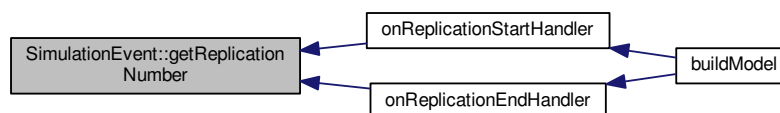
5.62.2.1 `Event* SimulationEvent::getEventProcessed ( ) const` `[inline]`

Here is the caller graph for this function:



5.62.2.2 `unsigned int SimulationEvent::getReplicationNumber ( ) const` `[inline]`

Here is the caller graph for this function:



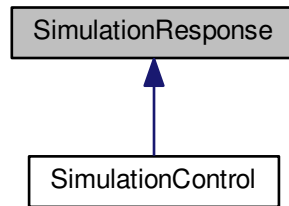
The documentation for this class was generated from the following file:

- [OnEventManager.h](#)

## 5.63 SimulationResponse Class Reference

```
#include <SimulationResponse.h>
```

Inheritance diagram for SimulationResponse:



### Public Member Functions

- [SimulationResponse](#) (std::string type, std::string name, [GetterMember](#) getterMember)
- [SimulationResponse](#) (const [SimulationResponse](#) &orig)
- virtual [~SimulationResponse](#) ()
- double [getValue](#) ()
- std::string [getName](#) () const
- std::string [getType](#) () const

### Protected Attributes

- std::string [\\_type](#)
- std::string [\\_name](#)
- [GetterMember](#) [\\_getterMemberFunction](#)

#### 5.63.1 Detailed Description

Represents any possible response of a simulation. Any element or event the model can declare one of its own attribute as a simulation response. It just have to create a [SimulationResponse](#) object, passing the access to the method that gets the response value and including this [SimulationResponse](#) in the corresponding list of the model

#### 5.63.2 Constructor & Destructor Documentation

5.63.2.1 [SimulationResponse::SimulationResponse](#) ( std::string *type*, std::string *name*, [GetterMember](#) *getterMember* )

5.63.2.2 [SimulationResponse::SimulationResponse](#) ( const [SimulationResponse](#) & *orig* )

5.63.2.3 [SimulationResponse::~~SimulationResponse](#) ( ) [virtual]

#### 5.63.3 Member Function Documentation

5.63.3.1 `std::string SimulationResponse::getName ( ) const`

5.63.3.2 `std::string SimulationResponse::getType ( ) const`

5.63.3.3 `double SimulationResponse::getValue ( )`

## 5.63.4 Member Data Documentation

5.63.4.1 **GetterMember** `SimulationResponse::_getterMemberFunction` `[protected]`

5.63.4.2 `std::string SimulationResponse::_name` `[protected]`

5.63.4.3 `std::string SimulationResponse::_type` `[protected]`

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

- [SimulationResponse.h](#)
- [SimulationResponse.cpp](#)

## 5.64 SimulationScenario Class Reference

```
#include <SimulationScenario.h>
```

### Public Member Functions

- [SimulationScenario](#) ( )
- [SimulationScenario](#) (const [SimulationScenario](#) &orig)
- virtual [~SimulationScenario](#) ( )
- void [setName](#) (std::string \_name)
- std::string [getName](#) ( ) const
- std::list< double > \* [getResponseValues](#) ( ) const
- std::list< double > \* [getControlValues](#) ( ) const
- void [setModelFilename](#) (std::string \_modelFilename)
- std::string [getModelFilename](#) ( ) const
- double [getResponseValue](#) ([SimulationResponse](#) \*value)
- double [getControlValue](#) ([SimulationControl](#) \*control)
- void [setControlValue](#) ([SimulationControl](#) \*control, double value)

### 5.64.1 Detailed Description

Represents a scenario where a specific model (defined my ModelFilename) will be simulated. To each scenario will be associated a set of [SimulationControl](#) and [SimulationResponse](#), and their values are set to the scenario by the ProcessAnalyser.



## 5.64.2 Constructor & Destructor Documentation

5.64.2.1 `SimulationScenario::SimulationScenario ( )`

5.64.2.2 `SimulationScenario::SimulationScenario ( const SimulationScenario & orig )`

5.64.2.3 `SimulationScenario::~SimulationScenario ( ) [virtual]`

## 5.64.3 Member Function Documentation

5.64.3.1 `double SimulationScenario::getControlValue ( SimulationControl * control )`

5.64.3.2 `std::list< double > * SimulationScenario::getControlValues ( ) const`

5.64.3.3 `std::string SimulationScenario::getModelFilename ( ) const`

5.64.3.4 `std::string SimulationScenario::getName ( ) const`

5.64.3.5 `double SimulationScenario::getResponseValue ( SimulationResponse * value )`

5.64.3.6 `std::list< double > * SimulationScenario::getResponseValues ( ) const`

5.64.3.7 `void SimulationScenario::setControlValue ( SimulationControl * control, double value )`

5.64.3.8 `void SimulationScenario::setModelFilename ( std::string _modelName )`

5.64.3.9 `void SimulationScenario::setName ( std::string _name )`

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

- [SimulationScenario.h](#)
- [SimulationScenario.cpp](#)

## 5.65 Simulator Class Reference

```
#include <Simulator.h>
```

## Public Member Functions

- [Simulator](#) ()
- [Simulator](#) (const [Simulator](#) &orig)
- virtual [~Simulator](#) ()
- [List< Model \\* > \\* getModels](#) () const  
*Returns the list of open models in the simulator.*
- [List< Plugin \\* > \\* getPlugins](#) () const
- std::string [getVersion](#) () const
- std::string [getLicense](#) () const
- std::string [getName](#) () const
- [Sampler\\_if \\* getSampler](#) () const  
*Returns the Sampler, used to generate samples accordingly to a probability distribution.*
- [Fitter\\_if \\* getFitter](#) () const  
*Returns the fitter, responsible for carrying out tests of adherence of theoretical distributions of probability with sampled data.*

### 5.65.1 Detailed Description

The main class of the ReGenesys KERNEL simulation. It gives access to simulation models and tools.

### 5.65.2 Constructor & Destructor Documentation

#### 5.65.2.1 Simulator::Simulator ( )

#### 5.65.2.2 Simulator::Simulator ( const Simulator & orig )

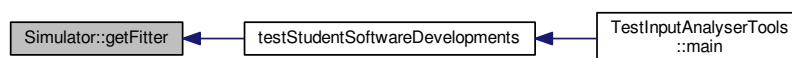
#### 5.65.2.3 Simulator::~~Simulator ( ) [virtual]

### 5.65.3 Member Function Documentation

#### 5.65.3.1 Fitter\_if \* Simulator::getFitter ( ) const

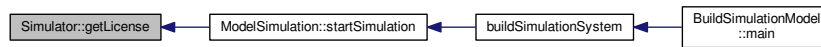
Returns the fitter, responsible for carrying out tests of adherence of theoretical distributions of probability with sampled data.

Here is the caller graph for this function:



### 5.65.3.2 `std::string Simulator::getLicense ( ) const`

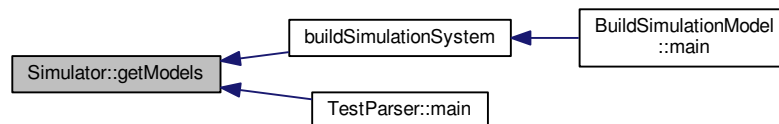
Here is the caller graph for this function:



### 5.65.3.3 `List< Model * > * Simulator::getModels ( ) const`

Returns the list of open models in the simulator.

Here is the caller graph for this function:



### 5.65.3.4 `std::string Simulator::getName ( ) const`

Here is the caller graph for this function:

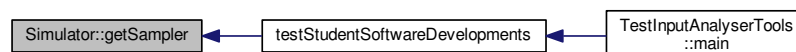


### 5.65.3.5 `List< Plugin * > * Simulator::getPlugins ( ) const`

### 5.65.3.6 `Sampler_if * Simulator::getSampler ( ) const`

Returns the Sampler, used to generate samples accordingly to a probability distribution.

Here is the caller graph for this function:



#### 5.65.3.7 `std::string Simulator::getVersion ( ) const`

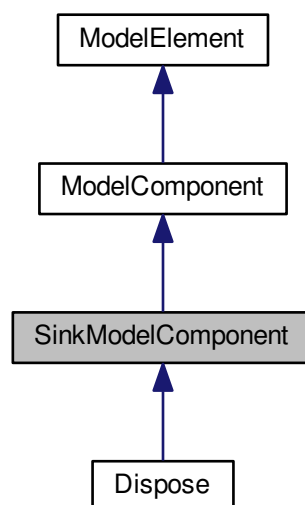
The documentation for this class was generated from the following files:

- [Simulator.h](#)
- [Simulator.cpp](#)

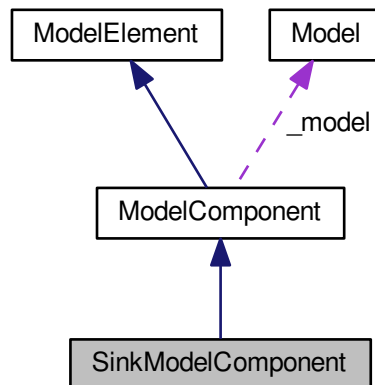
## 5.66 SinkModelComponent Class Reference

```
#include <SinkModelComponent.h>
```

Inheritance diagram for SinkModelComponent:



Collaboration diagram for SinkModelComponent:



### Public Member Functions

- [SinkModelComponent](#) ([Model](#) \*model)
- [SinkModelComponent](#) (const [SinkModelComponent](#) &orig)
- virtual [~SinkModelComponent](#) ()
- void [setCollectStatistics](#) (bool \_collectStatistics)
- bool [isCollectStatistics](#) () const

### Additional Inherited Members

#### 5.66.1 Detailed Description

This class is the basis for any component representing the end of a process flow, such as a [Dispose](#). It can remove entities from the system and collect statistics.

#### 5.66.2 Constructor & Destructor Documentation

5.66.2.1 [SinkModelComponent::SinkModelComponent](#) ( [Model](#) \* *model* )

5.66.2.2 [SinkModelComponent::SinkModelComponent](#) ( const [SinkModelComponent](#) & *orig* )

5.66.2.3 [SinkModelComponent::~~SinkModelComponent](#) ( ) [[virtual](#)]

#### 5.66.3 Member Function Documentation

5.66.3.1 bool [SinkModelComponent::isCollectStatistics](#) ( ) const

5.66.3.2 void [SinkModelComponent::setCollectStatistics](#) ( bool *\_collectStatistics* )

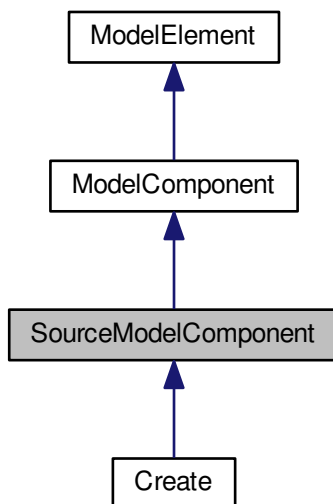
The documentation for this class was generated from the following files:

- [SinkModelComponent.h](#)
- [SinkModelComponent.cpp](#)

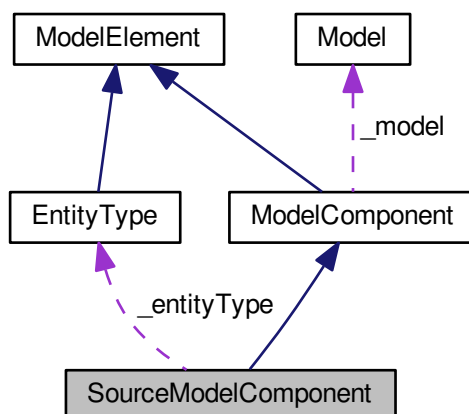
## 5.67 SourceModelComponent Class Reference

```
#include <SourceModelComponent.h>
```

Inheritance diagram for SourceModelComponent:



Collaboration diagram for SourceModelComponent:



## Public Member Functions

- [SourceModelComponent](#) ([Model](#) \*model)
- [SourceModelComponent](#) (const [SourceModelComponent](#) &orig)
- virtual [~SourceModelComponent](#) ()
- void [setFirstCreation](#) (double [\\_firstCreation](#))
- double [getFirstCreation](#) () const
- void [setCollectStatistics](#) (bool [\\_collectStatistics](#))
- bool [isCollectStatistics](#) () const
- void [setEntityType](#) ([EntityType](#) \*[\\_entityType](#))
- [EntityType](#) \* [getEntityType](#) () const
- void [setTimeUnit](#) ([Util::TimeUnit](#) [\\_timeUnit](#))
- [Util::TimeUnit](#) [getTimeUnit](#) () const
- void [setTimeBetweenCreationsExpression](#) (std::string [\\_timeBetweenCreations](#))
- std::string [getTimeBetweenCreationsExpression](#) () const
- void [setMaxCreations](#) (unsigned int [\\_maxCreations](#))
- unsigned int [getMaxCreations](#) () const
- unsigned int [getEntitiesCreated](#) () const
- void [setEntitiesCreated](#) (unsigned int [\\_entitiesCreated](#))
- void [setEntitiesPerCreation](#) (unsigned int [\\_entitiesPerCreation](#))
- unsigned int [getEntitiesPerCreation](#) () const
- virtual std::string [show](#) ()

## Protected Attributes

- [EntityType](#) \* [\\_entityType](#)
- double [\\_firstCreation](#) = 0.0
- unsigned int [\\_entitiesPerCreation](#) = 1
- unsigned int [\\_maxCreations](#) = std::numeric\_limits<unsigned int>::max()
- std::string [\\_timeBetweenCreationsExpression](#) = "10"
- [Util::TimeUnit](#) [\\_timeBetweenCreationsTimeUnit](#) = [Util::TimeUnit::second](#)
- bool [\\_collectStatistics](#) = true
- unsigned int [\\_entitiesCreatedSoFar](#) = 0

## Additional Inherited Members

### 5.67.1 Detailed Description

A source component implements the base for inserting entities into the model when its simulation is initialized. During the initialization, the new and empty future events list is populated by events of creating entities and sending them to the source components existing in the model

## 5.67.2 Constructor & Destructor Documentation

5.67.2.1 `SourceModelComponent::SourceModelComponent ( Model * model )`

5.67.2.2 `SourceModelComponent::SourceModelComponent ( const SourceModelComponent & orig )`

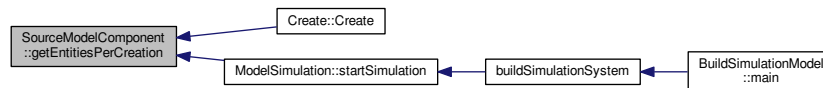
5.67.2.3 `SourceModelComponent::~SourceModelComponent ( ) [virtual]`

## 5.67.3 Member Function Documentation

5.67.3.1 `unsigned int SourceModelComponent::getEntitiesCreated ( ) const`

5.67.3.2 `unsigned int SourceModelComponent::getEntitiesPerCreation ( ) const`

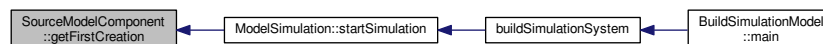
Here is the caller graph for this function:



5.67.3.3 `EntityType * SourceModelComponent::getEntityType ( ) const`

5.67.3.4 `double SourceModelComponent::getFirstCreation ( ) const`

Here is the caller graph for this function:



5.67.3.5 `unsigned int SourceModelComponent::getMaxCreations ( ) const`

5.67.3.6 `std::string SourceModelComponent::getTimeBetweenCreationsExpression ( ) const`

5.67.3.7 `Util::TimeUnit SourceModelComponent::getTimeUnit ( ) const`

5.67.3.8 `bool SourceModelComponent::isCollectStatistics ( ) const`

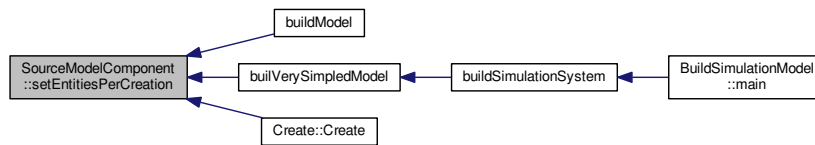
5.67.3.9 `void SourceModelComponent::setCollectStatistics ( bool _collectStatistics )`



5.67.3.10 void SourceModelComponent::setEntitiesCreated ( unsigned int *\_entitiesCreated* )

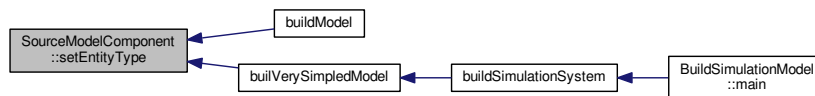
5.67.3.11 void SourceModelComponent::setEntitiesPerCreation ( unsigned int *\_entitiesPerCreation* )

Here is the caller graph for this function:



5.67.3.12 void SourceModelComponent::setEntityType ( EntityType \* *\_entityType* )

Here is the caller graph for this function:

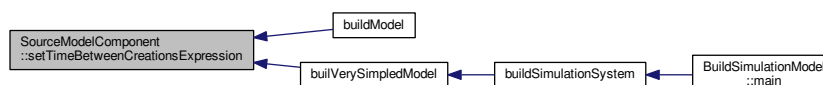


5.67.3.13 void SourceModelComponent::setFirstCreation ( double *\_firstCreation* )

5.67.3.14 void SourceModelComponent::setMaxCreations ( unsigned int *\_maxCreations* )

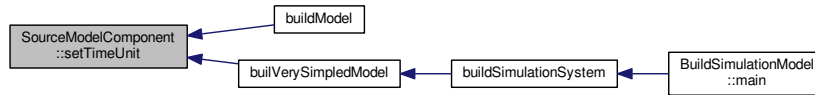
5.67.3.15 void SourceModelComponent::setTimeBetweenCreationsExpression ( std::string *\_timeBetweenCreations* )

Here is the caller graph for this function:



#### 5.67.3.16 void SourceModelComponent::setTimeUnit ( Util::TimeUnit \_timeUnit )

Here is the caller graph for this function:

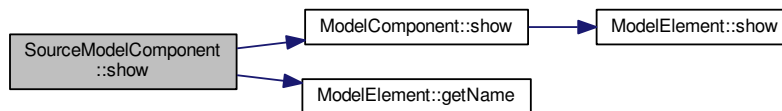


#### 5.67.3.17 std::string SourceModelComponent::show ( ) [virtual]

Reimplemented from [ModelComponent](#).

Reimplemented in [Create](#).

Here is the call graph for this function:



Here is the caller graph for this function:



### 5.67.4 Member Data Documentation

#### 5.67.4.1 bool SourceModelComponent::\_collectStatistics = true [protected]

#### 5.67.4.2 unsigned int SourceModelComponent::\_entitiesCreatedSoFar = 0 [protected]

#### 5.67.4.3 unsigned int SourceModelComponent::\_entitiesPerCreation = 1 [protected]

5.67.4.4 **EntityType\*** SourceModelComponent::\_entityType [protected]

5.67.4.5 **double** SourceModelComponent::\_firstCreation = 0.0 [protected]

5.67.4.6 **unsigned int** SourceModelComponent::\_maxCreations = std::numeric\_limits<unsigned int>::max()  
[protected]

5.67.4.7 **std::string** SourceModelComponent::\_timeBetweenCreationsExpression = "10" [protected]

5.67.4.8 **Util::TimeUnit** SourceModelComponent::\_timeBetweenCreationsTimeUnit = Util::TimeUnit::second  
[protected]

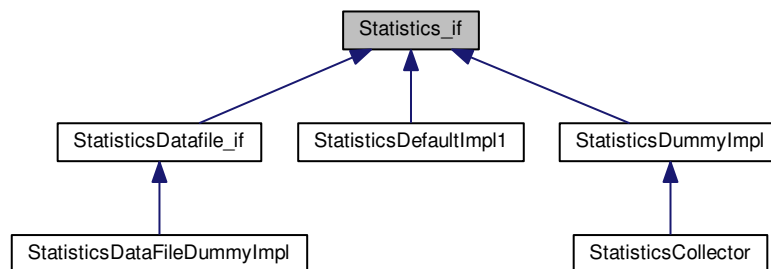
The documentation for this class was generated from the following files:

- [SourceModelComponent.h](#)
- [SourceModelComponent.cpp](#)

## 5.68 Statistics\_if Class Reference

```
#include <Statistics_if.h>
```

Inheritance diagram for Statistics\_if:



### Public Member Functions

- virtual [Collector\\_if](#) \* [getCollector](#) ()=0
- virtual void [setCollector](#) ([Collector\\_if](#) \*collector)=0
- virtual unsigned int [numElements](#) ()=0
- virtual double [min](#) ()=0
- virtual double [max](#) ()=0
- virtual double [average](#) ()=0
- virtual double [variance](#) ()=0
- virtual double [stddeviation](#) ()=0
- virtual double [variationCoef](#) ()=0
- virtual double [halfWidthConfidenceInterval](#) (double confidencelevel)=0
- virtual unsigned int [newSampleSize](#) (double confidencelevel, double halfWidth)=0

### 5.68.1 Detailed Description

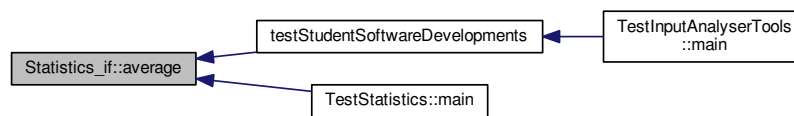
Interface for statist synthesis of a stochastic variable collected by a [Collector\\_if](#). The statistics generated may be updated based only on the previous statistics and the single newest added value or they may be updated based on a datafile, depending on the Collector implementation.

### 5.68.2 Member Function Documentation

#### 5.68.2.1 `virtual double Statistics_if::average ( ) [pure virtual]`

Implemented in [StatisticsDefaultImpl1](#), [StatisticsDummyImpl](#), and [StatisticsDataFileDummyImpl](#).

Here is the caller graph for this function:



#### 5.68.2.2 `virtual Collector_if* Statistics_if::getCollector ( ) [pure virtual]`

Implemented in [StatisticsDefaultImpl1](#), [StatisticsDummyImpl](#), and [StatisticsDataFileDummyImpl](#).

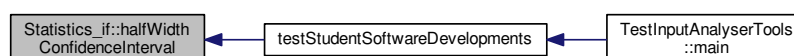
Here is the caller graph for this function:



#### 5.68.2.3 `virtual double Statistics_if::halfWidthConfidenceInterval ( double confidencelevel ) [pure virtual]`

Implemented in [StatisticsDefaultImpl1](#), [StatisticsDummyImpl](#), and [StatisticsDataFileDummyImpl](#).

Here is the caller graph for this function:



#### 5.68.2.4 virtual double Statistics\_if::max ( ) [pure virtual]

Implemented in [StatisticsDefaultImpl1](#), [StatisticsDummyImpl](#), and [StatisticsDataFileDummyImpl](#).

Here is the caller graph for this function:



#### 5.68.2.5 virtual double Statistics\_if::min ( ) [pure virtual]

Implemented in [StatisticsDefaultImpl1](#), [StatisticsDummyImpl](#), and [StatisticsDataFileDummyImpl](#).

Here is the caller graph for this function:



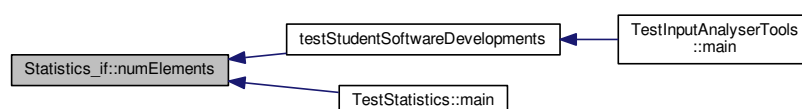
#### 5.68.2.6 virtual unsigned int Statistics\_if::newSampleSize ( double *confidencelevel*, double *halfWidth* ) [pure virtual]

Implemented in [StatisticsDefaultImpl1](#), [StatisticsDummyImpl](#), and [StatisticsDataFileDummyImpl](#).

#### 5.68.2.7 virtual unsigned int Statistics\_if::numElements ( ) [pure virtual]

Implemented in [StatisticsDefaultImpl1](#), [StatisticsDummyImpl](#), and [StatisticsDataFileDummyImpl](#).

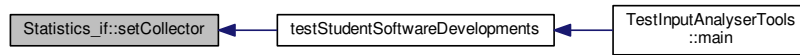
Here is the caller graph for this function:



5.68.2.8 `virtual void Statistics_if::setCollector ( Collector_if * collector ) [pure virtual]`

Implemented in [StatisticsDefaultImpl1](#), [StatisticsDummyImpl](#), and [StatisticsDataFileDummyImpl](#).

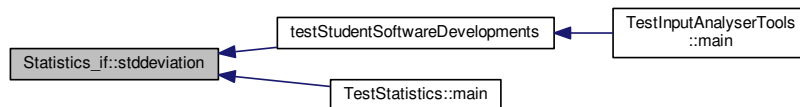
Here is the caller graph for this function:



5.68.2.9 `virtual double Statistics_if::stddeviation ( ) [pure virtual]`

Implemented in [StatisticsDefaultImpl1](#), [StatisticsDummyImpl](#), and [StatisticsDataFileDummyImpl](#).

Here is the caller graph for this function:



5.68.2.10 `virtual double Statistics_if::variance ( ) [pure virtual]`

Implemented in [StatisticsDefaultImpl1](#), [StatisticsDummyImpl](#), and [StatisticsDataFileDummyImpl](#).

5.68.2.11 `virtual double Statistics_if::variationCoef ( ) [pure virtual]`

Implemented in [StatisticsDefaultImpl1](#), [StatisticsDummyImpl](#), and [StatisticsDataFileDummyImpl](#).

Here is the caller graph for this function:



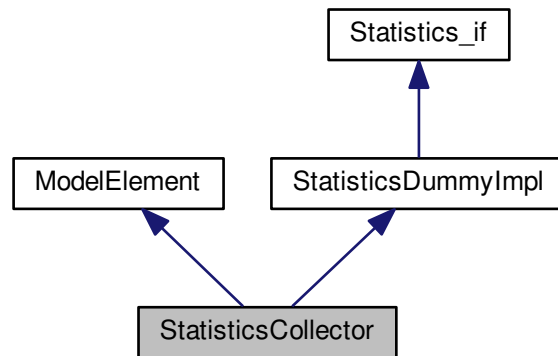
The documentation for this class was generated from the following file:

- [Statistics\\_if.h](#)

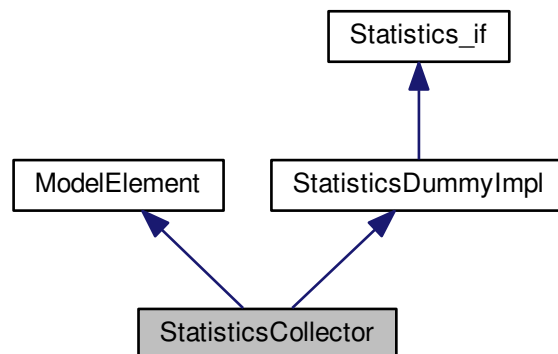
## 5.69 StatisticsCollector Class Reference

```
#include <StatisticsCollector.h>
```

Inheritance diagram for StatisticsCollector:



Collaboration diagram for StatisticsCollector:



### Public Member Functions

- [StatisticsCollector](#) ()
- [StatisticsCollector](#) (std::string name)
- [StatisticsCollector](#) (std::string name, [ModelElement](#) \*parent)
- [StatisticsCollector](#) (const [StatisticsCollector](#) &orig)
- virtual [~StatisticsCollector](#) ()
- virtual std::string [show](#) ()
- [ModelElement](#) \* [getParent](#) () const

## Protected Member Functions

- virtual void [\\_loadInstance](#) (std::list< std::string > words)
- virtual std::list< std::string > \* [\\_saveInstance](#) ()
- virtual bool [\\_verifySymbols](#) (std::string \*errorMessage)

## Additional Inherited Members

### 5.69.1 Constructor & Destructor Documentation

5.69.1.1 `StatisticsCollector::StatisticsCollector ( )`

5.69.1.2 `StatisticsCollector::StatisticsCollector ( std::string name )`

5.69.1.3 `StatisticsCollector::StatisticsCollector ( std::string name, ModelElement * parent )`

5.69.1.4 `StatisticsCollector::StatisticsCollector ( const StatisticsCollector & orig )`

5.69.1.5 `StatisticsCollector::~~StatisticsCollector ( )` [virtual]

### 5.69.2 Member Function Documentation

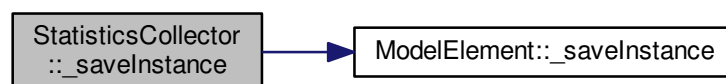
5.69.2.1 `void StatisticsCollector::_loadInstance ( std::list< std::string > words )` [protected],[virtual]

Implements [ModelElement](#).

5.69.2.2 `std::list< std::string > * StatisticsCollector::_saveInstance ( )` [protected],[virtual]

Reimplemented from [ModelElement](#).

Here is the call graph for this function:



5.69.2.3 `bool StatisticsCollector::_verifySymbols ( std::string * errorMessage )` [protected],[virtual]

Implements [ModelElement](#).



#### 5.69.2.4 `ModelElement * StatisticsCollector::getParent ( ) const`

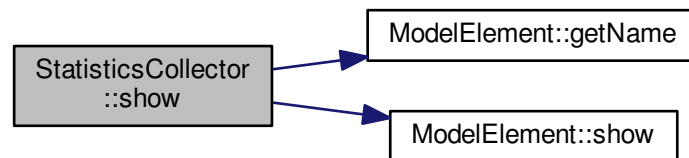
Here is the caller graph for this function:



#### 5.69.2.5 `std::string StatisticsCollector::show ( ) [virtual]`

Reimplemented from [ModelElement](#).

Here is the call graph for this function:



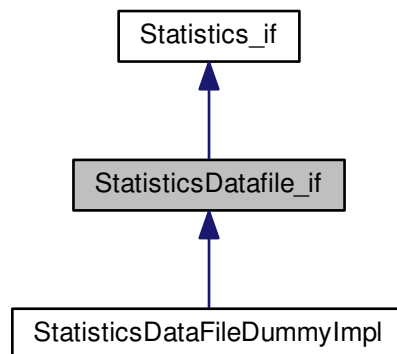
The documentation for this class was generated from the following files:

- [StatisticsCollector.h](#)
- [StatisticsCollector.cpp](#)

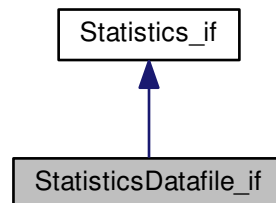
## 5.70 StatisticsDatafile\_if Class Reference

```
#include <StatisticsDataFile_if.h>
```

Inheritance diagram for StatisticsDatafile\_if:



Collaboration diagram for StatisticsDatafile\_if:



## Public Member Functions

- virtual double [mode](#) ()=0
- virtual double [mediane](#) ()=0
- virtual double [quartil](#) (unsigned short num)=0
- virtual double [decil](#) (unsigned short num)=0
- virtual double [centil](#) (unsigned short num)=0
- virtual void [setHistogramNumClasses](#) (unsigned short num)=0
- virtual unsigned short [histogramNumClasses](#) ()=0
- virtual double [histogramClassLowerLimit](#) (unsigned short classNum)=0
- virtual unsigned int [histogramClassFrequency](#) (unsigned short classNum)=0

## 5.70.1 Member Function Documentation

5.70.1.1 virtual double StatisticsDatafile\_if::centil ( unsigned short *num* ) [pure virtual]

Implemented in [StatisticsDataFileDummyImpl](#).

5.70.1.2 virtual double StatisticsDatafile\_if::decil ( unsigned short *num* ) [pure virtual]

Implemented in [StatisticsDataFileDummyImpl](#).

5.70.1.3 virtual unsigned int StatisticsDatafile\_if::histogramClassFrequency ( unsigned short *classNum* ) [pure virtual]

Implemented in [StatisticsDataFileDummyImpl](#).

5.70.1.4 virtual double StatisticsDatafile\_if::histogramClassLowerLimit ( unsigned short *classNum* ) [pure virtual]

Implemented in [StatisticsDataFileDummyImpl](#).

5.70.1.5 virtual unsigned short StatisticsDatafile\_if::histogramNumClasses ( ) [pure virtual]

Implemented in [StatisticsDataFileDummyImpl](#).

5.70.1.6 virtual double StatisticsDatafile\_if::mediane ( ) [pure virtual]

Implemented in [StatisticsDataFileDummyImpl](#).

5.70.1.7 virtual double StatisticsDatafile\_if::mode ( ) [pure virtual]

Implemented in [StatisticsDataFileDummyImpl](#).

5.70.1.8 virtual double StatisticsDatafile\_if::quartil ( unsigned short *num* ) [pure virtual]

Implemented in [StatisticsDataFileDummyImpl](#).

5.70.1.9 virtual void StatisticsDatafile\_if::setHistogramNumClasses ( unsigned short *num* ) [pure virtual]

Implemented in [StatisticsDataFileDummyImpl](#).

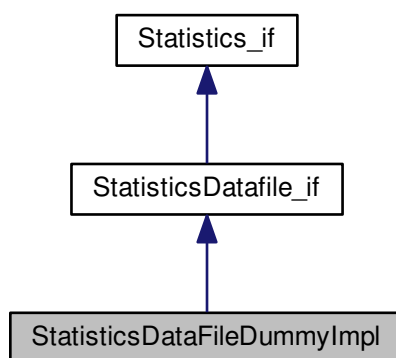
The documentation for this class was generated from the following file:

- [StatisticsDataFile\\_if.h](#)

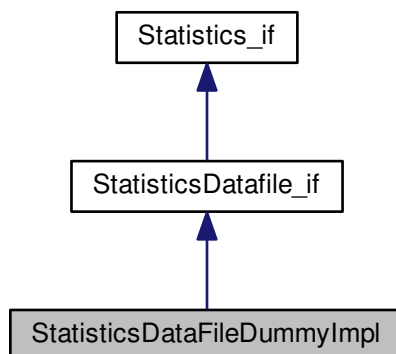
## 5.71 StatisticsDataFileDummyImpl Class Reference

```
#include <StatisticsDataFileDummyImpl.h>
```

Inheritance diagram for StatisticsDataFileDummyImpl:



Collaboration diagram for StatisticsDataFileDummyImpl:



### Public Member Functions

- [StatisticsDataFileDummyImpl](#) ()
- [StatisticsDataFileDummyImpl](#) (const [StatisticsDataFileDummyImpl](#) &orig)
- virtual [~StatisticsDataFileDummyImpl](#) ()
- virtual [Collector\\_if](#) \* [getCollector](#) ()
- void [setCollector](#) ([Collector\\_if](#) \*collector)

- virtual unsigned int [numElements](#) ()
- virtual double [min](#) ()
- virtual double [max](#) ()
- virtual double [average](#) ()
- virtual double [variance](#) ()
- virtual double [stddeviation](#) ()
- virtual double [variationCoef](#) ()
- virtual double [halfWidthConfidenceInterval](#) (double confidencelevel)
- virtual unsigned int [newSampleSize](#) (double confidencelevel, double halfWidth)
- virtual double [mode](#) ()
- virtual double [mediane](#) ()
- virtual double [quartil](#) (unsigned short num)
- virtual double [decil](#) (unsigned short num)
- virtual double [centil](#) (unsigned short num)
- virtual void [setHistogramNumClasses](#) (unsigned short num)
- virtual unsigned short [histogramNumClasses](#) ()
- virtual double [histogramClassLowerLimit](#) (unsigned short classNum)
- virtual unsigned int [histogramClassFrequency](#) (unsigned short classNum)

### 5.71.1 Constructor & Destructor Documentation

5.71.1.1 `StatisticsDataFileDummyImpl::StatisticsDataFileDummyImpl ( )`

5.71.1.2 `StatisticsDataFileDummyImpl::StatisticsDataFileDummyImpl ( const StatisticsDataFileDummyImpl & orig )`

5.71.1.3 `StatisticsDataFileDummyImpl::~StatisticsDataFileDummyImpl ( )` [virtual]

### 5.71.2 Member Function Documentation

5.71.2.1 `double StatisticsDataFileDummyImpl::average ( )` [virtual]

Implements [Statistics\\_if](#).

5.71.2.2 `double StatisticsDataFileDummyImpl::centil ( unsigned short num )` [virtual]

Implements [StatisticsDatafile\\_if](#).

5.71.2.3 `double StatisticsDataFileDummyImpl::decil ( unsigned short num )` [virtual]

Implements [StatisticsDatafile\\_if](#).

5.71.2.4 `Collector_if * StatisticsDataFileDummyImpl::getCollector ( )` [virtual]

Implements [Statistics\\_if](#).

5.71.2.5 `double StatisticsDataFileDummyImpl::halfWidthConfidenceInterval ( double confidencelevel )` [virtual]

Implements [Statistics\\_if](#).

5.71.2.6 `unsigned int StatisticsDataFileDummyImpl::histogramClassFrequency ( unsigned short classNum )` [virtual]

Implements [StatisticsDatafile\\_if](#).

5.71.2.7 `double StatisticsDataFileDummyImpl::histogramClassLowerLimit ( unsigned short classNum )` [virtual]

Implements [StatisticsDatafile\\_if](#).

5.71.2.8 `unsigned short StatisticsDataFileDummyImpl::histogramNumClasses ( )` [virtual]

Implements [StatisticsDatafile\\_if](#).

5.71.2.9 `double StatisticsDataFileDummyImpl::max ( )` [virtual]

Implements [Statistics\\_if](#).

5.71.2.10 `double StatisticsDataFileDummyImpl::median ( )` [virtual]

Implements [StatisticsDatafile\\_if](#).

5.71.2.11 `double StatisticsDataFileDummyImpl::min ( )` [virtual]

Implements [Statistics\\_if](#).

5.71.2.12 `double StatisticsDataFileDummyImpl::mode ( )` [virtual]

Implements [StatisticsDatafile\\_if](#).

5.71.2.13 `unsigned int StatisticsDataFileDummyImpl::newSampleSize ( double confidencelevel, double halfWidth )`  
[virtual]

Implements [Statistics\\_if](#).

5.71.2.14 `unsigned int StatisticsDataFileDummyImpl::numElements ( )` [virtual]

Implements [Statistics\\_if](#).

5.71.2.15 `double StatisticsDataFileDummyImpl::quartil ( unsigned short num )` [virtual]

Implements [StatisticsDatafile\\_if](#).

5.71.2.16 `void StatisticsDataFileDummyImpl::setCollector ( Collector_if * collector )` [virtual]

Implements [Statistics\\_if](#).

5.71.2.17 `void StatisticsDataFileDummyImpl::setHistogramNumClasses ( unsigned short num )` [virtual]

Implements [StatisticsDatafile\\_if](#).

5.71.2.18 `double StatisticsDataFileDummyImpl::stddeviation ( )` [virtual]

Implements [Statistics\\_if](#).

5.71.2.19 `double StatisticsDataFileDummyImpl::variance ( )` [virtual]

Implements [Statistics\\_if](#).

5.71.2.20 `double StatisticsDataFileDummyImpl::variationCoef ( )` [virtual]

Implements [Statistics\\_if](#).

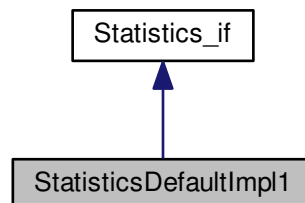
The documentation for this class was generated from the following files:

- [StatisticsDataFileDummyImpl.h](#)
- [StatisticsDataFileDummyImpl.cpp](#)

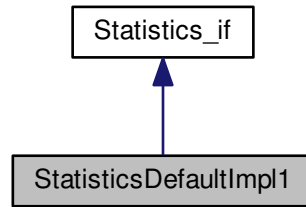
## 5.72 StatisticsDefaultImpl1 Class Reference

```
#include <StatisticsDefaultImpl1.h>
```

Inheritance diagram for StatisticsDefaultImpl1:



Collaboration diagram for StatisticsDefaultImpl1:



## Public Member Functions

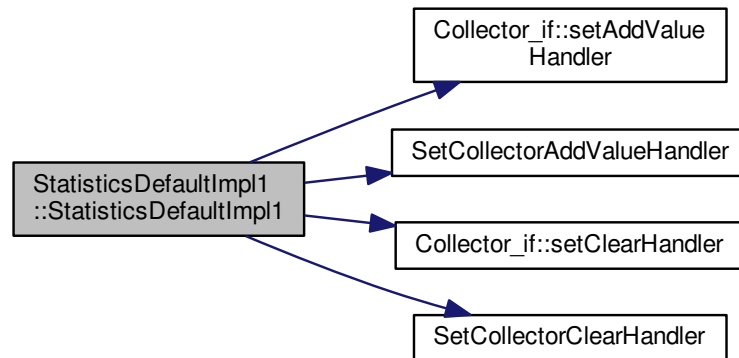
- [StatisticsDefaultImpl1](#) ()
- [StatisticsDefaultImpl1](#) (const [StatisticsDefaultImpl1](#) &orig)
- virtual [~StatisticsDefaultImpl1](#) ()
- virtual [Collector\\_if](#) \* [getCollector](#) ()
- void [setCollector](#) ([Collector\\_if](#) \*collector)
- virtual unsigned int [numElements](#) ()
- virtual double [min](#) ()
- virtual double [max](#) ()
- virtual double [average](#) ()
- virtual double [variance](#) ()
- virtual double [stddeviation](#) ()
- virtual double [variationCoef](#) ()
- virtual double [halfWidthConfidenceInterval](#) (double confidencelevel)
- virtual unsigned int [newSampleSize](#) (double confidencelevel, double halfWidth)

### 5.72.1 Constructor & Destructor Documentation



## 5.72.1.1 StatisticsDefaultImpl1::StatisticsDefaultImpl1 ( )

Here is the call graph for this function:



## 5.72.1.2 StatisticsDefaultImpl1::StatisticsDefaultImpl1 ( const StatisticsDefaultImpl1 &amp; orig )

## 5.72.1.3 StatisticsDefaultImpl1::~~StatisticsDefaultImpl1 ( ) [virtual]

Here is the call graph for this function:



## 5.72.2 Member Function Documentation

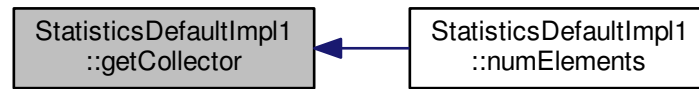
## 5.72.2.1 double StatisticsDefaultImpl1::average ( ) [virtual]

Implements [Statistics\\_if](#).

#### 5.72.2.2 `Collector_if * StatisticsDefaultImpl1::getCollector ( )` [virtual]

Implements [Statistics\\_if](#).

Here is the caller graph for this function:



#### 5.72.2.3 `double StatisticsDefaultImpl1::halfWidthConfidenceInterval ( double confidencelevel )` [virtual]

Implements [Statistics\\_if](#).

#### 5.72.2.4 `double StatisticsDefaultImpl1::max ( )` [virtual]

Implements [Statistics\\_if](#).

#### 5.72.2.5 `double StatisticsDefaultImpl1::min ( )` [virtual]

Implements [Statistics\\_if](#).

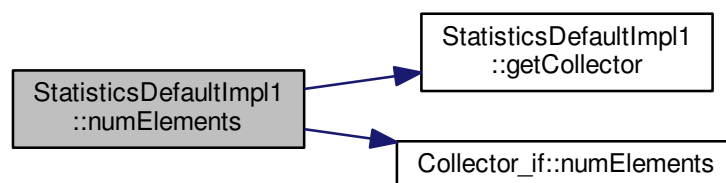
#### 5.72.2.6 `unsigned int StatisticsDefaultImpl1::newSampleSize ( double confidencelevel, double halfWidth )` [virtual]

Implements [Statistics\\_if](#).

#### 5.72.2.7 `unsigned int StatisticsDefaultImpl1::numElements ( )` [virtual]

Implements [Statistics\\_if](#).

Here is the call graph for this function:



5.72.2.8 `void StatisticsDefaultImpl1::setCollector ( Collector_if * collector )` [virtual]

Implements [Statistics\\_if](#).

5.72.2.9 `double StatisticsDefaultImpl1::stddeviation ( )` [virtual]

Implements [Statistics\\_if](#).

5.72.2.10 `double StatisticsDefaultImpl1::variance ( )` [virtual]

Implements [Statistics\\_if](#).

5.72.2.11 `double StatisticsDefaultImpl1::variationCoef ( )` [virtual]

Implements [Statistics\\_if](#).

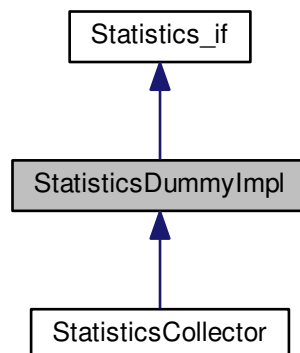
The documentation for this class was generated from the following files:

- [StatisticsDefaultImpl1.h](#)
- [StatisticsDefaultImpl1.cpp](#)

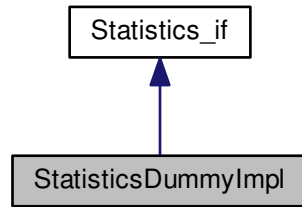
## 5.73 StatisticsDummyImpl Class Reference

```
#include <StatisticsDummyImpl.h>
```

Inheritance diagram for StatisticsDummyImpl:



Collaboration diagram for StatisticsDummyImpl:



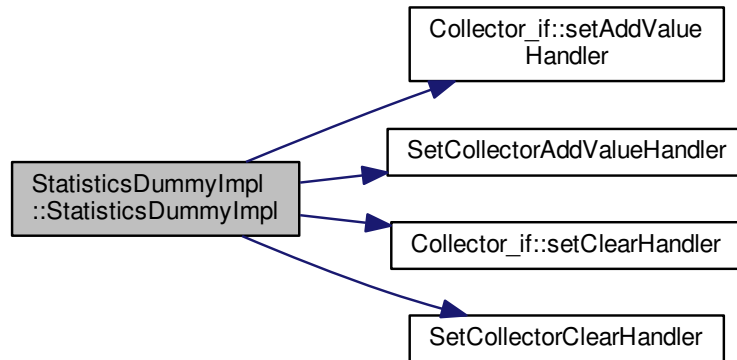
## Public Member Functions

- [StatisticsDummyImpl](#) ()
- [StatisticsDummyImpl](#) (const [StatisticsDummyImpl](#) &orig)
- virtual [~StatisticsDummyImpl](#) ()
- virtual [Collector\\_if](#) \* [getCollector](#) ()
- void [setCollector](#) ([Collector\\_if](#) \*collector)
- virtual unsigned int [numElements](#) ()
- virtual double [min](#) ()
- virtual double [max](#) ()
- virtual double [average](#) ()
- virtual double [variance](#) ()
- virtual double [stddeviation](#) ()
- virtual double [variationCoef](#) ()
- virtual double [halfWidthConfidenceInterval](#) (double confidencelevel)
- virtual unsigned int [newSampleSize](#) (double confidencelevel, double halfWidth)

### 5.73.1 Constructor & Destructor Documentation

## 5.73.1.1 StatisticsDummyImpl::StatisticsDummyImpl ( )

Here is the call graph for this function:



## 5.73.1.2 StatisticsDummyImpl::StatisticsDummyImpl ( const StatisticsDummyImpl &amp; orig )

## 5.73.1.3 StatisticsDummyImpl::~StatisticsDummyImpl ( ) [virtual]

## 5.73.2 Member Function Documentation

## 5.73.2.1 double StatisticsDummyImpl::average ( ) [virtual]

Implements [Statistics\\_if](#).

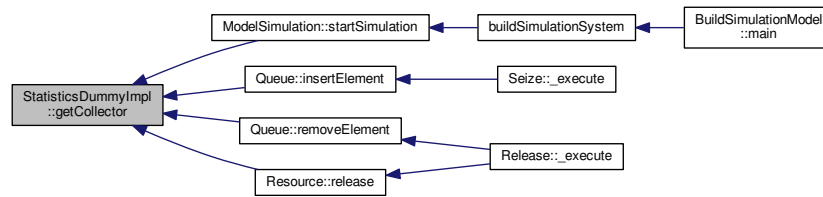
Here is the caller graph for this function:



## 5.73.2.2 Collector\_if \* StatisticsDummyImpl::getCollector ( ) [virtual]

Implements [Statistics\\_if](#).

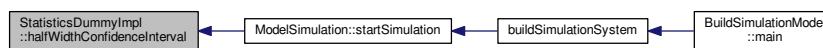
Here is the caller graph for this function:



#### 5.73.2.3 `double StatisticsDummyImpl::halfWidthConfidenceInterval ( double confidencelevel ) [virtual]`

Implements [Statistics\\_if](#).

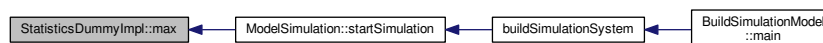
Here is the caller graph for this function:



#### 5.73.2.4 `double StatisticsDummyImpl::max ( ) [virtual]`

Implements [Statistics\\_if](#).

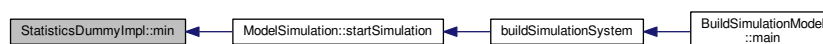
Here is the caller graph for this function:



#### 5.73.2.5 `double StatisticsDummyImpl::min ( ) [virtual]`

Implements [Statistics\\_if](#).

Here is the caller graph for this function:



5.73.2.6 unsigned int StatisticsDummyImpl::newSampleSize ( double *confidencelevel*, double *halfWidth* ) [virtual]

Implements [Statistics\\_if](#).

5.73.2.7 unsigned int StatisticsDummyImpl::numElements ( ) [virtual]

Implements [Statistics\\_if](#).

Here is the caller graph for this function:



5.73.2.8 void StatisticsDummyImpl::setCollector ( Collector\_if \* *collector* ) [virtual]

Implements [Statistics\\_if](#).

5.73.2.9 double StatisticsDummyImpl::stddeviation ( ) [virtual]

Implements [Statistics\\_if](#).

Here is the caller graph for this function:



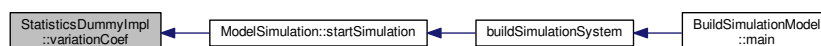
5.73.2.10 double StatisticsDummyImpl::variance ( ) [virtual]

Implements [Statistics\\_if](#).

5.73.2.11 double StatisticsDummyImpl::variationCoef ( ) [virtual]

Implements [Statistics\\_if](#).

Here is the caller graph for this function:



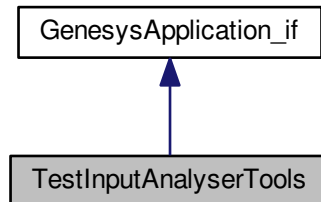
The documentation for this class was generated from the following files:

- [StatisticsDummyImpl.h](#)
- [StatisticsDummyImpl.cpp](#)

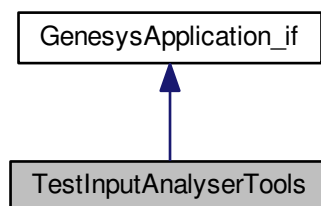
## 5.74 TestInputAnalyserTools Class Reference

```
#include <TestInputAnalyserTools.h>
```

Inheritance diagram for TestInputAnalyserTools:



Collaboration diagram for TestInputAnalyserTools:



### Public Member Functions

- [TestInputAnalyserTools](#) ()
- [main](#) (int argc, char \*\*argv)

### 5.74.1 Constructor & Destructor Documentation

5.74.1.1 [TestInputAnalyserTools::TestInputAnalyserTools](#) ( )

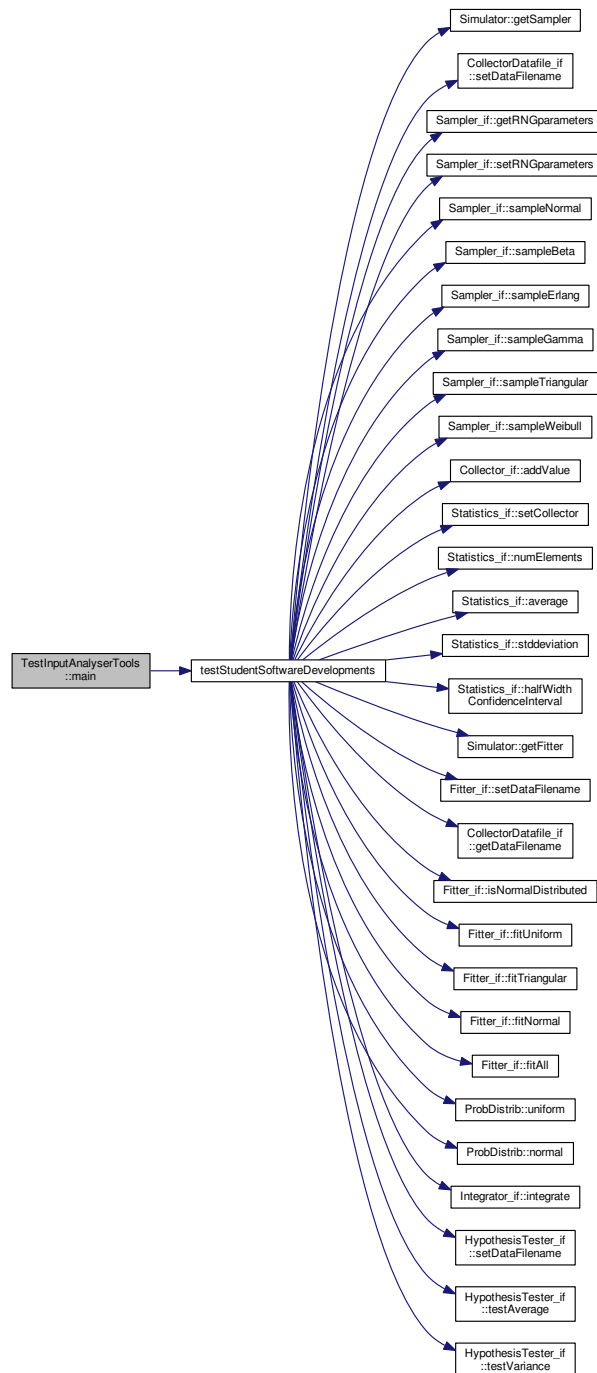
### 5.74.2 Member Function Documentation

5.74.2.1 [int TestInputAnalyserTools::main](#) ( int *argc*, char \*\* *argv* ) [virtual]

Implements [GenesysApplication\\_if](#).



Here is the call graph for this function:



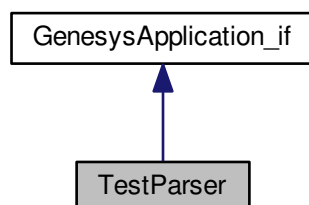
The documentation for this class was generated from the following files:

- [TestInputAnalyserTools.h](#)
- [TestInputAnalyserTools.cpp](#)

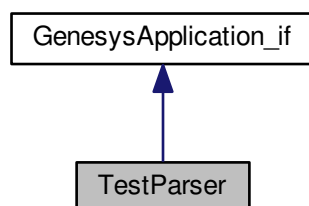
## 5.75 TestParser Class Reference

```
#include <TestParser.h>
```

Inheritance diagram for TestParser:



Collaboration diagram for TestParser:



### Public Member Functions

- [TestParser](#) ()
- [TestParser](#) (const [TestParser](#) &orig)
- virtual [~TestParser](#) ()
- virtual int [main](#) (int argc, char \*\*argv)

### 5.75.1 Constructor & Destructor Documentation

5.75.1.1 [TestParser::TestParser](#) ( )

5.75.1.2 [TestParser::TestParser](#) ( const [TestParser](#) & orig )

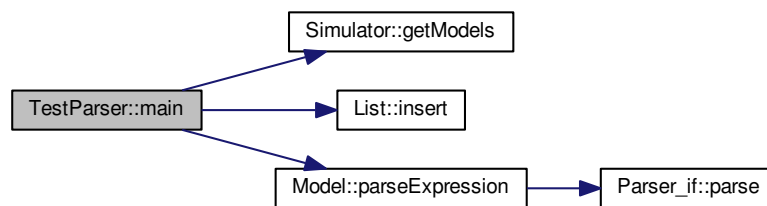
5.75.1.3 `TestParser::~TestParser ( )` [virtual]

## 5.75.2 Member Function Documentation

5.75.2.1 `int TestParser::main ( int argc, char ** argv )` [virtual]

Implements [GenesysApplication\\_if](#).

Here is the call graph for this function:



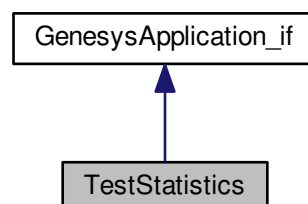
The documentation for this class was generated from the following files:

- [TestParser.h](#)
- [TestParser.cpp](#)

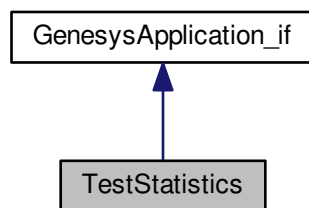
## 5.76 TestStatistics Class Reference

```
#include <TestStatistics.h>
```

Inheritance diagram for `TestStatistics`:



Collaboration diagram for TestStatistics:



## Public Member Functions

- [TestStatistics](#) ()
- int [main](#) (int argc, char \*\*argv)

### 5.76.1 Constructor & Destructor Documentation

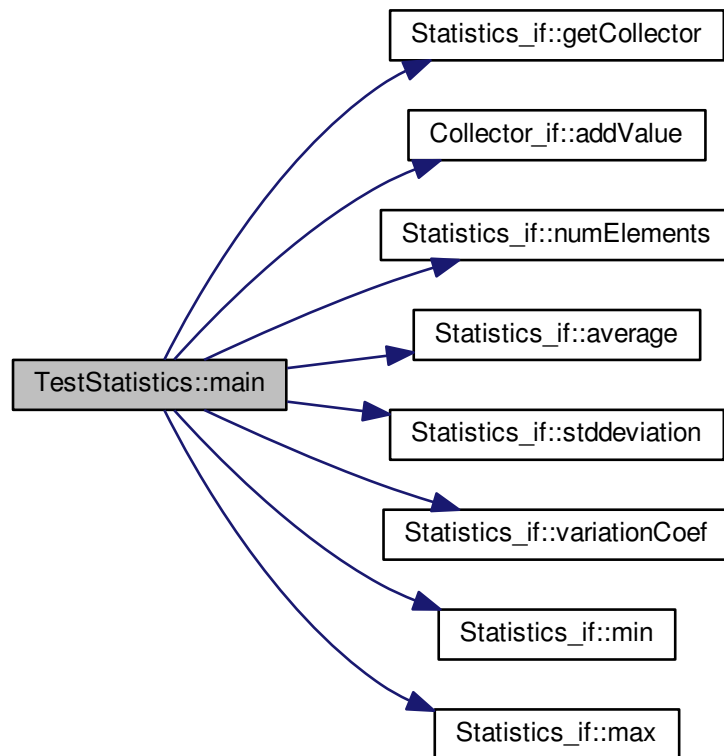
#### 5.76.1.1 TestStatistics::TestStatistics ( )

### 5.76.2 Member Function Documentation

#### 5.76.2.1 int TestStatistics::main ( int *argc*, char \*\* *argv* ) [virtual]

Implements [GenesysApplication\\_if](#).

Here is the call graph for this function:



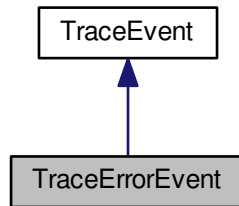
The documentation for this class was generated from the following files:

- [TestStatistics.h](#)
- [TestStatistics.cpp](#)

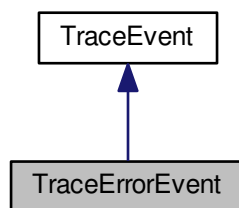
## 5.77 TraceErrorEvent Class Reference

```
#include <TraceManager.h>
```

Inheritance diagram for `TraceErrorEvent`:



Collaboration diagram for `TraceErrorEvent`:



## Public Member Functions

- [TraceErrorEvent](#) (`std::string text`, `std::exception e`)
- `std::exception` [getException](#) () const

### 5.77.1 Constructor & Destructor Documentation

5.77.1.1 `TraceErrorEvent::TraceErrorEvent ( std::string text, std::exception e )` `[inline]`

### 5.77.2 Member Function Documentation

5.77.2.1 `std::exception TraceErrorEvent::getException ( ) const` `[inline]`

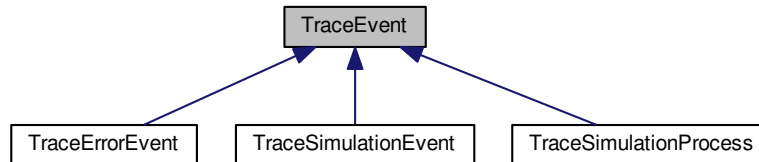
The documentation for this class was generated from the following file:

- [TraceManager.h](#)

## 5.78 TraceEvent Class Reference

```
#include <TraceManager.h>
```

Inheritance diagram for TraceEvent:



### Public Member Functions

- [TraceEvent](#) ([Util::TraceLevel](#) tracelevel, [std::string](#) text)
- [Util::TraceLevel](#) [getTracelevel](#) () const
- [std::string](#) [getText](#) () const

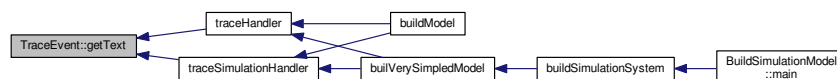
### 5.78.1 Constructor & Destructor Documentation

5.78.1.1 [TraceEvent::TraceEvent \( Util::TraceLevel tracelevel, std::string text \)](#) [\[inline\]](#)

### 5.78.2 Member Function Documentation

5.78.2.1 [std::string](#) [TraceEvent::getText \( \)](#) const [\[inline\]](#)

Here is the caller graph for this function:



5.78.2.2 [Util::TraceLevel](#) [TraceEvent::getTracelevel \( \)](#) const [\[inline\]](#)

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

- [TraceManager.h](#)

## 5.79 TraceManager Class Reference

```
#include <TraceManager.h>
```

### Public Member Functions

- [TraceManager](#) ([Model](#) \*model)
- [TraceManager](#) (const [TraceManager](#) &orig)
- virtual [~TraceManager](#) ()
- void [addTraceHandler](#) ([traceListener](#) *traceListener*)
- void [addTraceErrorHandler](#) ([traceErrorListener](#) *traceErrorListener*)
- void [addTraceReportHandler](#) ([traceListener](#) *traceReportListener*)
- void [addTraceSimulationHandler](#) ([traceSimulationListener](#) *traceSimulationListener*)
- void [trace](#) ([Util::TraceLevel](#) *tracelevel*, std::string *text*)
- void [traceError](#) (std::exception *e*, std::string *text*)
- void [traceSimulation](#) ([Util::TraceLevel](#) *tracelevel*, double *time*, [Entity](#) \**entity*, [ModelComponent](#) \**component*, std::string *text*)
- void [traceReport](#) ([Util::TraceLevel](#) *tracelevel*, std::string *text*)
- [List](#)< std::string > \* [getErrorMessages](#) () const
- void [setTraceLevel](#) ([Util::TraceLevel](#) *\_traceLevel*)
- [Util::TraceLevel](#) [getTraceLevel](#) () const

### 5.79.1 Detailed Description

The [TraceManager](#) is used to trace back model simulation information and track/debug the simulation. It works as the model simulation output (cout) and allows external methods to hook up such output as listeners.

### 5.79.2 Constructor & Destructor Documentation

5.79.2.1 [TraceManager::TraceManager](#) ( [Model](#) \* *model* )

5.79.2.2 [TraceManager::TraceManager](#) ( const [TraceManager](#) & *orig* )

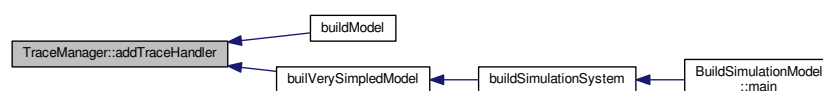
5.79.2.3 [TraceManager::~~TraceManager](#) ( ) [virtual]

### 5.79.3 Member Function Documentation

5.79.3.1 void [TraceManager::addTraceErrorHandler](#) ( [traceErrorListener](#) *traceErrorListener* )

5.79.3.2 void [TraceManager::addTraceHandler](#) ( [traceListener](#) *traceListener* )

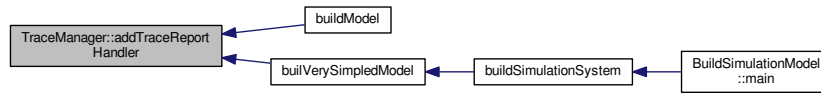
Here is the caller graph for this function:





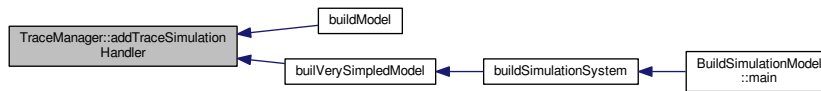
### 5.79.3.3 void TraceManager::addTraceReportHandler ( *traceListener* *traceReportListener* )

Here is the caller graph for this function:



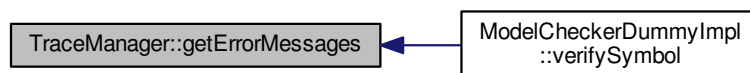
### 5.79.3.4 void TraceManager::addTraceSimulationHandler ( *traceSimulationListener* *traceSimulationListener* )

Here is the caller graph for this function:



### 5.79.3.5 List< std::string > \* TraceManager::getErrorMessage ( ) const

Here is the caller graph for this function:

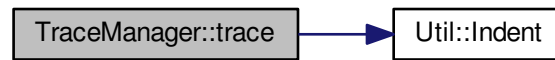


### 5.79.3.6 Util::TraceLevel TraceManager::getTraceLevel ( ) const

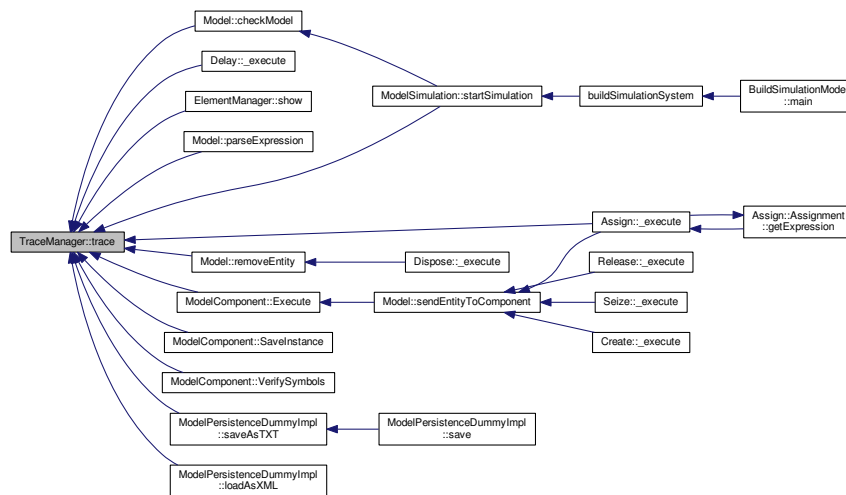
### 5.79.3.7 void TraceManager::setTraceLevel ( Util::TraceLevel *\_traceLevel* )

### 5.79.3.8 void TraceManager::trace ( Util::TraceLevel *tracelevel*, std::string *text* )

Here is the call graph for this function:

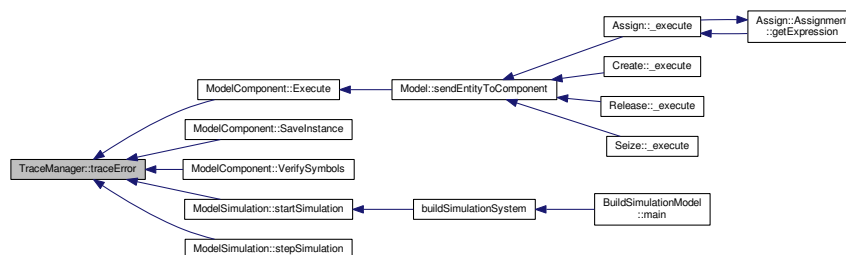


Here is the caller graph for this function:



### 5.79.3.9 void TraceManager::traceError ( std::exception *e*, std::string *text* )

Here is the caller graph for this function:

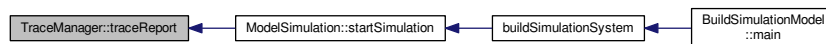


5.79.3.10 void TraceManager::traceReport ( Util::TraceLevel *tracelevel*, std::string *text* )

Here is the call graph for this function:

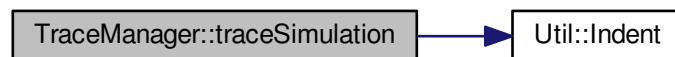


Here is the caller graph for this function:

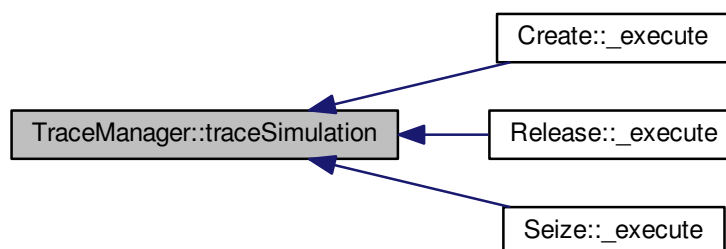


5.79.3.11 void TraceManager::traceSimulation ( Util::TraceLevel *tracelevel*, double *time*, Entity \* *entity*, ModelComponent \* *component*, std::string *text* )

Here is the call graph for this function:



Here is the caller graph for this function:



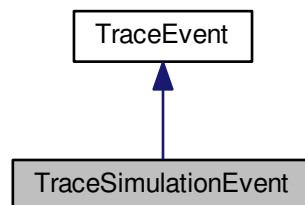
The documentation for this class was generated from the following files:

- [TraceManager.h](#)
- [TraceManager.cpp](#)

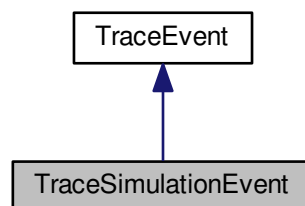
## 5.80 TraceSimulationEvent Class Reference

```
#include <TraceManager.h>
```

Inheritance diagram for TraceSimulationEvent:



Collaboration diagram for TraceSimulationEvent:



### Public Member Functions

- [ModelComponent](#) \* [getComponent](#) () const
- [Entity](#) \* [getEntity](#) () const
- double [getTime](#) () const
- [TraceSimulationEvent](#) ([Util::TraceLevel](#) tracelevel, double time, [Entity](#) \*entity, [ModelComponent](#) \*component, std::string text)

### 5.80.1 Constructor & Destructor Documentation

5.80.1.1 `TraceSimulationEvent::TraceSimulationEvent ( Util::TraceLevel tracelevel, double time, Entity * entity, ModelComponent * component, std::string text )` `[inline]`

### 5.80.2 Member Function Documentation

5.80.2.1 `ModelComponent* TraceSimulationEvent::getComponent ( ) const` `[inline]`

5.80.2.2 `Entity* TraceSimulationEvent::getEntity ( ) const` `[inline]`

5.80.2.3 `double TraceSimulationEvent::getTime ( ) const` `[inline]`

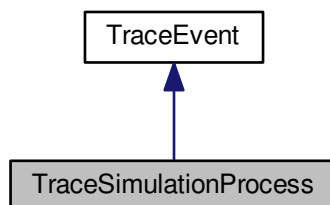
The documentation for this class was generated from the following file:

- [TraceManager.h](#)

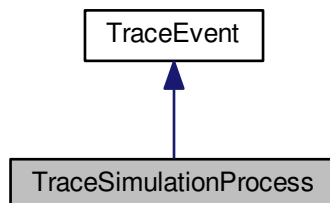
## 5.81 TraceSimulationProcess Class Reference

```
#include <TraceManager.h>
```

Inheritance diagram for TraceSimulationProcess:



Collaboration diagram for TraceSimulationProcess:



## Public Member Functions

- [TraceSimulationProcess](#) ([Util::TraceLevel](#) tracelevel, std::string text)

### 5.81.1 Detailed Description

Events related to simulation "process" (usually process analyser), associated to entire replication or simulation events (begin/end/pause of replication/simulation) TODO: CLASS NOT COMPLETE

### 5.81.2 Constructor & Destructor Documentation

5.81.2.1 `TraceSimulationProcess::TraceSimulationProcess ( Util::TraceLevel tracelevel, std::string text )` `[inline]`

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

- [TraceManager.h](#)

## 5.82 Traits< T > Struct Template Reference

```
#include <Traits.h>
```

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

- [Traits.h](#)

## 5.83 Traits< Collector\_if > Struct Template Reference

```
#include <Traits.h>
```

## Public Types

- typedef [CollectorDatafileDefaultImpl1](#) Implementation

### 5.83.1 Member Typedef Documentation

5.83.1.1 `typedef CollectorDatafileDefaultImpl1 Traits< Collector\_if >::Implementation`

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

- [Traits.h](#)

## 5.84 Traits< ExperimentDesign\_if > Struct Template Reference

```
#include <Traits.h>
```

### Public Types

- typedef [ExperimentDesignDummyImpl](#) Implementation

#### 5.84.1 Member Typedef Documentation

##### 5.84.1.1 typedef ExperimentDesignDummyImpl Traits< ExperimentDesign\_if >::Implementation

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

- [Traits.h](#)

## 5.85 Traits< Fitter\_if > Struct Template Reference

```
#include <Traits.h>
```

### Public Types

- typedef [FitterDummyImpl](#) Implementation

#### 5.85.1 Member Typedef Documentation

##### 5.85.1.1 typedef FitterDummyImpl Traits< Fitter\_if >::Implementation

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

- [Traits.h](#)

## 5.86 Traits< GenesysApplication\_if > Struct Template Reference

```
#include <Traits.h>
```

### Public Types

- typedef [BuildSimulationModel](#) Application

### 5.86.1 Member Typedef Documentation

#### 5.86.1.1 `typedef BuildSimulationModel Traits< GenesysApplication_if >::Application`

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

- [Traits.h](#)

## 5.87 Traits< HypothesisTester\_if > Struct Template Reference

```
#include <Traits.h>
```

### Public Types

- `typedef HypothesisTesterDummyImpl Implementation`

### 5.87.1 Member Typedef Documentation

#### 5.87.1.1 `typedef HypothesisTesterDummyImpl Traits< HypothesisTester_if >::Implementation`

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

- [Traits.h](#)

## 5.88 Traits< Integrator\_if > Struct Template Reference

```
#include <Traits.h>
```

### Public Types

- `typedef IntegratorDefaultImpl1 Implementation`

### Static Public Attributes

- static constexpr unsigned int [MaxIterations](#) = 1000
- static constexpr double [Precision](#) = 1e-9



### 5.88.1 Member Typedef Documentation

5.88.1.1 `typedef IntegratorDefaultImpl1 Traits< Integrator_if >::Implementation`

### 5.88.2 Member Data Documentation

5.88.2.1 `constexpr unsigned int Traits< Integrator_if >::MaxIterations = 1000` `[static]`

5.88.2.2 `constexpr double Traits< Integrator_if >::Precision = 1e-9` `[static]`

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

- [Traits.h](#)

## 5.89 Traits< Model > Struct Template Reference

```
#include <Traits.h>
```

### Static Public Attributes

- static const bool `debugged` = true
- static const `Util::TraceLevel traceLevel` = `Util::TraceLevel::mostDetailed`

### 5.89.1 Member Data Documentation

5.89.1.1 `const bool Traits< Model >::debugged = true` `[static]`

5.89.1.2 `const Util::TraceLevel Traits< Model >::traceLevel = Util::TraceLevel::mostDetailed` `[static]`

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

- [Traits.h](#)

## 5.90 Traits< ModelChecker\_if > Struct Template Reference

```
#include <Traits.h>
```

### Public Types

- typedef `ModelCheckerDummyImpl Implementation`

### 5.90.1 Member Typedef Documentation

#### 5.90.1.1 `typedef ModelCheckerDummyImpl Traits< ModelChecker_if >::Implementation`

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

- [Traits.h](#)

## 5.91 Traits< ModelComponent > Struct Template Reference

```
#include <Traits.h>
```

### Public Types

- `typedef CollectorDatafileDefaultImpl1 CollectorImplementation`
- `typedef CollectorDefaultImpl1 StatisticsCollectorImplementation`

### 5.91.1 Member Typedef Documentation

#### 5.91.1.1 `typedef CollectorDatafileDefaultImpl1 Traits< ModelComponent >::CollectorImplementation`

#### 5.91.1.2 `typedef CollectorDefaultImpl1 Traits< ModelComponent >::StatisticsCollectorImplementation`

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

- [Traits.h](#)

## 5.92 Traits< ModelPersistence\_if > Struct Template Reference

```
#include <Traits.h>
```

### Public Types

- `typedef ModelPersistenceDummyImpl Implementation`

### 5.92.1 Member Typedef Documentation

#### 5.92.1.1 `typedef ModelPersistenceDummyImpl Traits< ModelPersistence_if >::Implementation`

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

- [Traits.h](#)

## 5.93 Traits< Parser\_if > Struct Template Reference

```
#include <Traits.h>
```

### Public Types

- typedef ParserFlexBisonImpl [Implementation](#)

### 5.93.1 Member Typedef Documentation

#### 5.93.1.1 typedef ParserFlexBisonImpl Traits< Parser\_if >::Implementation

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

- [Traits.h](#)

## 5.94 Traits< ProcessAnalyser\_if > Struct Template Reference

```
#include <Traits.h>
```

### Public Types

- typedef ProcessAnalyserDummyImpl [Implementation](#)

### 5.94.1 Member Typedef Documentation

#### 5.94.1.1 typedef ProcessAnalyserDummyImpl Traits< ProcessAnalyser\_if >::Implementation

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

- [Traits.h](#)

## 5.95 Traits< Sampler\_if > Struct Template Reference

```
#include <Traits.h>
```

### Public Types

- typedef SamplerDefaultImpl1 [Implementation](#)
- typedef SamplerDefaultImpl1::DefaultImpl1RNG\_Parameters [Parameters](#)

### 5.95.1 Member Typedef Documentation

5.95.1.1 `typedef SamplerDefaultImpl1 Traits< Sampler_if >::Implementation`

5.95.1.2 `typedef SamplerDefaultImpl1::DefaultImpl1RNG_Parameters Traits< Sampler_if >::Parameters`

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

- [Traits.h](#)

## 5.96 Traits< Statistics\_if > Struct Template Reference

```
#include <Traits.h>
```

### Public Types

- typedef [StatisticsDefaultImpl1 Implementation](#)
- typedef [CollectorDefaultImpl1 CollectorImplementation](#)

### 5.96.1 Member Typedef Documentation

5.96.1.1 `typedef CollectorDefaultImpl1 Traits< Statistics_if >::CollectorImplementation`

5.96.1.2 `typedef StatisticsDefaultImpl1 Traits< Statistics_if >::Implementation`

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

- [Traits.h](#)

## 5.97 Util Class Reference

```
#include <Util.h>
```

### Public Types

- enum [TimeUnit](#) : int {  
[TimeUnit::picosecond](#) = 1, [TimeUnit::nanosecond](#) = 2, [TimeUnit::microsecond](#) = 3, [TimeUnit::millisecond](#) = 4,  
[TimeUnit::second](#) = 5, [TimeUnit::minute](#) = 6, [TimeUnit::hour](#) = 7, [TimeUnit::day](#) = 8,  
[TimeUnit::week](#) = 9 }
- enum [TraceLevel](#) : int {  
[TraceLevel::noTraces](#) = 0, [TraceLevel::errors](#) = 10, [TraceLevel::report](#) = 20, [TraceLevel::simulation](#) = 30,  
[TraceLevel::transferOnly](#) = 40, [TraceLevel::blockArrival](#) = 50, [TraceLevel::blockInternal](#) = 60, [TraceLevel::mostDetailed](#) = 70 }
- typedef unsigned long [identification](#)
- typedef unsigned int [rank](#)

## Static Public Member Functions

- static void [IncIndent](#) ()
- static void [DeclIndent](#) ()
- static std::string [Indent](#) ()
- static [Util::identification](#) [GenerateNewId](#) ()
- static double [TimeUnitConvert](#) ([Util::TimeUnit](#) timeUnit1, [Util::TimeUnit](#) timeUnit2)
- static [Util::identification](#) [GenerateNewIdOfType](#) (std::string objtyp)
- template<class T >  
static std::string [TypeOf](#) ()
- template<class T >  
static [Util::identification](#) [GenerateNewIdOfType](#) ()

## 5.97.1 Member Typedef Documentation

5.97.1.1 typedef unsigned long [Util::identification](#)

5.97.1.2 typedef unsigned int [Util::rank](#)

## 5.97.2 Member Enumeration Documentation

5.97.2.1 enum [Util::TimeUnit](#) : int [strong]

Enumerator

***picosecond***  
***nanosecond***  
***microsecond***  
***millisecond***  
***second***  
***minute***  
***hour***  
***day***  
***week***

5.97.2.2 enum [Util::TraceLevel](#) : int [strong]

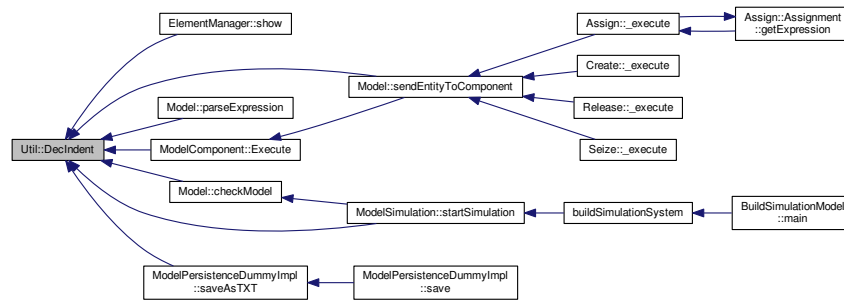
Enumerator

***noTraces***  
***errors***  
***report***  
***simulation***  
***transferOnly***  
***blockArrival***  
***blockInternal***  
***mostDetailed***

### 5.97.3 Member Function Documentation

#### 5.97.3.1 void Util::DeclIndent ( ) [static]

Here is the caller graph for this function:



#### 5.97.3.2 Util::identification Util::GenerateNewId ( ) [static]

#### 5.97.3.3 Util::identification Util::GenerateNewIdOfType ( std::string objtyp ) [static]

#### 5.97.3.4 template<class T> static Util::identification Util::GenerateNewIdOfType ( ) [inline],[static]

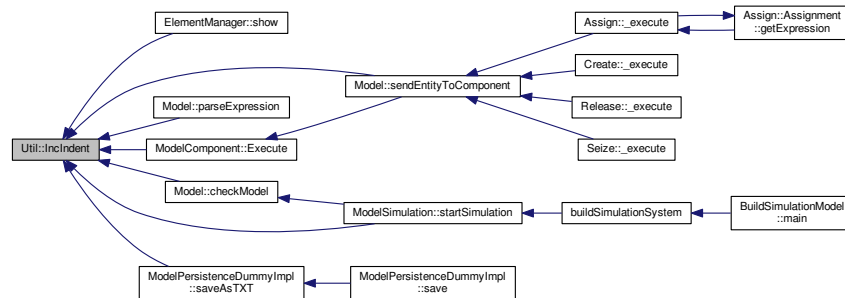
Every component or element has a unique ID for its class, but not unique for other classes. IDs are generated sequentially for each class.

Here is the caller graph for this function:



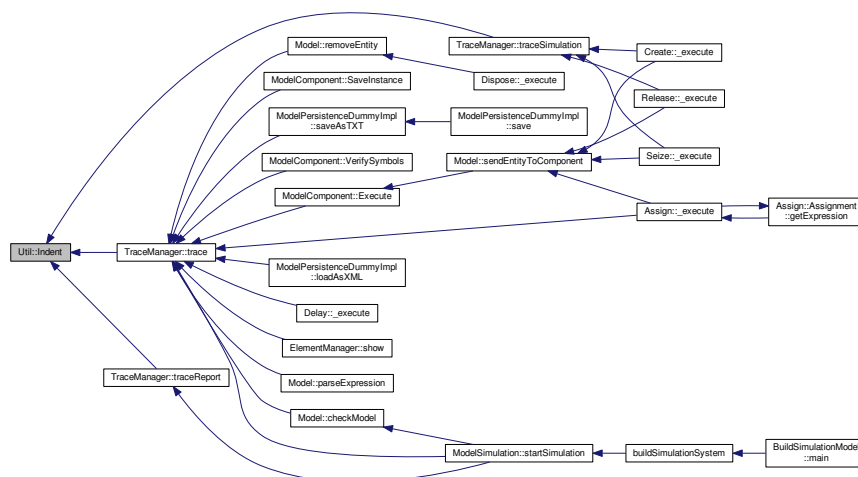
## 5.97.3.5 void Util::IncIndent ( ) [static]

Here is the caller graph for this function:



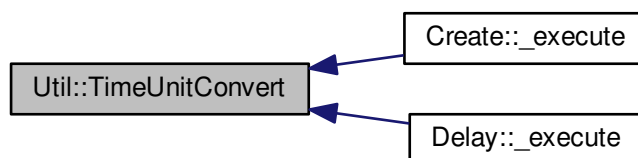
## 5.97.3.6 std::string Util::Indent ( ) [static]

Here is the caller graph for this function:



5.97.3.7 `double Util::TimeUnitConvert ( Util::TimeUnit timeUnit1, Util::TimeUnit timeUnit2 )` `[static]`

Here is the caller graph for this function:



5.97.3.8 `template<class T> static std::string Util::TypeOf ( )` `[inline],[static]`

Return the name of the class used as T.

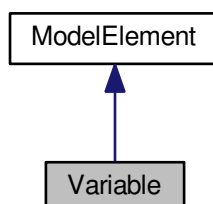
The documentation for this class was generated from the following files:

- [Util.h](#)
- [Util.cpp](#)

## 5.98 Variable Class Reference

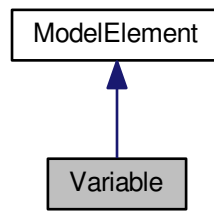
```
#include <Variable.h>
```

Inheritance diagram for Variable:





Collaboration diagram for Variable:



## Public Member Functions

- [Variable](#) ()
- [Variable](#) (std::string name)
- [Variable](#) (const [Variable](#) &orig)
- virtual [~Variable](#) ()
- virtual std::string [show](#) ()
- double [getValue](#) ()
- double [getValue](#) (std::string index)
- void [setValue](#) (double value)
- void [setValue](#) (std::string index, double value)

## Protected Member Functions

- virtual void [\\_loadInstance](#) (std::list< std::string > words)
- virtual std::list< std::string > \* [\\_saveInstance](#) ()
- virtual bool [\\_verifySymbols](#) (std::string \*errorMessage)

## Additional Inherited Members

### 5.98.1 Constructor & Destructor Documentation

5.98.1.1 `Variable::Variable ( )`

5.98.1.2 `Variable::Variable ( std::string name )`

5.98.1.3 `Variable::Variable ( const Variable & orig )`

5.98.1.4 `Variable::~~Variable ( )` `[virtual]`

### 5.98.2 Member Function Documentation

5.98.2.1 `void Variable::\_loadInstance ( std::list< std::string > words )` `[protected]`, `[virtual]`

Implements [ModelElement](#).

5.98.2.2 `std::list< std::string > * Variable::_saveInstance ( )` [protected],[virtual]

Reimplemented from [ModelElement](#).

Here is the call graph for this function:



5.98.2.3 `bool Variable::_verifySymbols ( std::string * errorMessage )` [protected],[virtual]

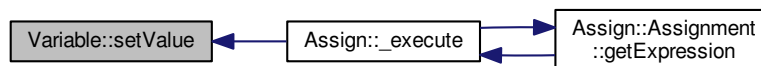
Implements [ModelElement](#).

5.98.2.4 `double Variable::getValue ( )`

5.98.2.5 `double Variable::getValue ( std::string index )`

5.98.2.6 `void Variable::setValue ( double value )`

Here is the caller graph for this function:

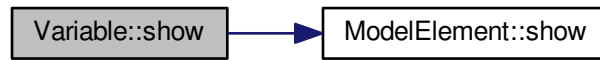


5.98.2.7 `void Variable::setValue ( std::string index, double value )`

5.98.2.8 `std::string Variable::show ( )` [virtual]

Reimplemented from [ModelElement](#).

Here is the call graph for this function:



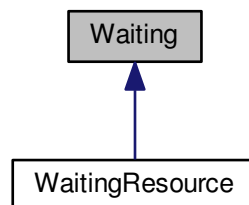
The documentation for this class was generated from the following files:

- [Variable.h](#)
- [Variable.cpp](#)

## 5.99 Waiting Class Reference

```
#include <Waiting.h>
```

Inheritance diagram for Waiting:



### Public Member Functions

- [Waiting](#) ([Entity](#) \*entity, [ModelComponent](#) \*component, double timeStartedWaiting)
- [Waiting](#) (const [Waiting](#) &orig)
- virtual [~Waiting](#) ()
- virtual std::string [show](#) ()
- double [getTimeStartedWaiting](#) () const
- [ModelComponent](#) \* [getComponent](#) () const
- [Entity](#) \* [getEntity](#) () const

### 5.99.1 Constructor & Destructor Documentation

5.99.1.1 `Waiting::Waiting ( Entity * entity, ModelComponent * component, double timeStartedWaiting )`

5.99.1.2 `Waiting::Waiting ( const Waiting & orig )`

5.99.1.3 `Waiting::~Waiting ( ) [virtual]`

### 5.99.2 Member Function Documentation

5.99.2.1 `ModelComponent * Waiting::getComponent ( ) const`

Here is the caller graph for this function:



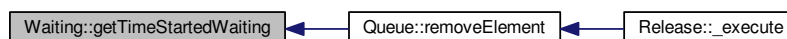
5.99.2.2 `Entity * Waiting::getEntity ( ) const`

Here is the caller graph for this function:



5.99.2.3 `double Waiting::getTimeStartedWaiting ( ) const`

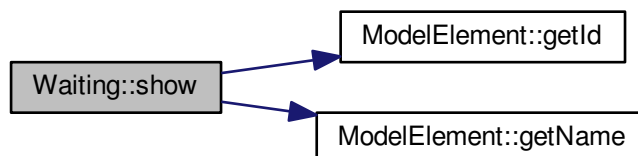
Here is the caller graph for this function:



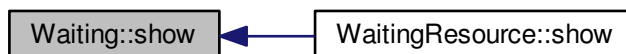
5.99.2.4 `std::string Waiting::show ( )` [virtual]

Reimplemented in [WaitingResource](#).

Here is the call graph for this function:



Here is the caller graph for this function:



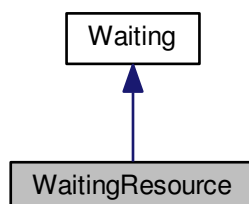
The documentation for this class was generated from the following files:

- [Waiting.h](#)
- [Waiting.cpp](#)

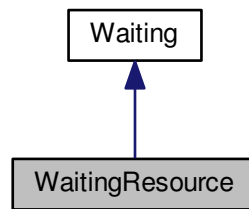
## 5.100 WaitingResource Class Reference

```
#include <WaitingResource.h>
```

Inheritance diagram for `WaitingResource`:



Collaboration diagram for WaitingResource:



## Public Member Functions

- [WaitingResource](#) ([Entity](#) \*entity, [ModelComponent](#) \*component, double timeStartedWaiting, unsigned int quantity)
- [WaitingResource](#) (const [WaitingResource](#) &orig)
- virtual [~WaitingResource](#) ()
- virtual std::string [show](#) ()
- unsigned int [getQuantity](#) () const

### 5.100.1 Constructor & Destructor Documentation

5.100.1.1 `WaitingResource::WaitingResource ( Entity * entity, ModelComponent * component, double timeStartedWaiting, unsigned int quantity )`

5.100.1.2 `WaitingResource::WaitingResource ( const WaitingResource & orig )`

5.100.1.3 `WaitingResource::~~WaitingResource ( )` [virtual]

### 5.100.2 Member Function Documentation

5.100.2.1 `unsigned int WaitingResource::getQuantity ( ) const`

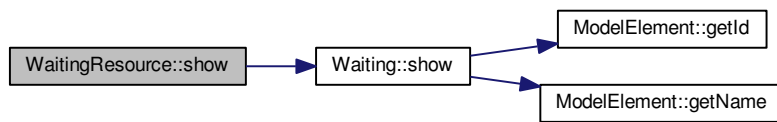
Here is the caller graph for this function:



### 5.100.2.2 `std::string WaitingResource::show ( )` [virtual]

Reimplemented from [Waiting](#).

Here is the call graph for this function:



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

- [WaitingResource.h](#)
- [WaitingResource.cpp](#)





## Chapter 6

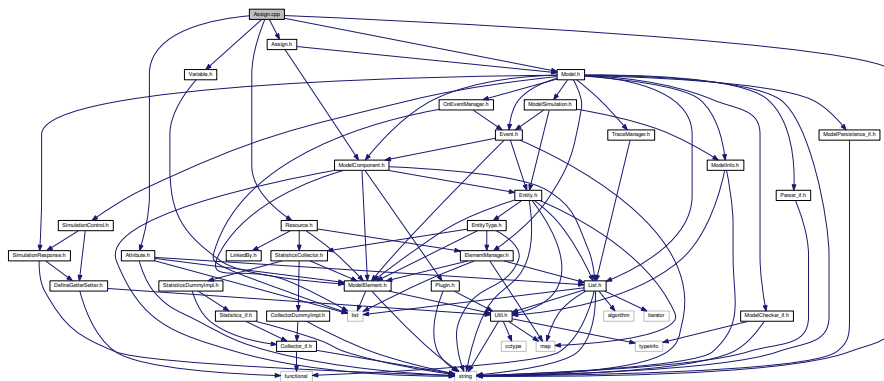
# File Documentation

### 6.1 .dep.inc File Reference

### 6.2 Assign.cpp File Reference

```
#include <string>
#include "Model.h"
#include "Assign.h"
#include "Variable.h"
#include "Attribute.h"
#include "Resource.h"
```

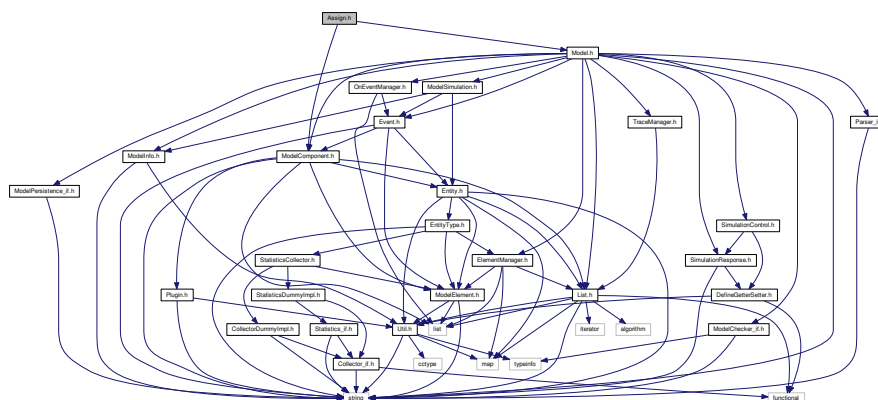
Include dependency graph for Assign.cpp:



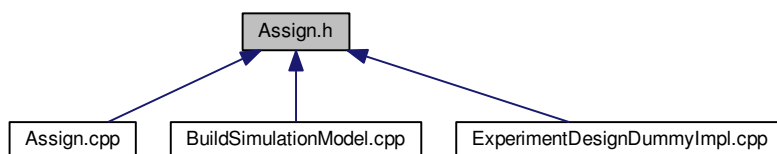
### 6.3 Assign.h File Reference

```
#include "ModelComponent.h"
#include "Model.h"
```

Include dependency graph for Assign.h:



This graph shows which files directly or indirectly include this file:



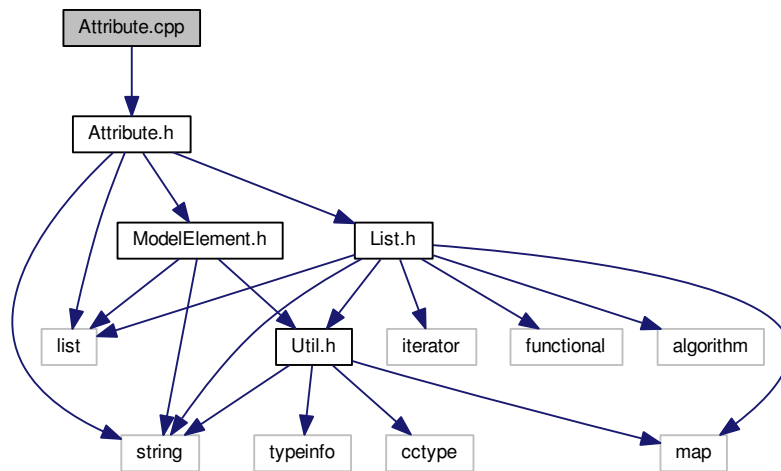
## Classes

- class Assign
- class Assign::Assignment

## 6.4 Attribute.cpp File Reference

```
#include "Attribute.h"
```

Include dependency graph for Attribute.cpp:



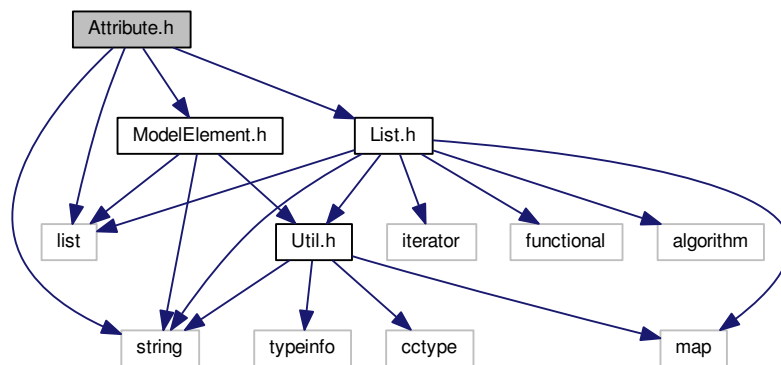
## 6.5 Attribute.h File Reference

```

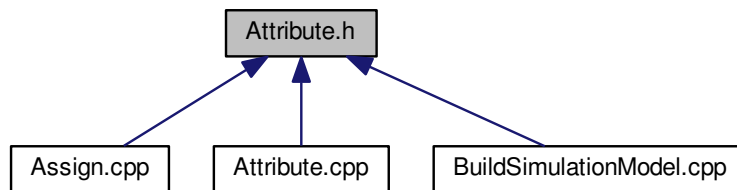
#include <string>
#include <list>
#include "List.h"
#include "ModelElement.h"

```

Include dependency graph for Attribute.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Attribute](#)

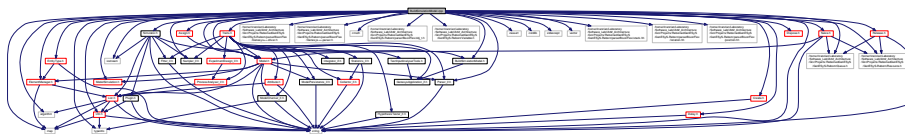
## 6.6 BuildSimulationModel.cpp File Reference

```

#include "BuildSimulationModel.h"
#include "Simulator.h"
#include "Traits.h"
#include "Create.h"
#include "Delay.h"
#include "Dispose.h"
#include "Seize.h"
#include "Release.h"
#include "Assign.h"
#include "ModelSimulation.h"
#include "ElementManager.h"
#include "EntityType.h"
#include "Attribute.h"

```

Include dependency graph for BuildSimulationModel.cpp:



## Functions

- void [traceHandler](#) ([TraceEvent](#) e)
- void [traceSimulationHandler](#) ([TraceSimulationEvent](#) e)
- void [onSimulationStartHandler](#) ([SimulationEvent](#) \*re)
- void [onReplicationStartHandler](#) ([SimulationEvent](#) \*re)
- void [onProcessEventHandler](#) ([SimulationEvent](#) \*re)
- void [onReplicationEndHandler](#) ([SimulationEvent](#) \*re)
- void [onEntityRemoveHandler](#) ([SimulationEvent](#) \*re)
- void [buildModel](#) ([Model](#) \*model)
- void [builVerySimplifiedModel](#) ([Model](#) \*model)
- void [buildSimulationSystem](#) ()

## 6.6.1 Function Documentation

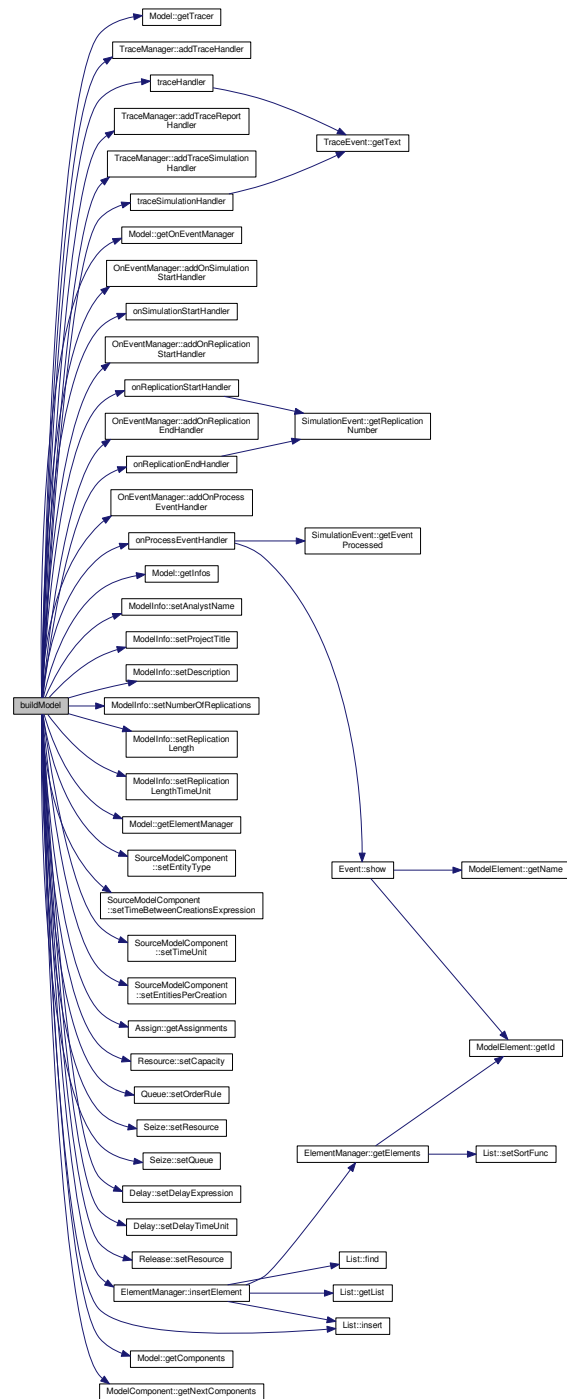
### 6.6.1.1 void buildModel ( Model \* *model* )

This function shows an example of how to create a simulation model. It creates some handlers for tracing (debug) and for events, set model infos and than creates the model itself. The model is a composition of components (and elements that they use), connected to form a process/fluxogram

#### Parameters

<i>model</i>	- The instance returned that will contains the built model
--------------	--

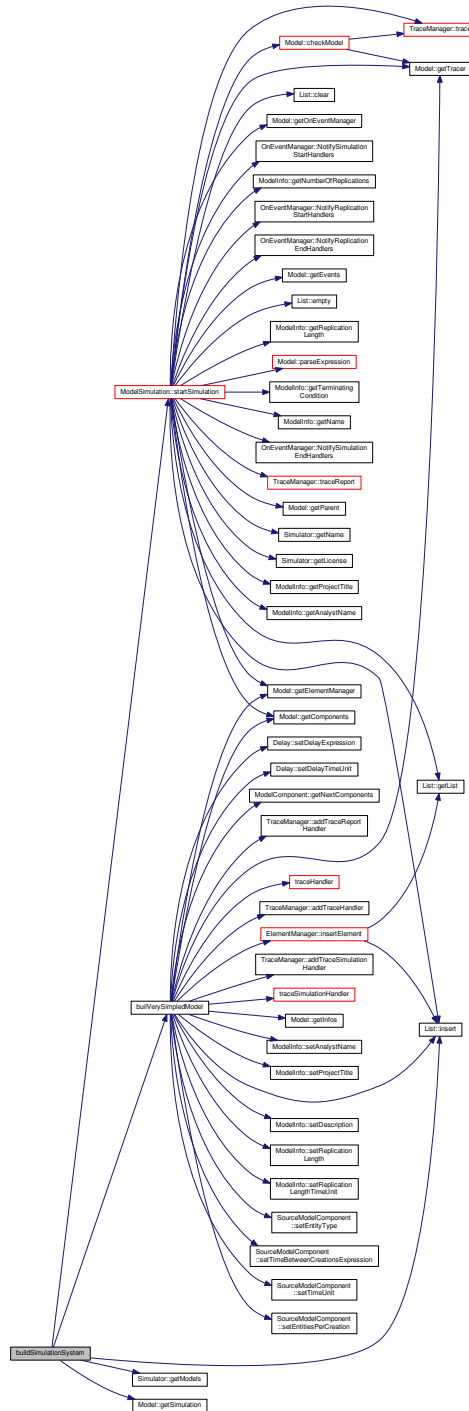
Here is the call graph for this function:



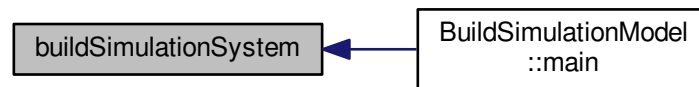
#### 6.6.1.2 void buildSimulationSystem ( )

This is the main function of the [BuildSimulationModel](#) application. It instantiates the simulator, builds a simulation model and then simulate that model.

Here is the call graph for this function:



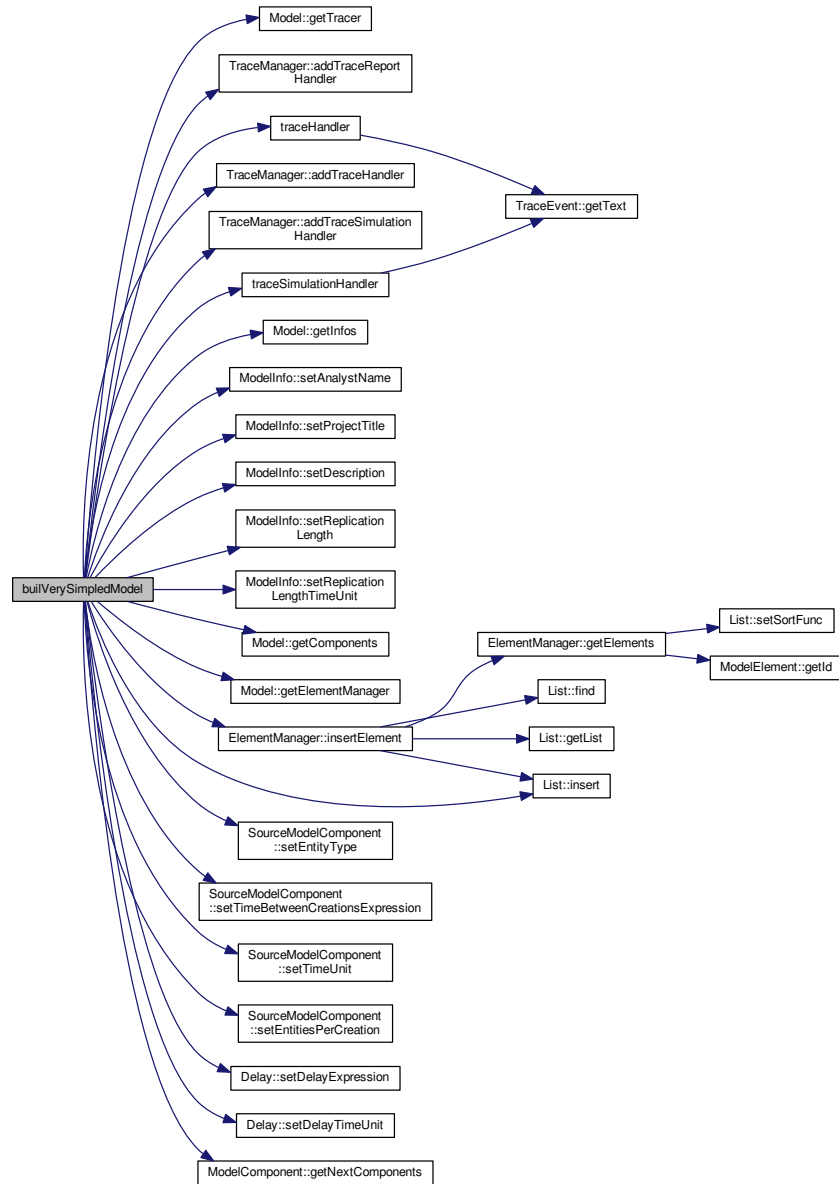
Here is the caller graph for this function:



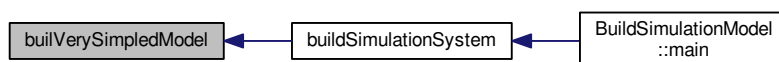


## 6.6.1.3 void builVerySimpdModel ( Model \* model )

Here is the call graph for this function:

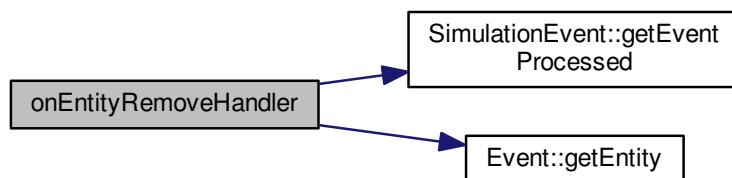


Here is the caller graph for this function:



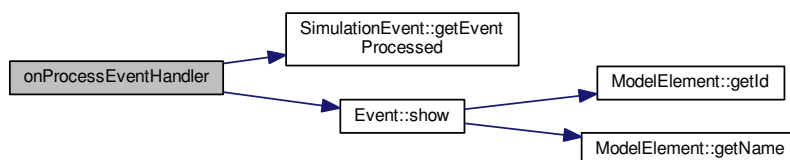
#### 6.6.1.4 void onEntityRemoveHandler ( SimulationEvent \* re )

Here is the call graph for this function:

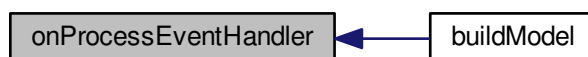


#### 6.6.1.5 void onProcessEventHandler ( SimulationEvent \* re )

Here is the call graph for this function:

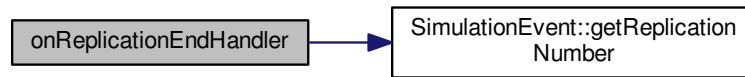


Here is the caller graph for this function:

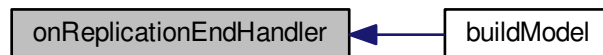


#### 6.6.1.6 void onReplicationEndHandler ( SimulationEvent \* re )

Here is the call graph for this function:

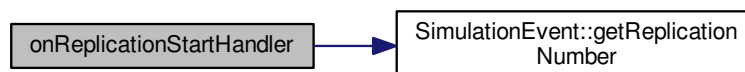


Here is the caller graph for this function:



#### 6.6.1.7 void onReplicationStartHandler ( SimulationEvent \* re )

Here is the call graph for this function:

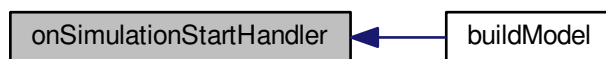


Here is the caller graph for this function:



#### 6.6.1.8 void onSimulationStartHandler ( SimulationEvent \* re )

Here is the caller graph for this function:

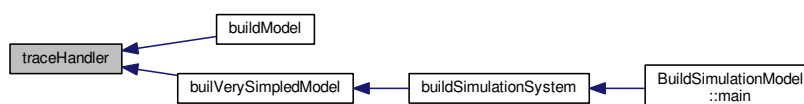


#### 6.6.1.9 void traceHandler ( TraceEvent e )

Here is the call graph for this function:



Here is the caller graph for this function:

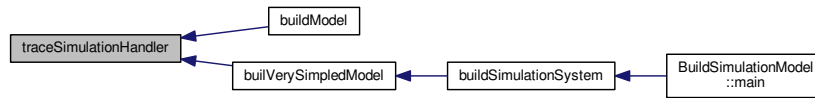


#### 6.6.1.10 void traceSimulationHandler ( TraceSimulationEvent e )

Here is the call graph for this function:

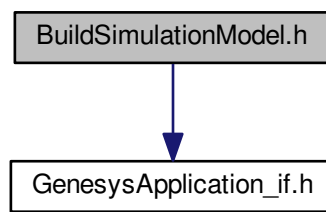


Here is the caller graph for this function:

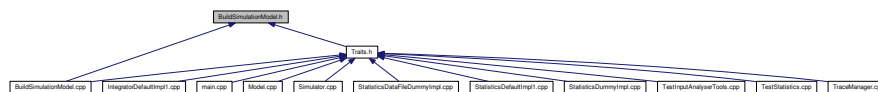


## 6.7 BuildSimulationModel.h File Reference

```
#include "GenesysApplication_if.h"
Include dependency graph for BuildSimulationModel.h:
```



This graph shows which files directly or indirectly include this file:



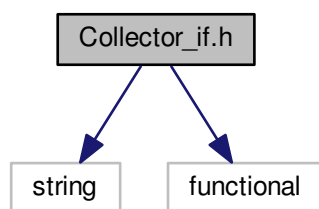
### Classes

- class [BuildSimulationModel](#)

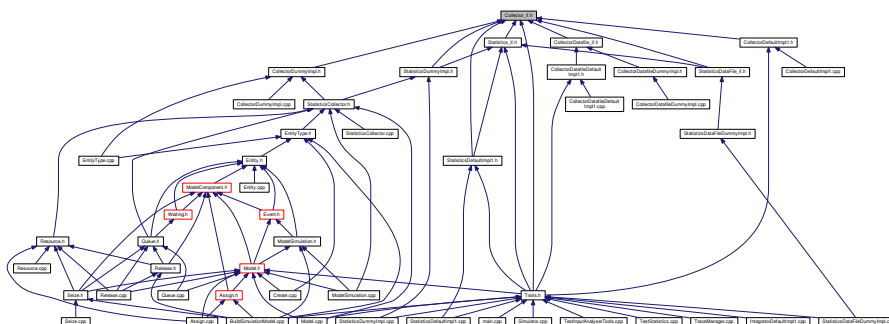
## 6.8 Collector\_if.h File Reference

```
#include <string>
#include <functional>
```

Include dependency graph for `Collector_if.h`:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Collector\\_if](#)

## Typedefs

- typedef `std::function< void(double) >` [CollectorAddValueHandler](#)
- typedef `std::function< void() >` [CollectorClearHandler](#)

## Functions

- template<typename Class >  
[CollectorAddValueHandler](#) [SetCollectorAddValueHandler](#) (void(Class::\*function)(double), Class \*object)
- template<typename Class >  
[CollectorClearHandler](#) [SetCollectorClearHandler](#) (void(Class::\*function)(), Class \*object)

## 6.8.1 Typedef Documentation

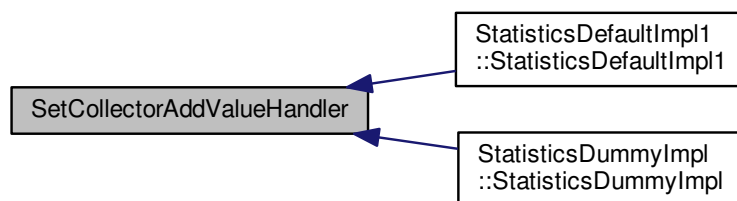
6.8.1.1 `typedef std::function<void(double)> CollectorAddValueHandler`

6.8.1.2 `typedef std::function<void()> CollectorClearHandler`

## 6.8.2 Function Documentation

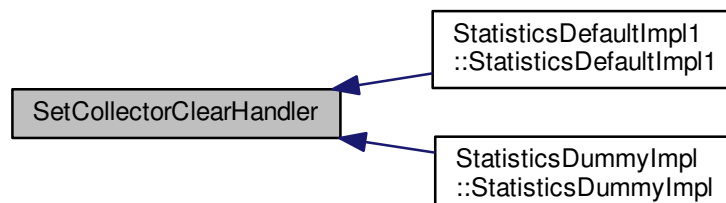
6.8.2.1 `template<typename Class> CollectorAddValueHandler SetCollectorAddValueHandler ( void(Class::*)(double) function, Class * object )`

Here is the caller graph for this function:



6.8.2.2 `template<typename Class> CollectorClearHandler SetCollectorClearHandler ( void(Class::*)() function, Class * object )`

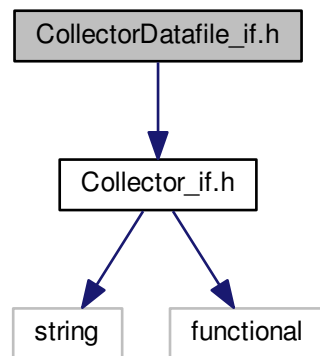
Here is the caller graph for this function:



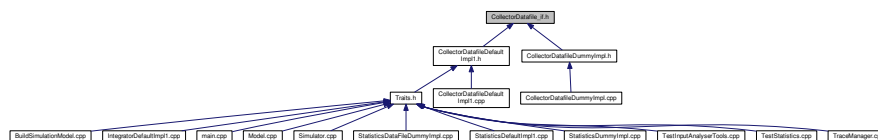
## 6.9 CollectorDatafile\_if.h File Reference

```
#include "Collector_if.h"
```

Include dependency graph for CollectorDatafile\_if.h:



This graph shows which files directly or indirectly include this file:



### Classes

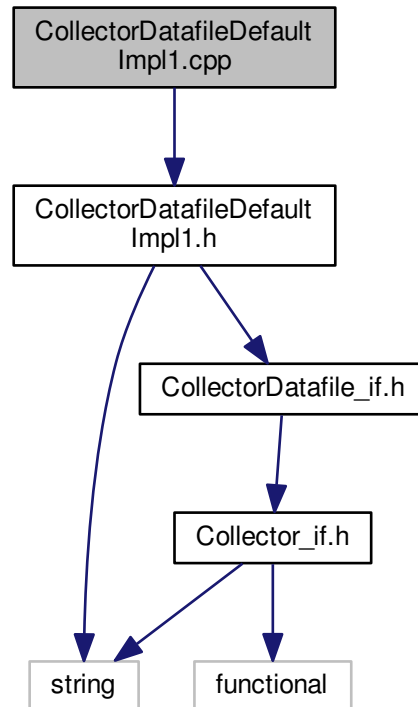
- class [CollectorDatafile\\_if](#)

## 6.10 CollectorDatafileDefaultImpl1.cpp File Reference

```
#include "CollectorDatafileDefaultImpl1.h"
```



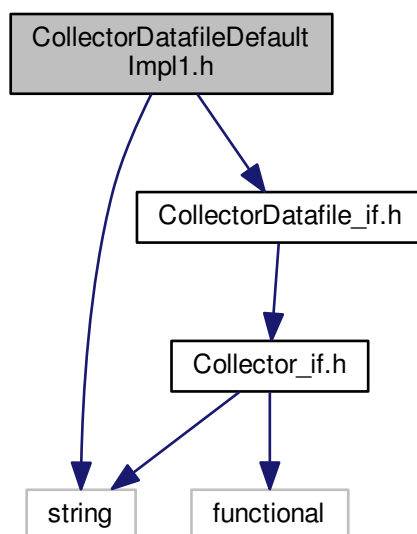
Include dependency graph for CollectorDatafileDefaultImpl1.cpp:



## 6.11 CollectorDatafileDefaultImpl1.h File Reference

```
#include <string>
#include "CollectorDatafile_if.h"
```

Include dependency graph for CollectorDatafileDefaultImpl1.h:



This graph shows which files directly or indirectly include this file:



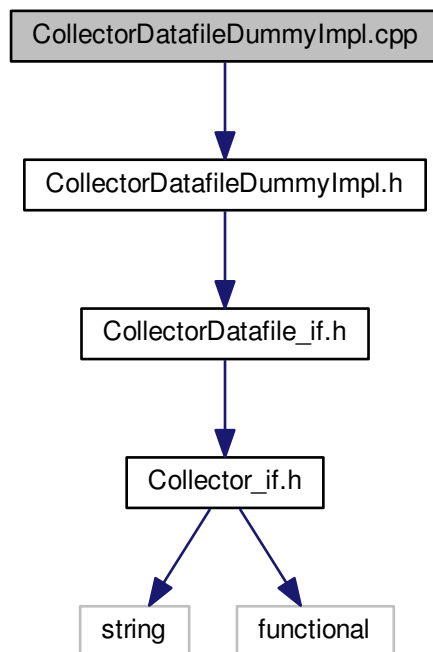
## Classes

- class [CollectorDatafileDefaultImpl1](#)

## 6.12 CollectorDatafileDummyImpl.cpp File Reference

```
#include "CollectorDatafileDummyImpl.h"
```

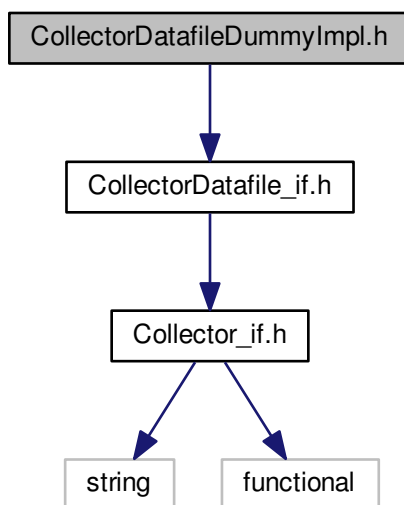
Include dependency graph for CollectorDatafileDummyImpl.cpp:



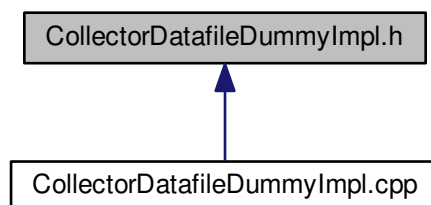
## 6.13 CollectorDatafileDummyImpl.h File Reference

```
#include "CollectorDatafile_if.h"
```

Include dependency graph for CollectorDatafileDummyImpl.h:



This graph shows which files directly or indirectly include this file:



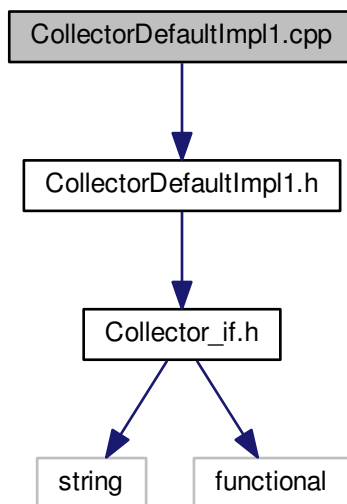
## Classes

- class [CollectorDatafileDummyImpl](#)

## 6.14 CollectorDefaultImpl1.cpp File Reference

```
#include "CollectorDefaultImpl1.h"
```

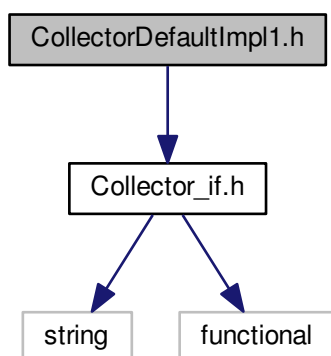
Include dependency graph for CollectorDefaultImpl1.cpp:



## 6.15 CollectorDefaultImpl1.h File Reference

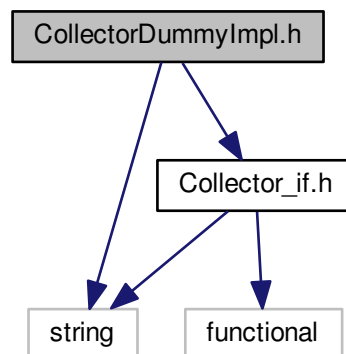
```
#include "Collector_if.h"
```

Include dependency graph for CollectorDefaultImpl1.h:

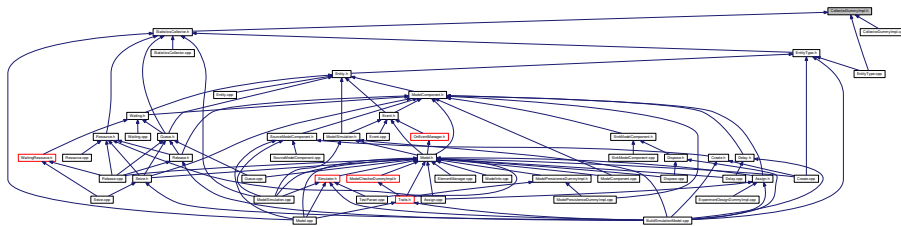




Include dependency graph for CollectorDummyImpl.h:



This graph shows which files directly or indirectly include this file:



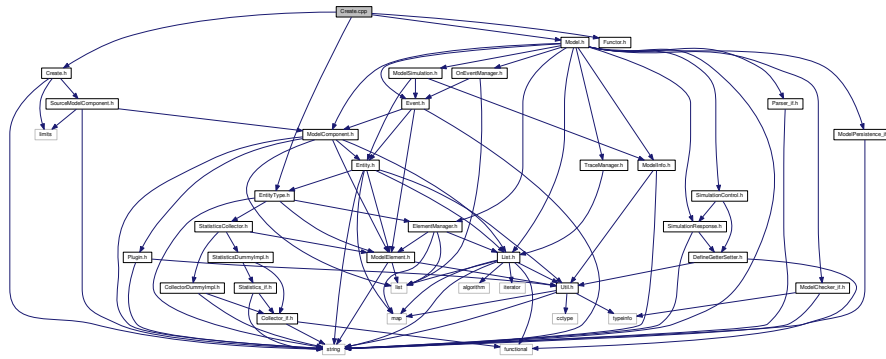
## Classes

- class `CollectorDummyImpl`

## 6.18 Create.cpp File Reference

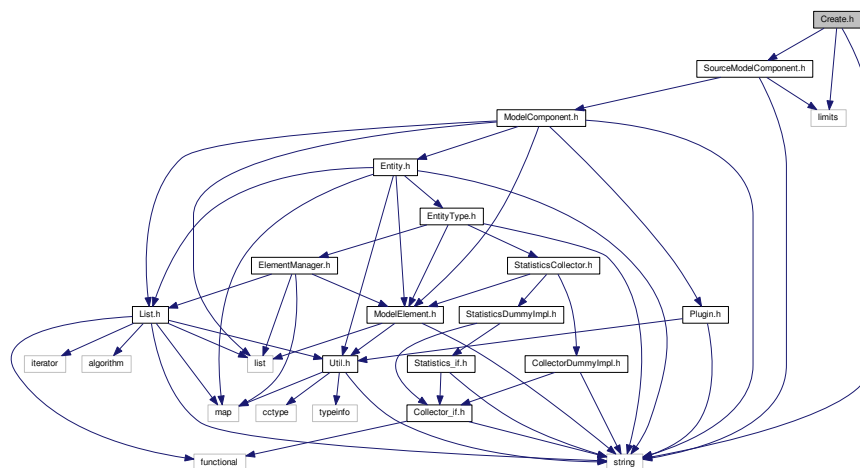
```
#include "Create.h"
#include "Model.h"
#include "EntityType.h"
#include "Functor.h"
```

Include dependency graph for Create.cpp:



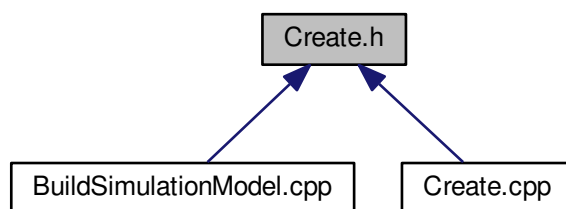
## 6.19 Create.h File Reference

```
#include <string>
#include <limits>
#include "SourceModelComponent.h"
Include dependency graph for Create.h:
```





This graph shows which files directly or indirectly include this file:



### Classes

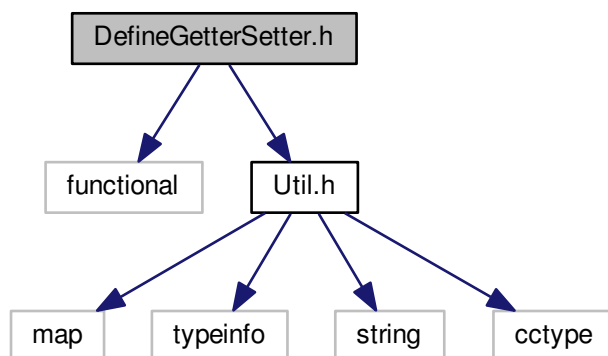
- class [Create](#)

## 6.20 DefineGetterSetter.h File Reference

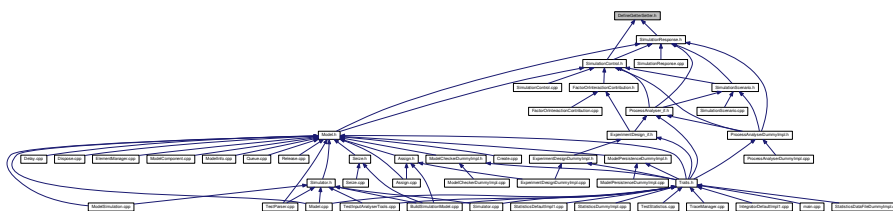
```
#include <functional>
```

```
#include "Util.h"
```

Include dependency graph for `DefineGetterSetter.h`:



This graph shows which files directly or indirectly include this file:



## Typedefs

- typedef std::function< double() > [GetterMember](#)
- typedef std::function< void(double) > [SetterMember](#)

## Functions

- template<typename Class >  
[GetterMember DefineGetterMember](#) (Class \*object, double(Class::\*function)())
- template<typename Class >  
[SetterMember DefineSetterMember](#) (Class \*object, void(Class::\*function)(double))
- template<typename Class >  
[GetterMember DefineGetterMember](#) (Class \*object, unsigned int(Class::\*function)() const)
- template<typename Class >  
[SetterMember DefineSetterMember](#) (Class \*object, void(Class::\*function)(unsigned int))
- template<typename Class >  
[GetterMember DefineGetterMember](#) (Class \*object, bool(Class::\*function)() const)
- template<typename Class >  
[SetterMember DefineSetterMember](#) (Class \*object, void(Class::\*function)(bool))
- template<typename Class >  
[GetterMember DefineGetterMember](#) (Class \*object, std::string(Class::\*function)() const)
- template<typename Class >  
[SetterMember DefineSetterMember](#) (Class \*object, void(Class::\*function)(std::string))
- template<typename Class >  
[GetterMember DefineGetterMember](#) (Class \*object, [Util::TimeUnit](#)(Class::\*function)() const)
- template<typename Class >  
[SetterMember DefineSetterMember](#) (Class \*object, void(Class::\*function)([Util::TimeUnit](#)))

### 6.20.1 Typedef Documentation

6.20.1.1 typedef std::function<double() > [GetterMember](#)

6.20.1.2 typedef std::function<void(double) > [SetterMember](#)

### 6.20.2 Function Documentation

6.20.2.1 template<typename Class > [GetterMember DefineGetterMember](#) ( Class \* *object*, double(Class::\*)*function* )

6.20.2.2 template<typename Class > [GetterMember DefineGetterMember](#) ( Class \* *object*, unsigned int(Class::\*)*function* )

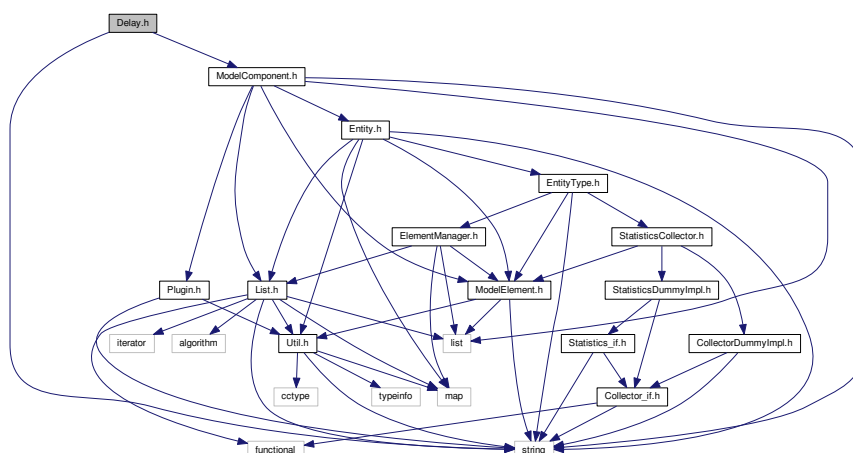
6.20.2.3 template<typename Class > [GetterMember DefineGetterMember](#) ( Class \* *object*, bool(Class::\*)*function* )

6.20.2.4 template<typename Class > [GetterMember DefineGetterMember](#) ( Class \* *object*, std::string(Class::\*)*function* )

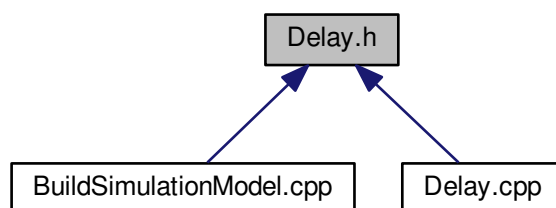
6.20.2.5 template<typename Class > [GetterMember DefineGetterMember](#) ( Class \* *object*, [Util::TimeUnit](#)(Class::\*)*function* )



Include dependency graph for Delay.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Delay](#)

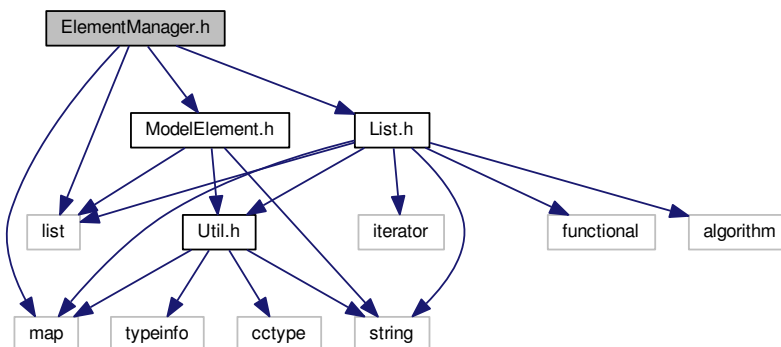
## 6.23 Dispose.cpp File Reference

```
#include "Dispose.h"
#include "Model.h"
```

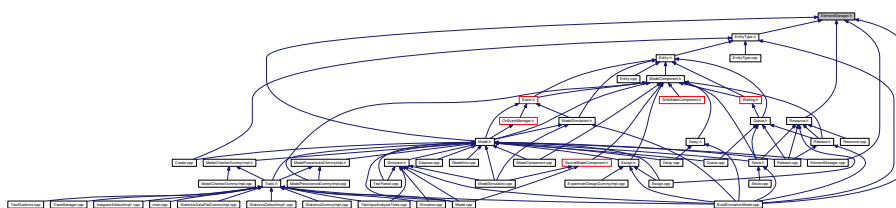




Include dependency graph for ElementManager.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [ElementManager](#)

## 6.27 ElementManager\_if.h File Reference

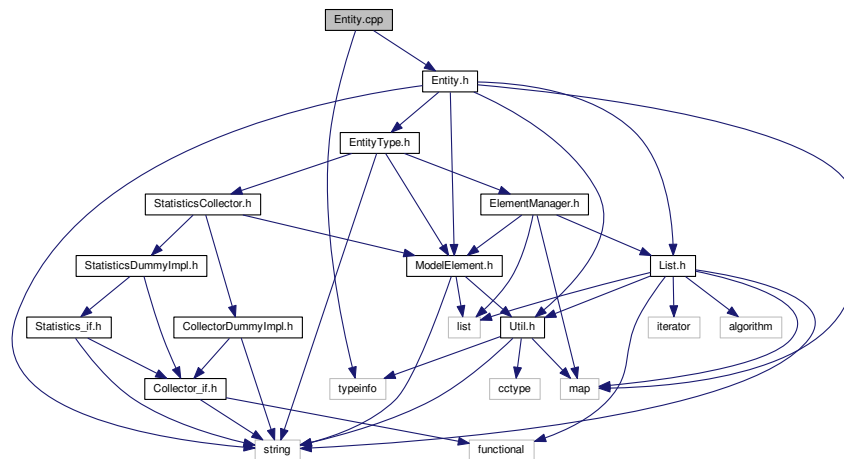
## Classes

- class [ElementManager\\_if](#)

## 6.28 Entity.cpp File Reference

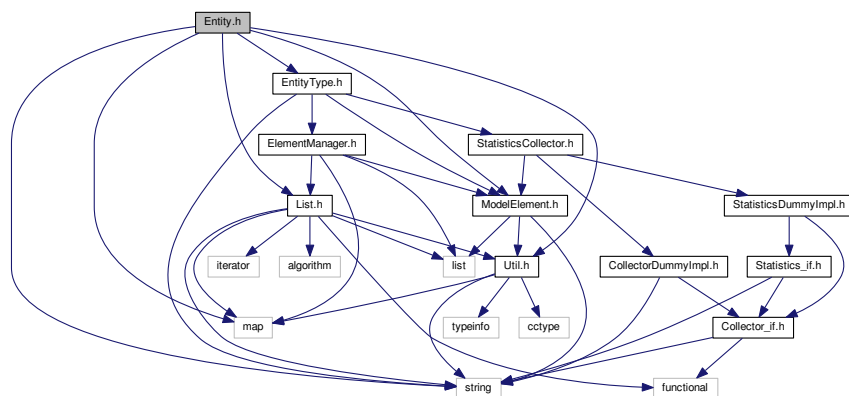
```
#include <typeinfo>
#include "Entity.h"
```

Include dependency graph for Entity.cpp:

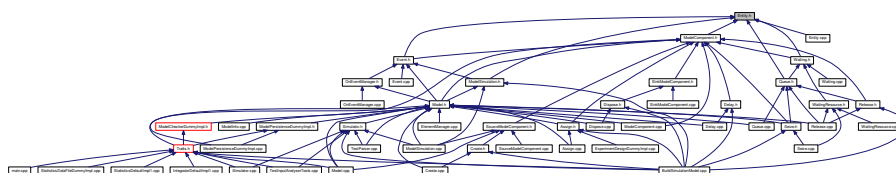


## 6.29 Entity.h File Reference

```
#include <string>
#include <map>
#include "Util.h"
#include "List.h"
#include "ModelElement.h"
#include "EntityType.h"
Include dependency graph for Entity.h:
```



This graph shows which files directly or indirectly include this file:



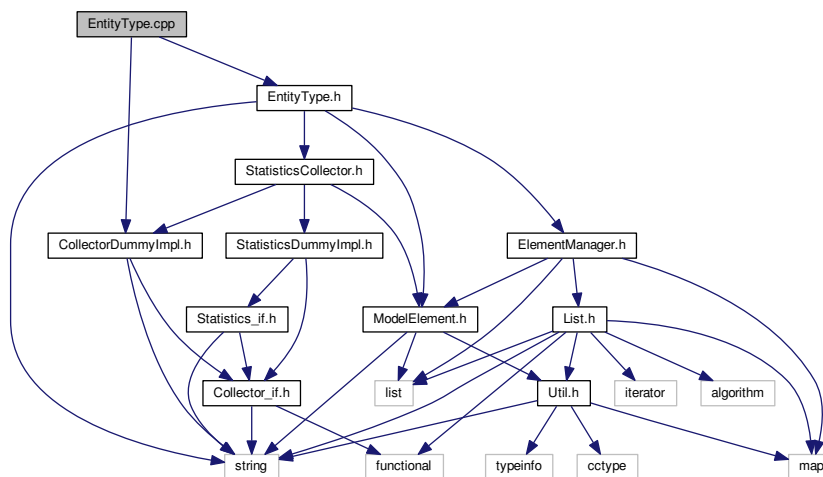


## Classes

- class [Entity](#)

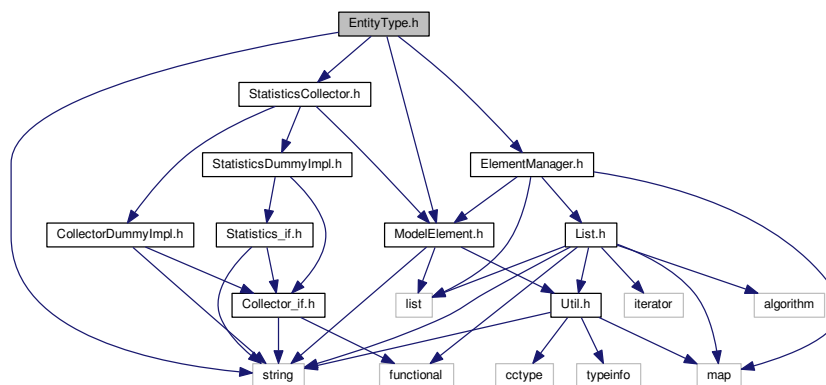
## 6.30 EntityType.cpp File Reference

```
#include "EntityType.h"
#include "CollectorDummyImpl.h"
Include dependency graph for EntityType.cpp:
```

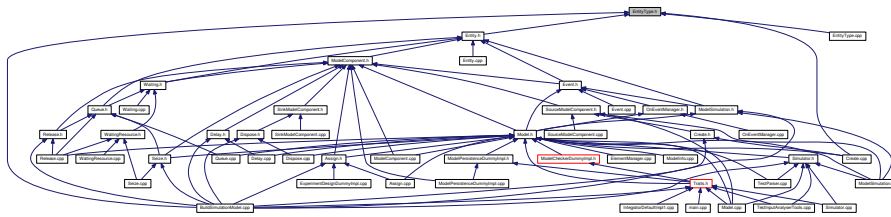


## 6.31 EntityType.h File Reference

```
#include <string>
#include "ModelElement.h"
#include "StatisticsCollector.h"
#include "ElementManager.h"
Include dependency graph for EntityType.h:
```



This graph shows which files directly or indirectly include this file:



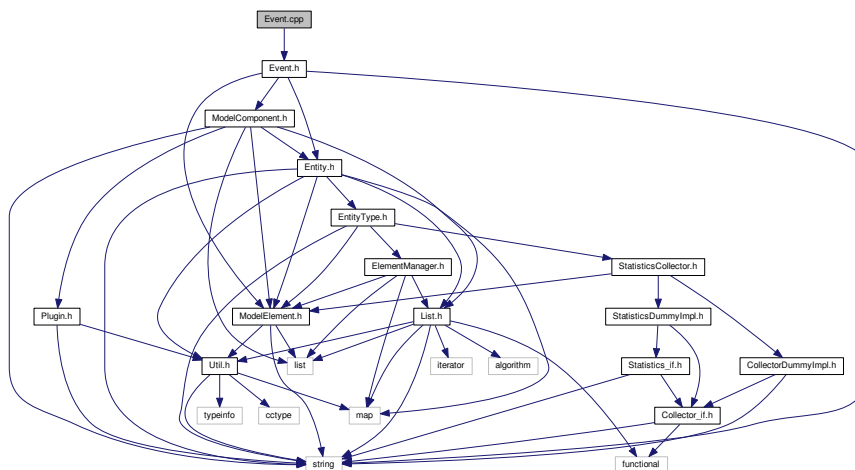
## Classes

- class [EntityType](#)

## 6.32 Event.cpp File Reference

```
#include "Event.h"
```

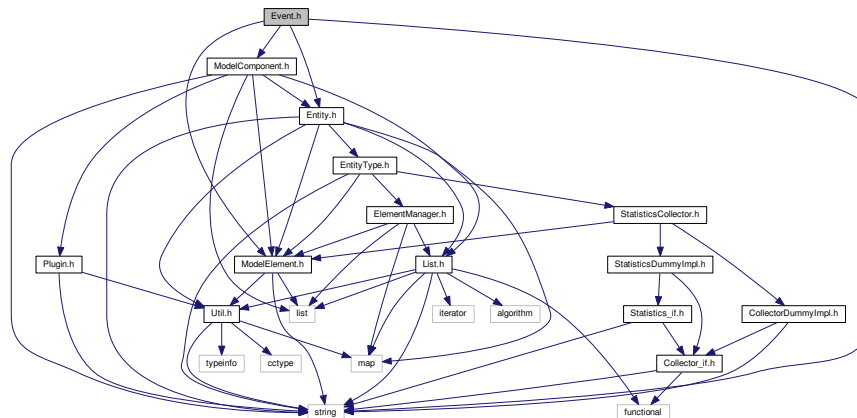
Include dependency graph for Event.cpp:



## 6.33 Event.h File Reference

```
#include <string>
#include "ModelElement.h"
#include "Entity.h"
#include "ModelComponent.h"
```

Include dependency graph for Event.h:



This graph shows which files directly or indirectly include this file:



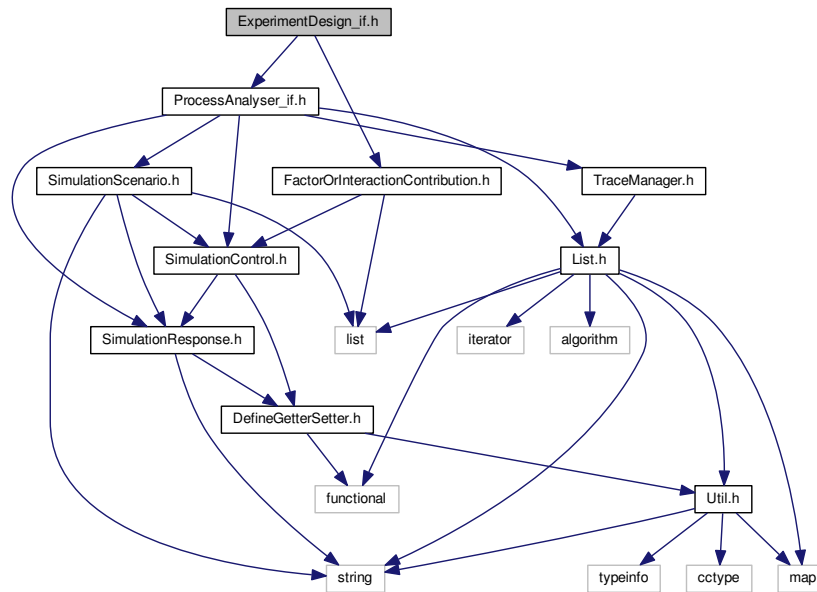
## Classes

- class [Event](#)

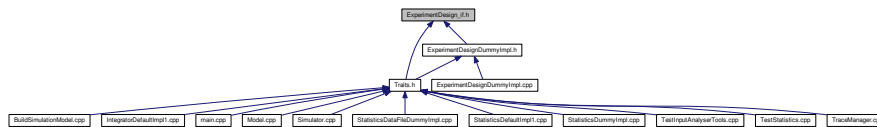
## 6.34 ExperimentDesign\_if.h File Reference

```
#include "FactorOrInteractionContribution.h"
#include "ProcessAnalyser_if.h"
```

Include dependency graph for ExperimentDesign\_if.h:



This graph shows which files directly or indirectly include this file:



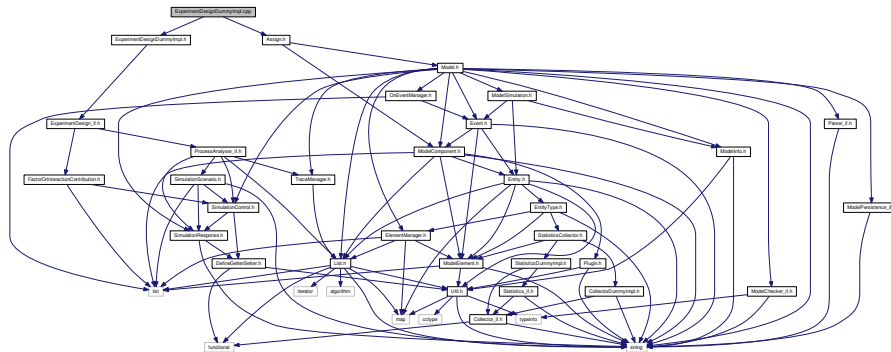
## Classes

- class [ExperimentDesign\\_if](#)

## 6.35 ExperimentDesignDummyImpl.cpp File Reference

```
#include "ExperimentDesignDummyImpl.h"
#include "Assign.h"
```

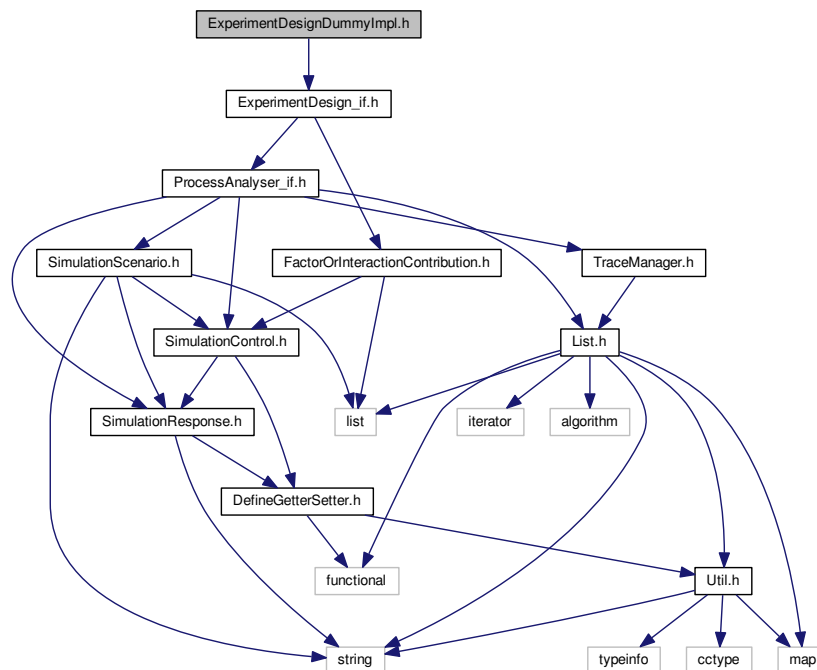
Include dependency graph for ExperimentDesignDummyImpl.cpp:



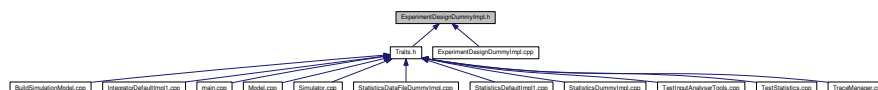
## 6.36 ExperimentDesignDummyImpl.h File Reference

```
#include "ExperimentDesign_if.h"
```

Include dependency graph for ExperimentDesignDummyImpl.h:



This graph shows which files directly or indirectly include this file:



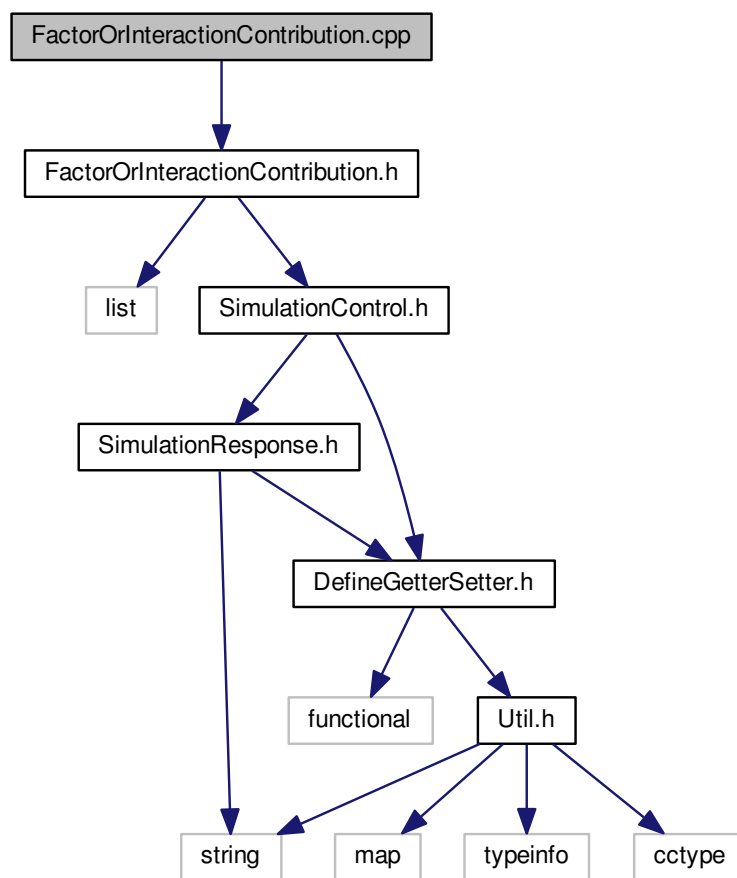
## Classes

- class [ExperimentDesignDummyImpl](#)

## 6.37 FactorOrInteractionContribution.cpp File Reference

```
#include "FactorOrInteractionContribution.h"
```

Include dependency graph for FactorOrInteractionContribution.cpp:

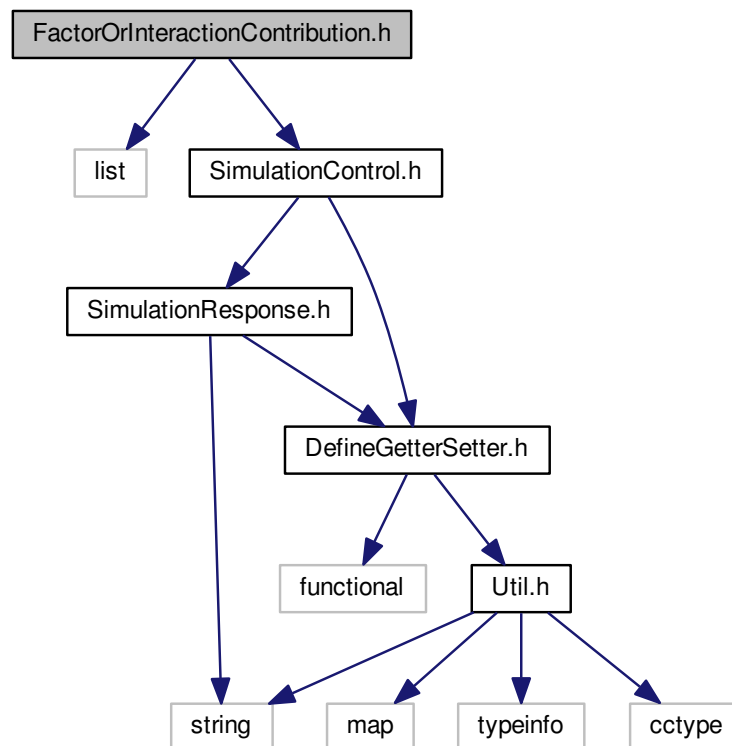


## 6.38 FactorOrInteractionContribution.h File Reference

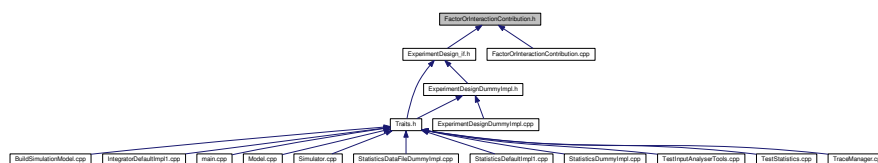
```
#include <list>
```

```
#include "SimulationControl.h"
```

Include dependency graph for FactorOrInteractionContribution.h:



This graph shows which files directly or indirectly include this file:



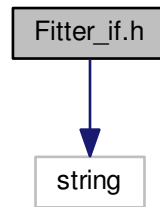
## Classes

- class [FactorOrInteractionContribution](#)

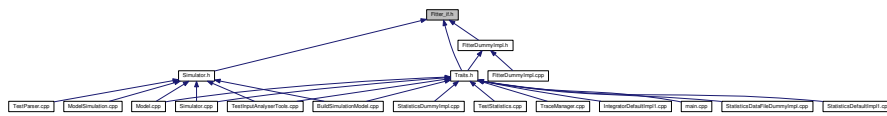
## 6.39 Fitter\_if.h File Reference

```
#include <string>
```

Include dependency graph for Fitter\_if.h:



This graph shows which files directly or indirectly include this file:



## Classes

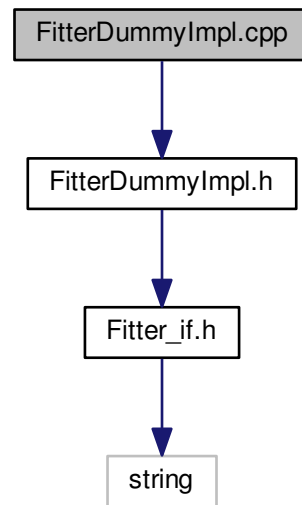
- class [Fitter\\_if](#)

## 6.40 FitterDummyImpl.cpp File Reference

```
#include "FitterDummyImpl.h"
```



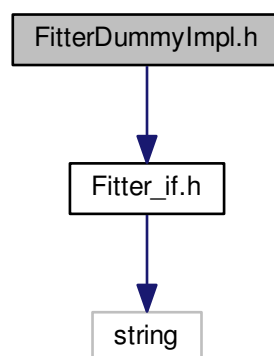
Include dependency graph for FitterDummyImpl.cpp:



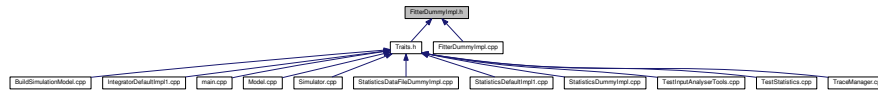
## 6.41 FitterDummyImpl.h File Reference

```
#include "Fitter_if.h"
```

Include dependency graph for FitterDummyImpl.h:



This graph shows which files directly or indirectly include this file:

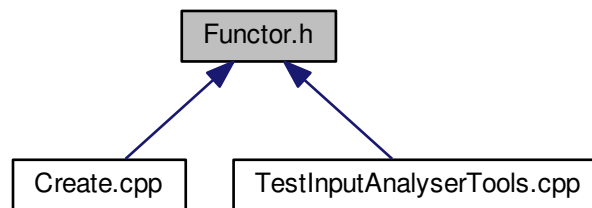


## Classes

- class [FitterDummyImpl](#)

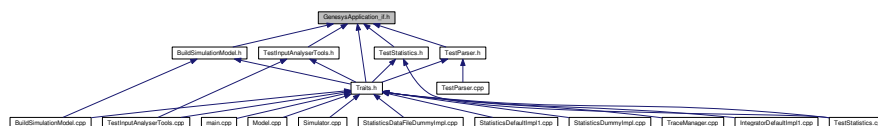
## 6.42 Functor.h File Reference

This graph shows which files directly or indirectly include this file:



## 6.43 GenesysApplication\_if.h File Reference

This graph shows which files directly or indirectly include this file:



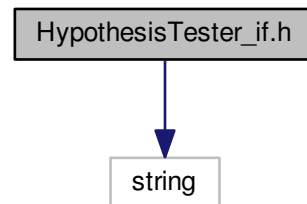
## Classes

- class [GenesysApplication\\_if](#)

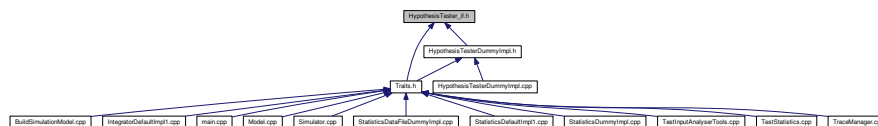
## 6.44 HypothesisTester\_if.h File Reference

```
#include <string>
```

Include dependency graph for HypothesisTester\_if.h:



This graph shows which files directly or indirectly include this file:



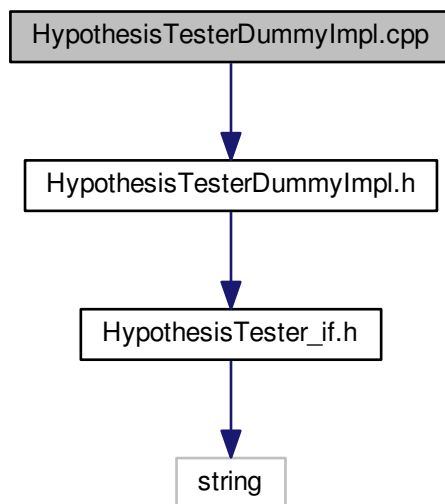
### Classes

- class [HypothesisTester\\_if](#)

## 6.45 HypothesisTesterDummyImpl.cpp File Reference

```
#include "HypothesisTesterDummyImpl.h"
```

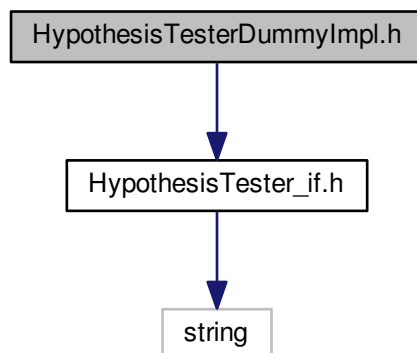
Include dependency graph for HypothesisTesterDummyImpl.cpp:



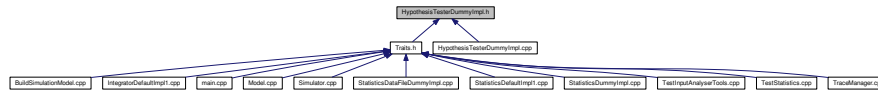
## 6.46 HypothesisTesterDummyImpl.h File Reference

```
#include "HypothesisTester_if.h"
```

Include dependency graph for HypothesisTesterDummyImpl.h:



This graph shows which files directly or indirectly include this file:

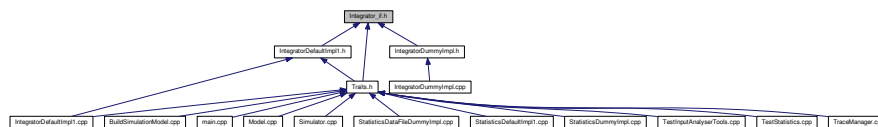


## Classes

- class [HypothesisTesterDummyImpl](#)

## 6.47 Integrator\_if.h File Reference

This graph shows which files directly or indirectly include this file:



## Classes

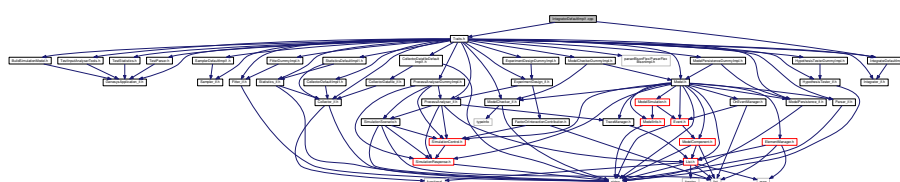
- class [Integrator\\_if](#)

## 6.48 IntegratorDefaultImpl1.cpp File Reference

```
#include "IntegratorDefaultImpl1.h"
```

```
#include "Traits.h"
```

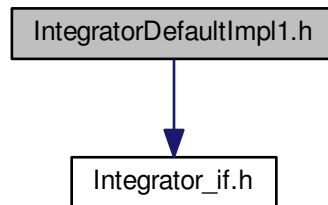
Include dependency graph for IntegratorDefaultImpl1.cpp:



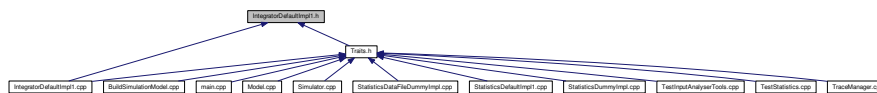
## 6.49 IntegratorDefaultImpl1.h File Reference

```
#include "Integrator_if.h"
```

Include dependency graph for IntegratorDefaultImpl1.h:



This graph shows which files directly or indirectly include this file:



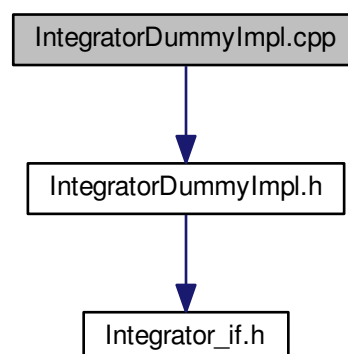
### Classes

- class [IntegratorDefaultImpl1](#)

## 6.50 IntegratorDummyImpl.cpp File Reference

```
#include "IntegratorDummyImpl.h"
```

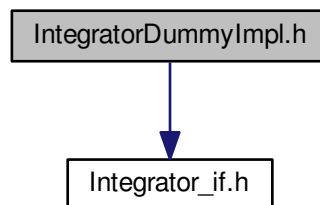
Include dependency graph for IntegratorDummyImpl.cpp:



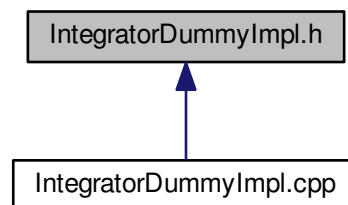
## 6.51 IntegratorDummyImpl.h File Reference

```
#include "Integrator_if.h"
```

Include dependency graph for IntegratorDummyImpl.h:



This graph shows which files directly or indirectly include this file:



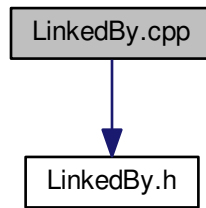
### Classes

- class [IntegratorDummyImpl](#)

## 6.52 LinkedBy.cpp File Reference

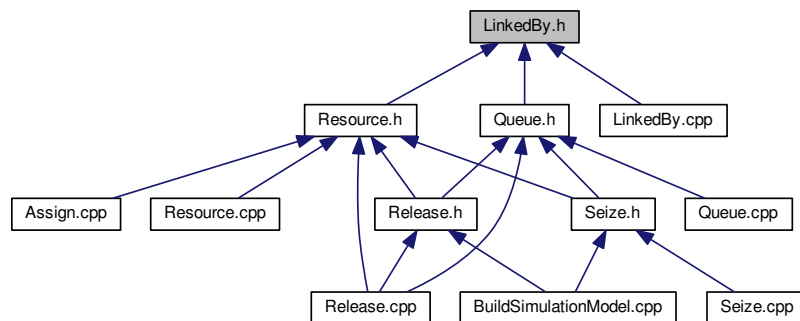
```
#include "LinkedBy.h"
```

Include dependency graph for `LinkedBy.cpp`:



### 6.53 LinkedBy.h File Reference

This graph shows which files directly or indirectly include this file:



#### Classes

- class [LinkedBy](#)

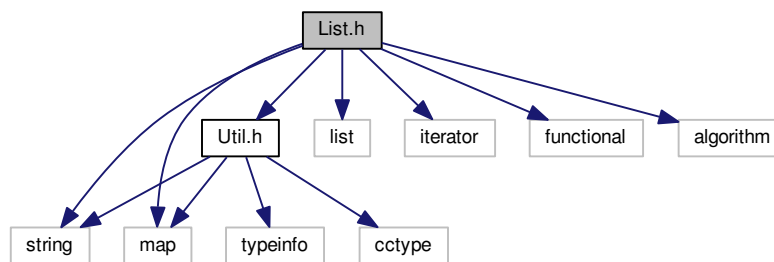
### 6.54 List.h File Reference

```

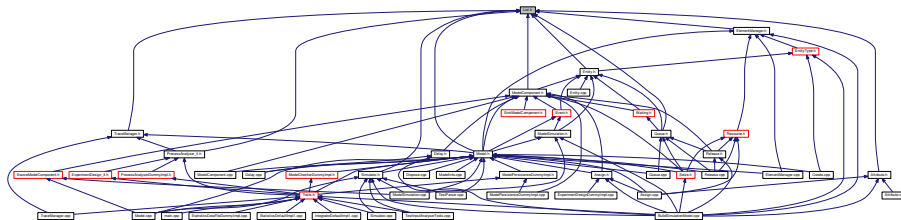
#include <string>
#include <list>
#include <map>
#include <iterator>
#include <functional>
#include <algorithm>
#include "Util.h"
  
```



Include dependency graph for List.h:



This graph shows which files directly or indirectly include this file:



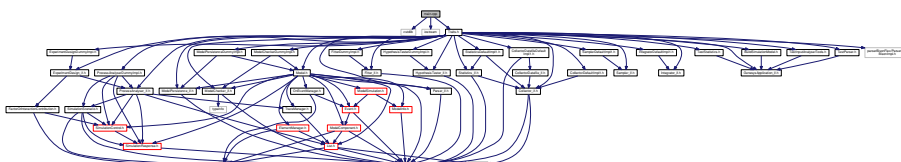
## Classes

- class [List< T >](#)

## 6.55 main.cpp File Reference

```
#include <cstdlib>
#include <iostream>
#include "Traits.h"
```

Include dependency graph for main.cpp:



## Functions

- int [main](#) (int argc, char \*\*argv)

## 6.55.1 Function Documentation

### 6.55.1.1 `int main ( int argc, char ** argv )`

Here is the call graph for this function:



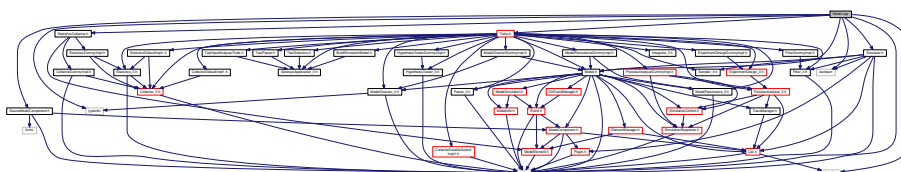
## 6.56 Model.cpp File Reference

```

#include <typeinfo>
#include <iostream>
#include <algorithm>
#include <string>
#include "Model.h"
#include "SourceModelComponent.h"
#include "Simulator.h"
#include "StatisticsCollector.h"
#include "Traits.h"

```

Include dependency graph for Model.cpp:



## Functions

- `bool EventCompare (const Event *a, const Event *b)`
- `double getReplicationLengthNotMemberFunction ()`
- `void setReplicationLengthNotMemberFunction (double value)`

## 6.56.1 Function Documentation

## 6.56.1.1 bool EventCompare ( const Event \* a, const Event \* b )

Here is the call graph for this function:



## 6.56.1.2 double getReplicationLengthNotMemberFunction ( )

## 6.56.1.3 void setReplicationLengthNotMemberFunction ( double value )

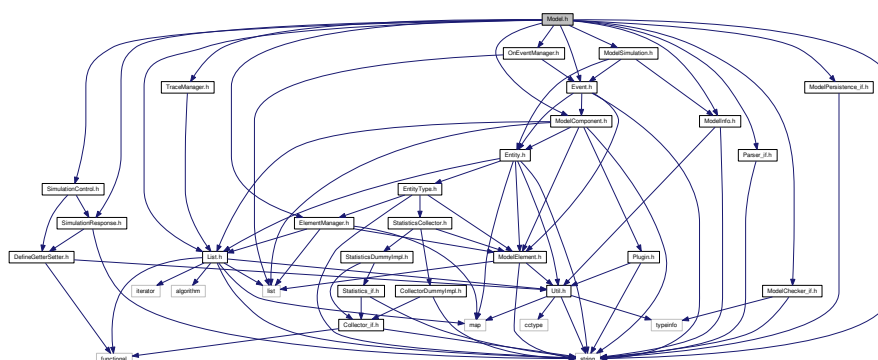
## 6.57 Model.h File Reference

```

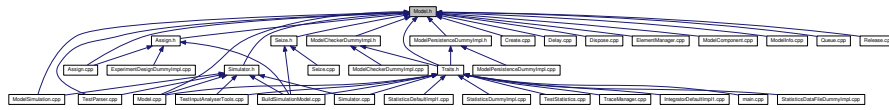
#include <string>
#include "List.h"
#include "ModelComponent.h"
#include "Event.h"
#include "ModelChecker_if.h"
#include "Parser_if.h"
#include "ModelPersistence_if.h"
#include "EventManager.h"
#include "TraceManager.h"
#include "OnEventManager.h"
#include "ModelInfo.h"
#include "ModelSimulation.h"
#include "SimulationResponse.h"
#include "SimulationControl.h"

```

Include dependency graph for Model.h:



This graph shows which files directly or indirectly include this file:

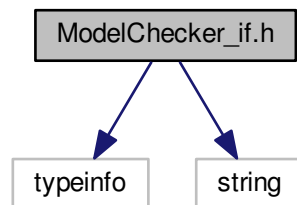


## Classes

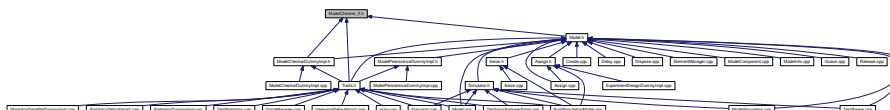
- class [Model](#)

## 6.58 ModelChecker\_if.h File Reference

```
#include <typeinfo>
#include <string>
Include dependency graph for ModelChecker_if.h:
```



This graph shows which files directly or indirectly include this file:

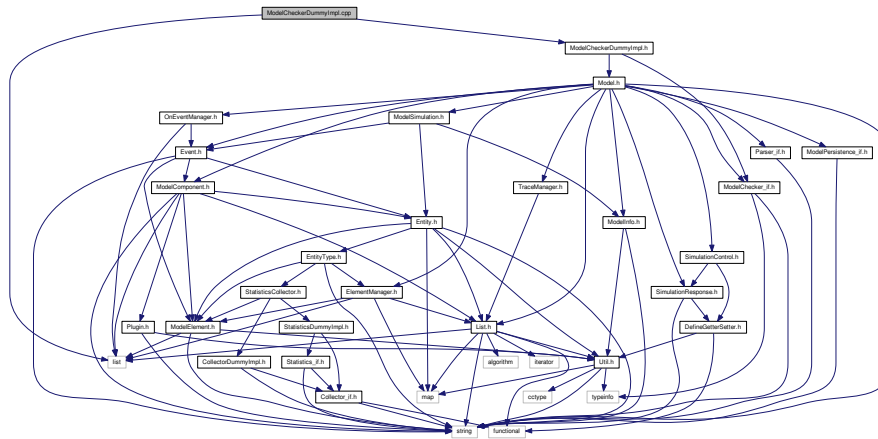


## Classes

- class [ModelChecker\\_if](#)

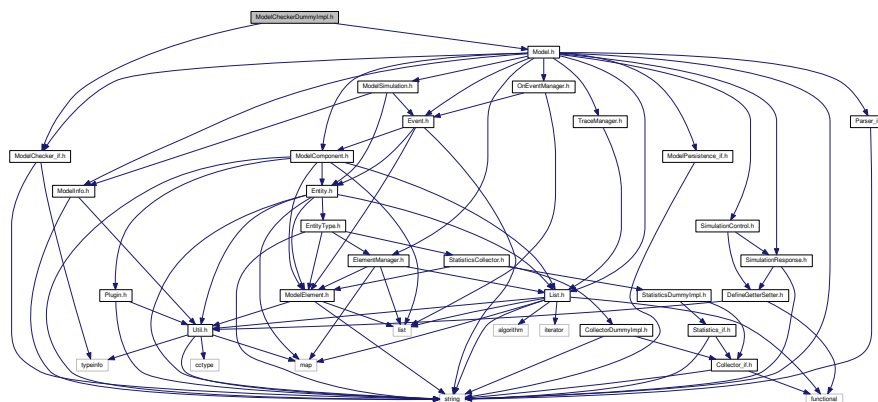
## 6.59 ModelCheckerDummyImpl.cpp File Reference

```
#include <list>
#include "ModelCheckerDummyImpl.h"
Include dependency graph for ModelCheckerDummyImpl.cpp:
```

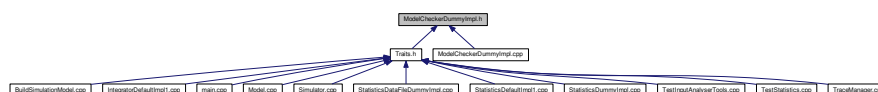


## 6.60 ModelCheckerDummyImpl.h File Reference

```
#include "ModelChecker_if.h"
#include "Model.h"
Include dependency graph for ModelCheckerDummyImpl.h:
```

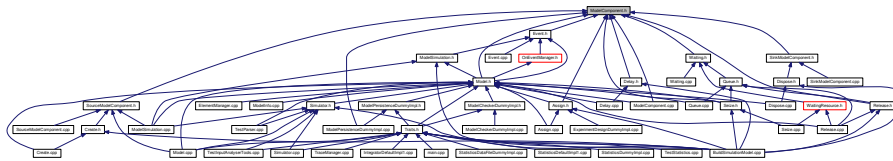


This graph shows which files directly or indirectly include this file:





This graph shows which files directly or indirectly include this file:



## Classes

- class [ModelComponent](#)

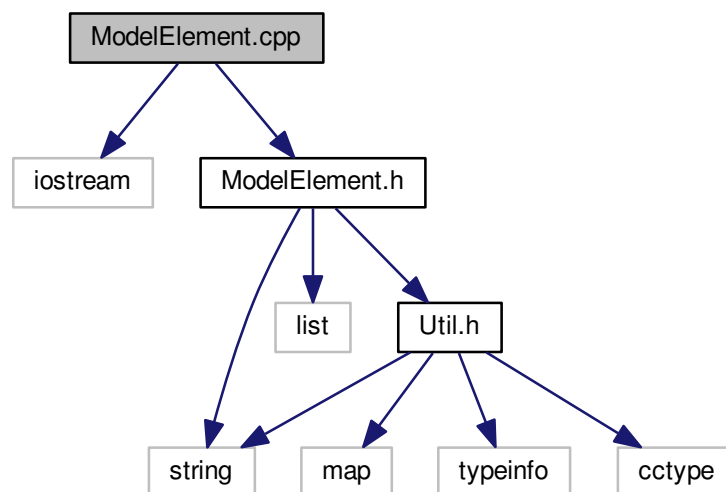
## 6.63 ModelComponentManager\_if.h File Reference

## Classes

- class [ModelComponentManager\\_if](#)

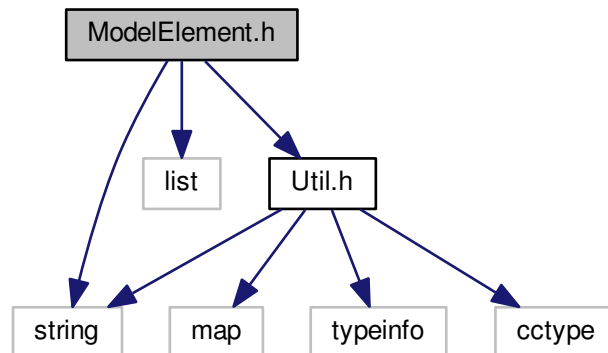
## 6.64 ModelElement.cpp File Reference

```
#include <iostream>
#include "ModelElement.h"
Include dependency graph for ModelElement.cpp:
```

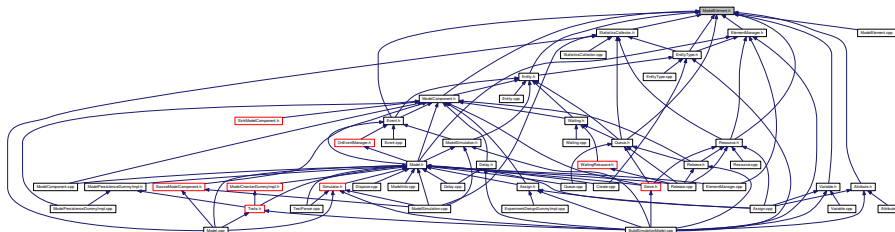


## 6.65 ModelElement.h File Reference

```
#include <string>
#include <list>
#include "Util.h"
Include dependency graph for ModelElement.h:
```



This graph shows which files directly or indirectly include this file:



### Classes

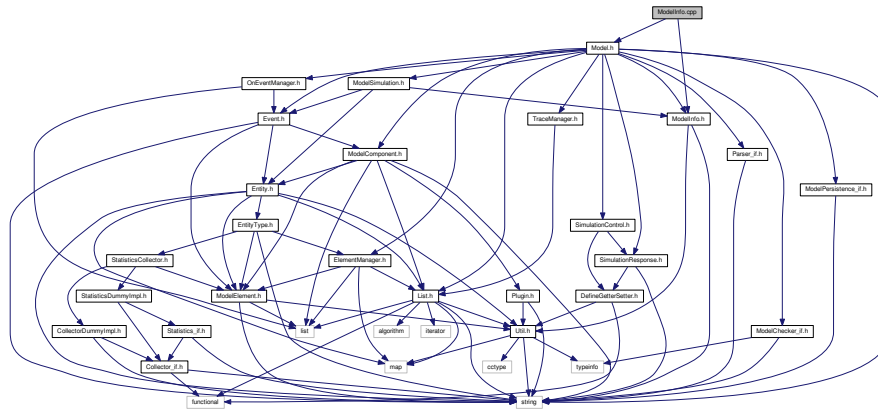
- class [ModelElement](#)

## 6.66 ModelInfo.cpp File Reference

```
#include "ModelInfo.h"
#include "Model.h"
```



Include dependency graph for ModelInfo.cpp:

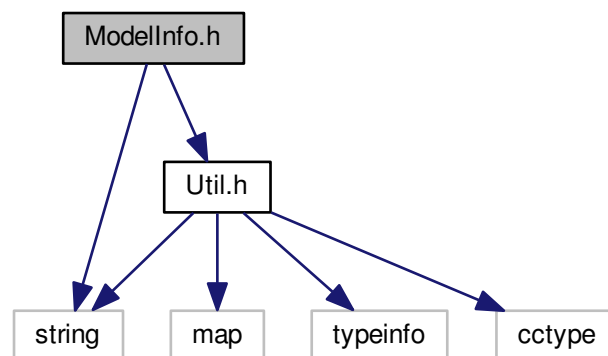


## 6.67 ModellInfo.h File Reference

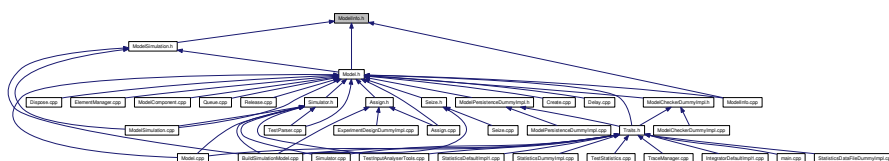
```
#include <string>
```

```
#include "Util.h"
```

Include dependency graph for ModellInfo.h:



This graph shows which files directly or indirectly include this file:



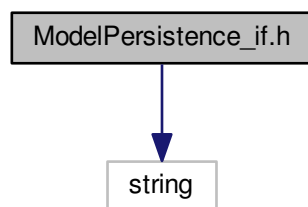
## Classes

- class [ModelInfo](#)

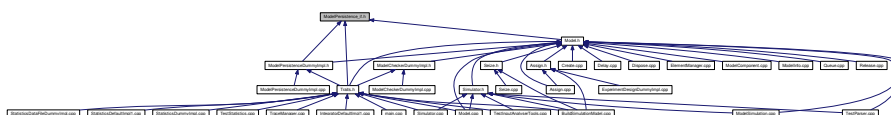
## 6.68 ModelPersistence\_if.h File Reference

```
#include <string>
```

Include dependency graph for ModelPersistence\_if.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [ModelPersistence\\_if](#)

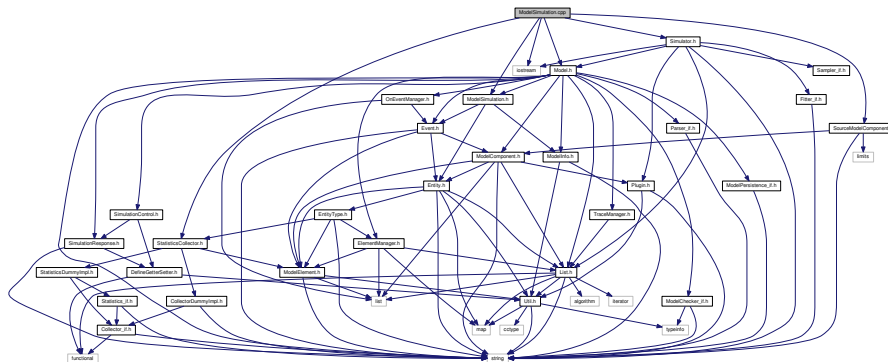
## 6.69 ModelPersistenceDummyImpl.cpp File Reference

```
#include <iostream>
#include "ModelPersistenceDummyImpl.h"
#include "ModelComponent.h"
```



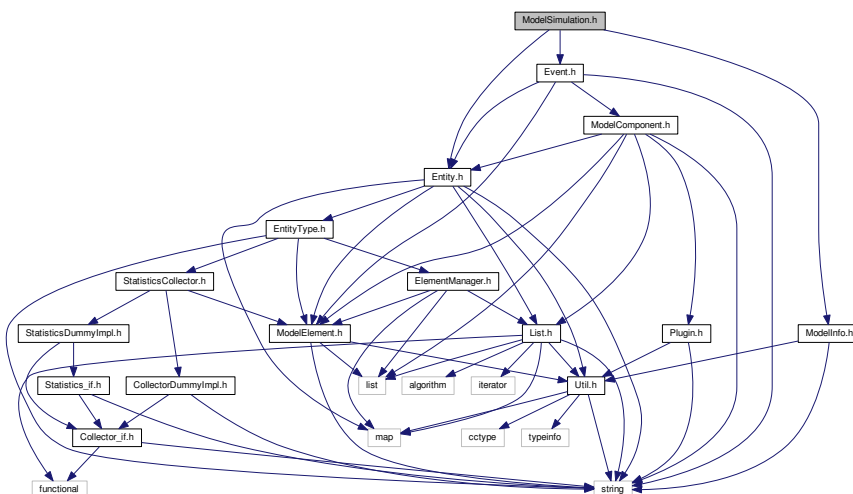
## 6.71 ModelSimulation.cpp File Reference

```
#include <iostream>
#include "ModelSimulation.h"
#include "Model.h"
#include "Simulator.h"
#include "SourceModelComponent.h"
#include "StatisticsCollector.h"
Include dependency graph for ModelSimulation.cpp:
```



## 6.72 ModelSimulation.h File Reference

```
#include "Event.h"
#include "Entity.h"
#include "ModelInfo.h"
Include dependency graph for ModelSimulation.h:
```



This graph shows which files directly or indirectly include this file:



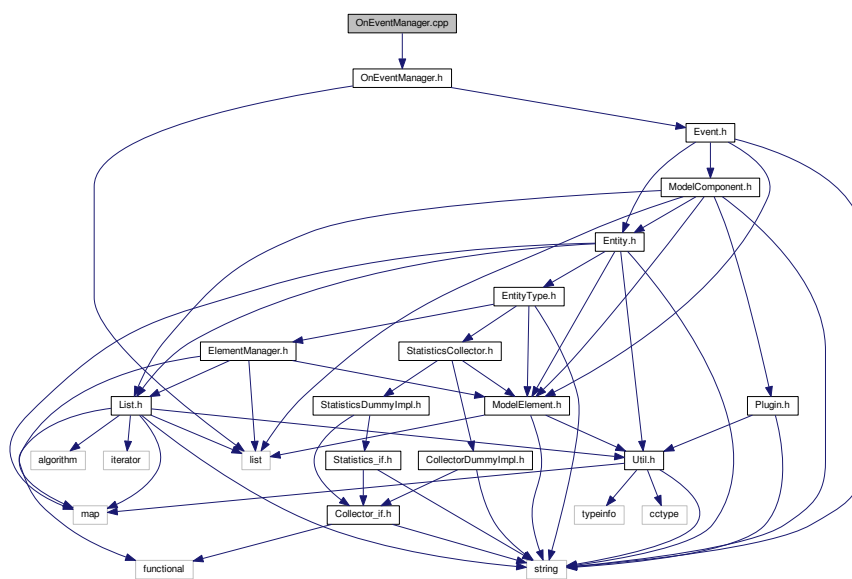
## Classes

- class [ModelSimulation](#)

## 6.73 OnEventManager.cpp File Reference

```
#include "OnEventManager.h"
```

Include dependency graph for OnEventManager.cpp:



## 6.74 OnEventManager.h File Reference

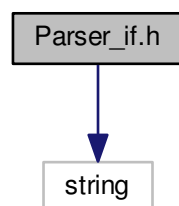
```
#include <list>
#include "Event.h"
```



## 6.75 Parser\_if.h File Reference

```
#include <string>
```

Include dependency graph for Parser\_if.h:



This graph shows which files directly or indirectly include this file:



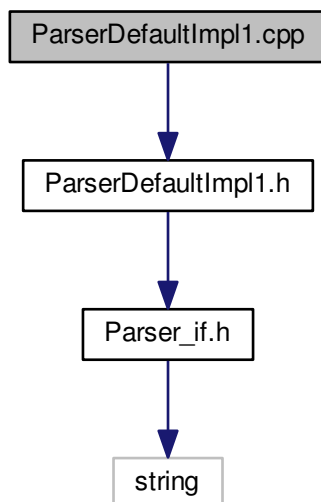
### Classes

- class [Parser\\_if](#)

## 6.76 ParserDefaultImpl1.cpp File Reference

```
#include "ParserDefaultImpl1.h"
```

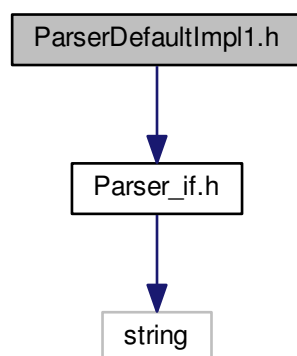
Include dependency graph for ParserDefaultImpl1.cpp:



## 6.77 ParserDefaultImpl1.h File Reference

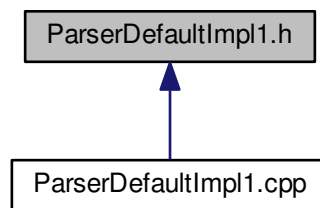
```
#include "Parser_if.h"
```

Include dependency graph for ParserDefaultImpl1.h:





This graph shows which files directly or indirectly include this file:



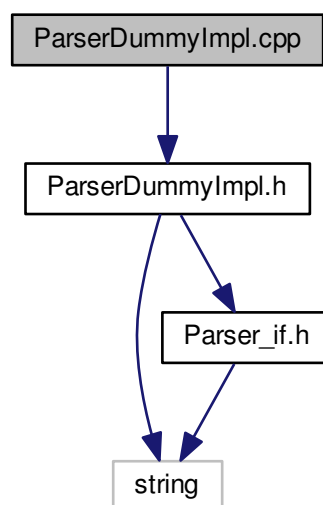
## Classes

- class [ParserDefaultImpl1](#)

## 6.78 ParserDummyImpl.cpp File Reference

```
#include "ParserDummyImpl.h"
```

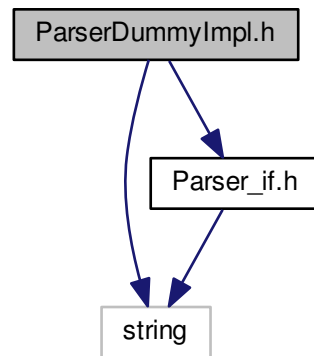
Include dependency graph for ParserDummyImpl.cpp:



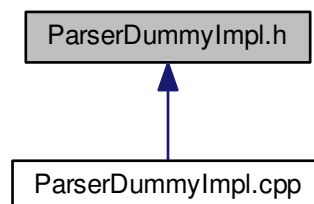
## 6.79 ParserDummyImpl.h File Reference

```
#include <string>
#include "Parser_if.h"
```

Include dependency graph for ParserDummyImpl.h:



This graph shows which files directly or indirectly include this file:



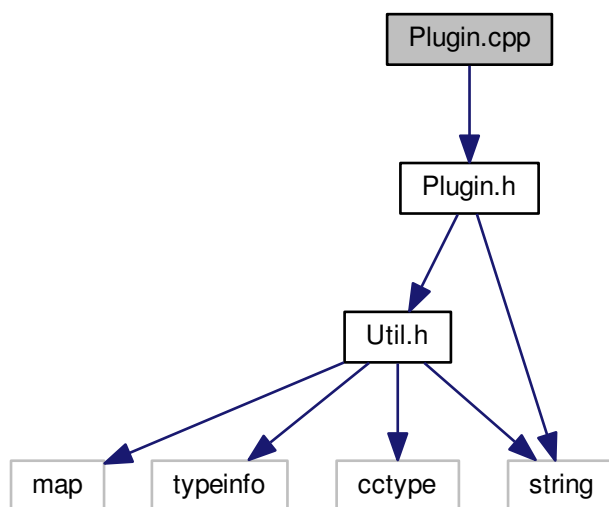
### Classes

- class [ParserDummyImpl](#)

## 6.80 Plugin.cpp File Reference

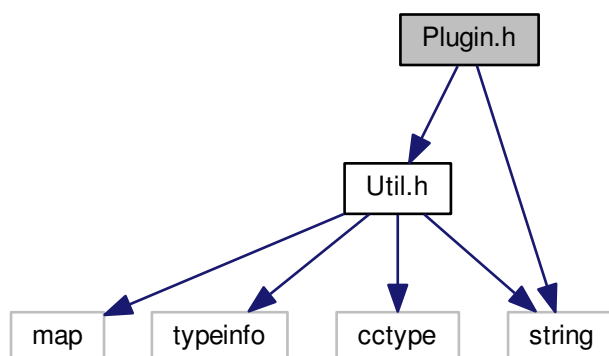
```
#include "Plugin.h"
```

Include dependency graph for Plugin.cpp:

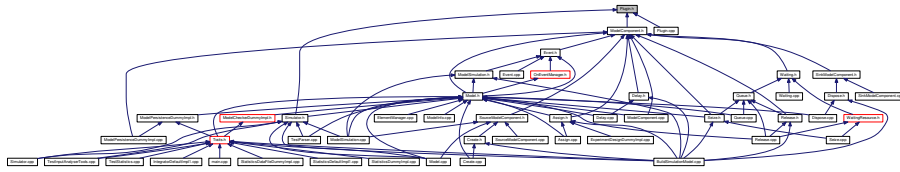


## 6.81 Plugin.h File Reference

```
#include "Util.h"
#include <string>
Include dependency graph for Plugin.h:
```



This graph shows which files directly or indirectly include this file:



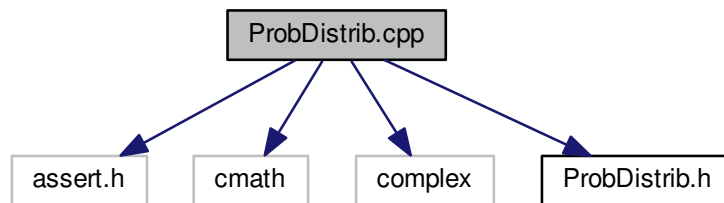
## Classes

- class [Plugin](#)

## 6.82 ProbDistrib.cpp File Reference

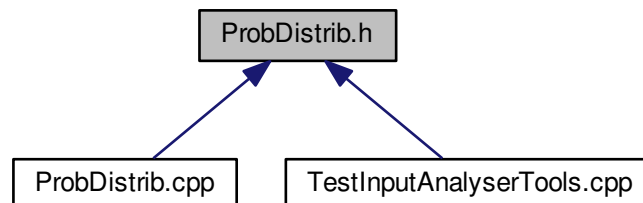
```
#include <assert.h>
#include <cmath>
#include <complex>
#include "ProbDistrib.h"
```

Include dependency graph for ProbDistrib.cpp:



## 6.83 ProbDistrib.h File Reference

This graph shows which files directly or indirectly include this file:



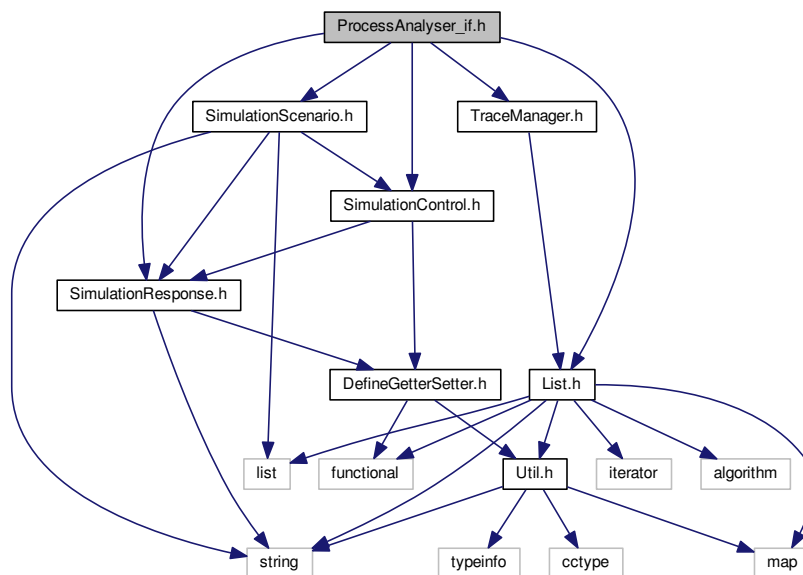
## Classes

- class **ProbDistrib**

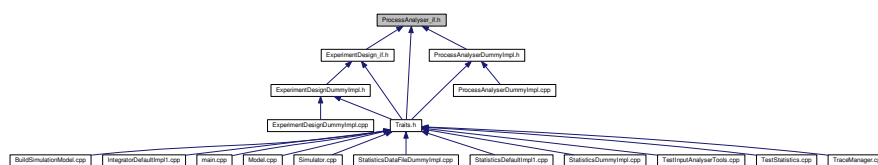
## 6.84 ProcessAnalyser\_if.h File Reference

```
#include "List.h"
#include "SimulationScenario.h"
#include "SimulationControl.h"
#include "SimulationResponse.h"
#include "TraceManager.h"
```

Include dependency graph for ProcessAnalyser\_if.h:



This graph shows which files directly or indirectly include this file:



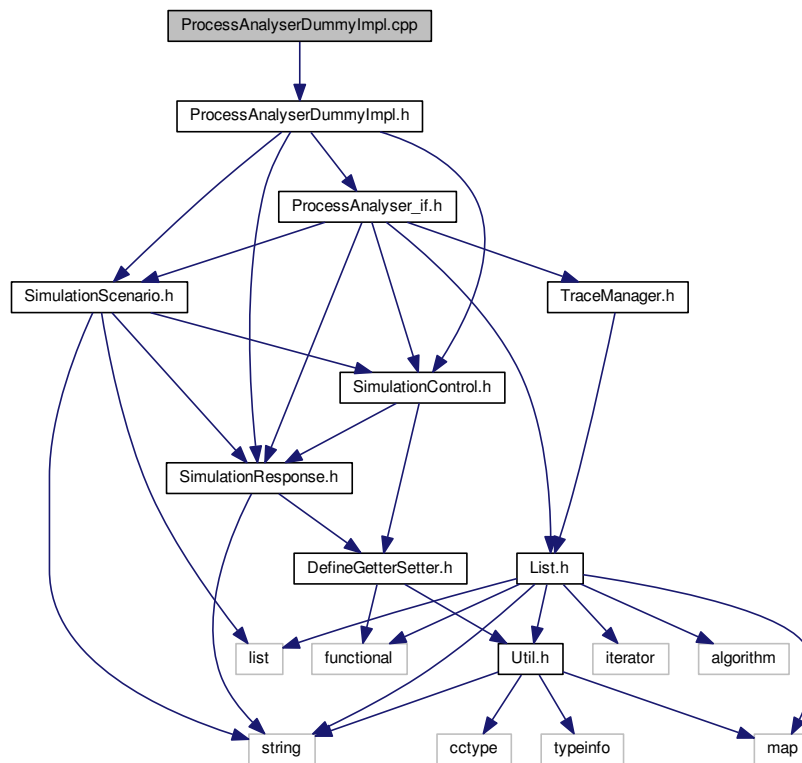
## Classes

- class `ProcessAnalyser_if`

## 6.85 ProcessAnalyserDummyImpl.cpp File Reference

```
#include "ProcessAnalyserDummyImpl.h"
```

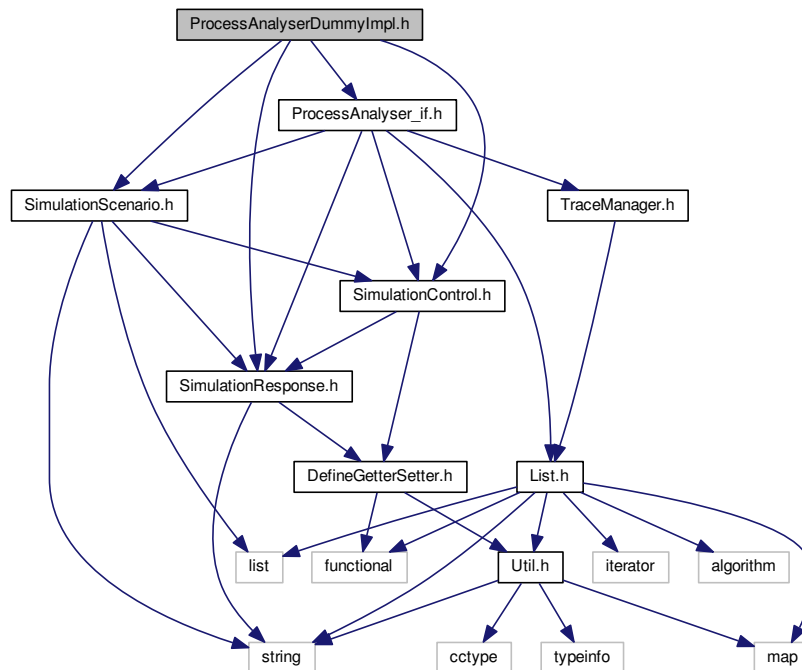
Include dependency graph for ProcessAnalyserDummyImpl.cpp:



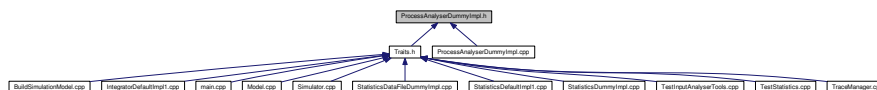
## 6.86 ProcessAnalyserDummyImpl.h File Reference

```
#include "ProcessAnalyser_if.h"
#include "SimulationScenario.h"
#include "SimulationResponse.h"
#include "SimulationControl.h"
```

Include dependency graph for ProcessAnalyserDummyImpl.h:



This graph shows which files directly or indirectly include this file:



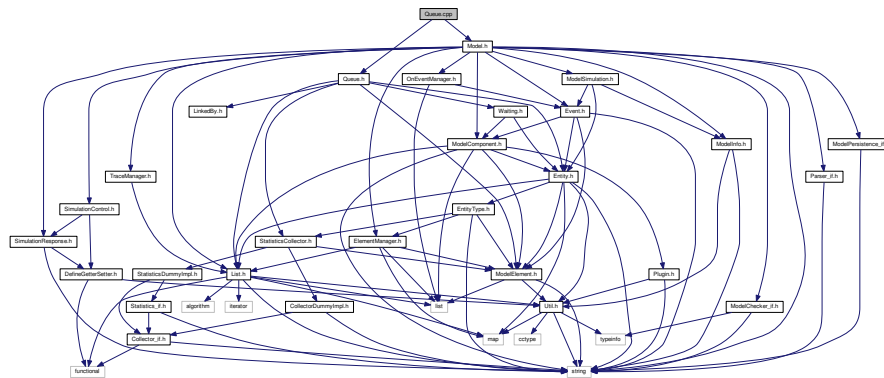
## Classes

- class [ProcessAnalyserDummyImpl](#)

## 6.87 Queue.cpp File Reference

```
#include "Queue.h"
#include "Model.h"
```

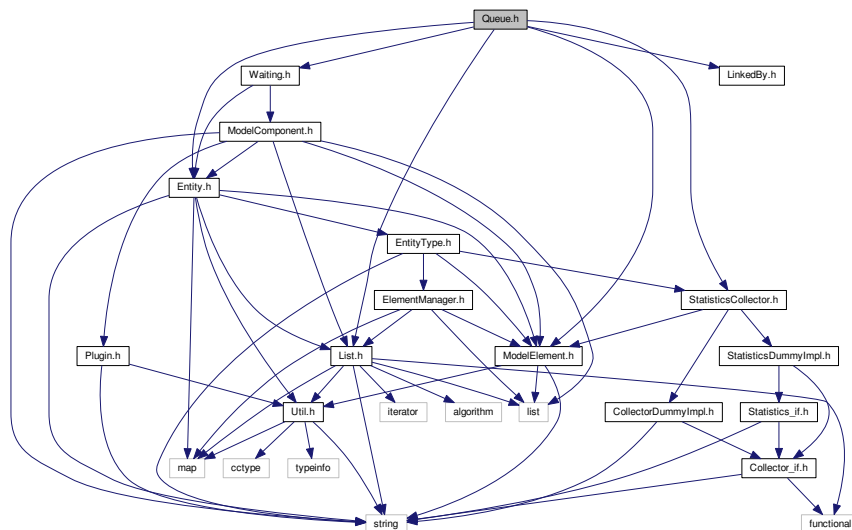
Include dependency graph for Queue.cpp:



## 6.88 Queue.h File Reference

```
#include "ModelElement.h"
#include "LinkedBy.h"
#include "List.h"
#include "Entity.h"
#include "Waiting.h"
#include "StatisticsCollector.h"
```

Include dependency graph for Queue.h:

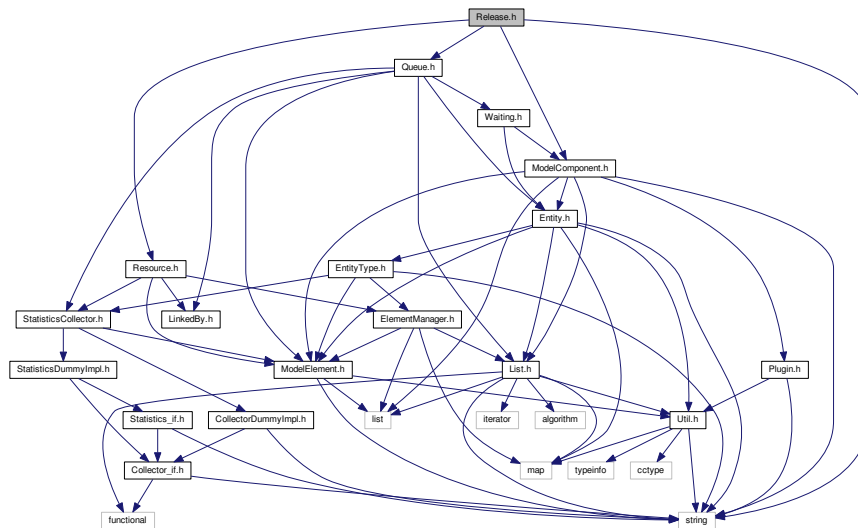




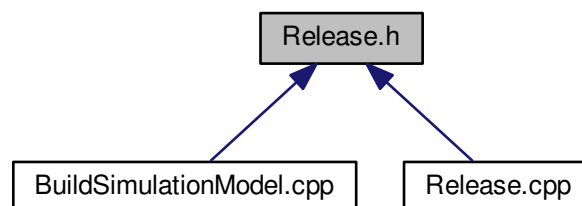


## 6.91 Release.h File Reference

```
#include <string>
#include "ModelComponent.h"
#include "Resource.h"
#include "Queue.h"
Include dependency graph for Release.h:
```



This graph shows which files directly or indirectly include this file:



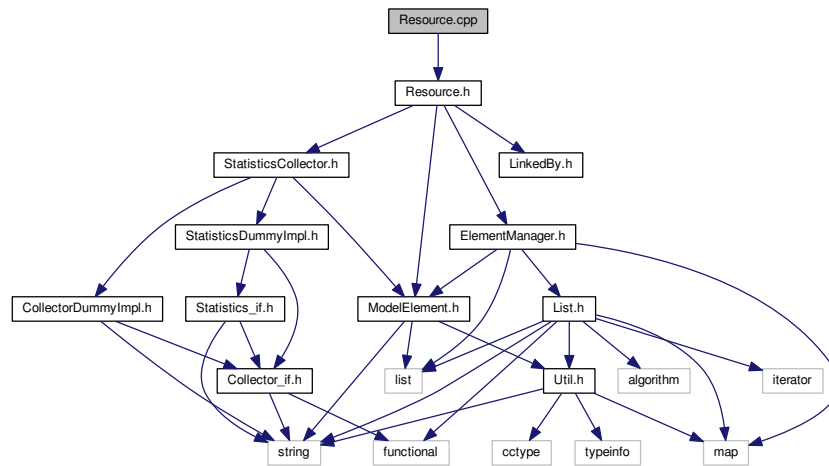
## Classes

- class [Release](#)

## 6.92 Resource.cpp File Reference

```
#include "Resource.h"
```

Include dependency graph for Resource.cpp:



## 6.93 Resource.h File Reference

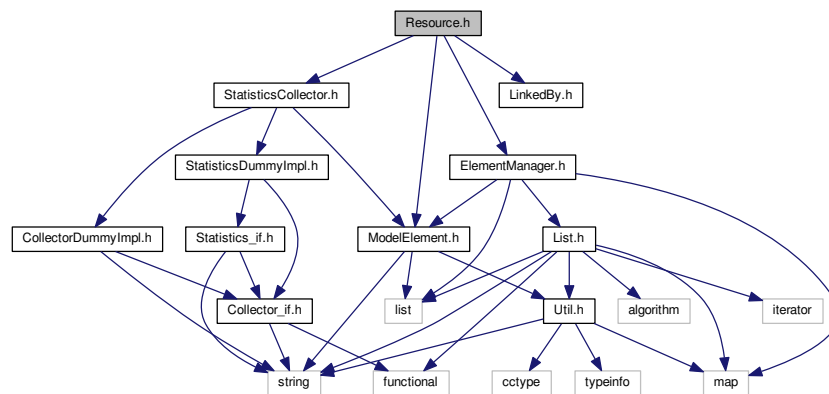
```
#include "ModelElement.h"
```

```
#include "LinkBy.h"
```

```
#include "StatisticsCollector.h"
```

```
#include "ElementManager.h"
```

Include dependency graph for Resource.h:



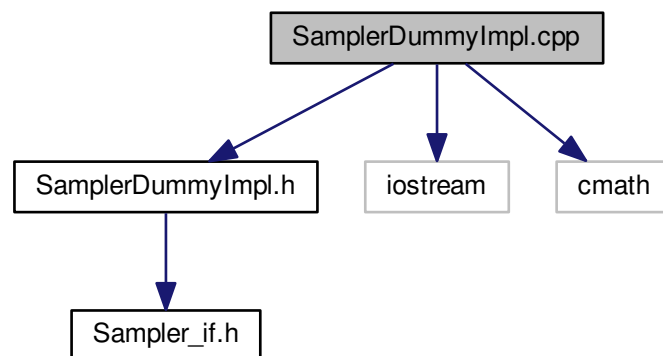




## 6.97 SamplerDummyImpl.cpp File Reference

```
#include "SamplerDummyImpl.h"  
#include <iostream>  
#include <cmath>
```

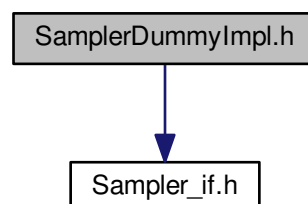
Include dependency graph for SamplerDummyImpl.cpp:



## 6.98 SamplerDummyImpl.h File Reference

```
#include "Sampler_if.h"
```

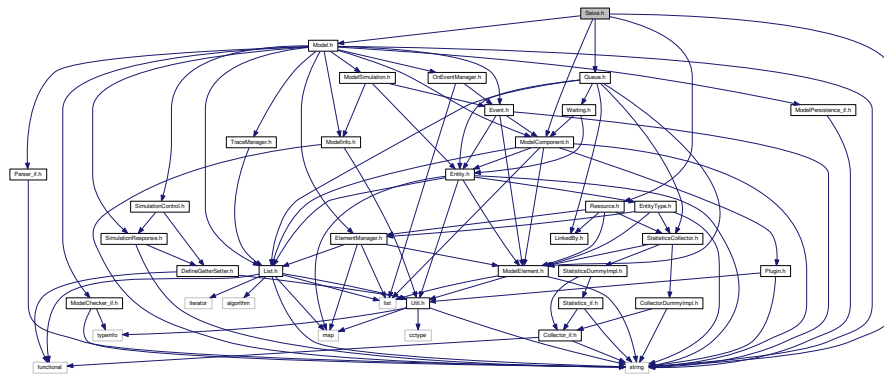
Include dependency graph for SamplerDummyImpl.h:



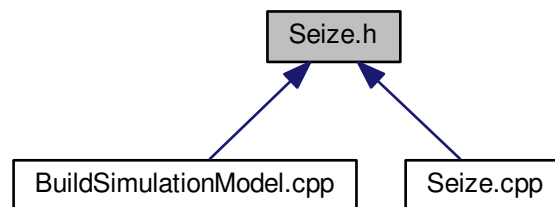


## 6.101 Seize.h File Reference

```
#include <string>
#include "ModelComponent.h"
#include "Model.h"
#include "Resource.h"
#include "Queue.h"
Include dependency graph for Seize.h:
```



This graph shows which files directly or indirectly include this file:



### Classes

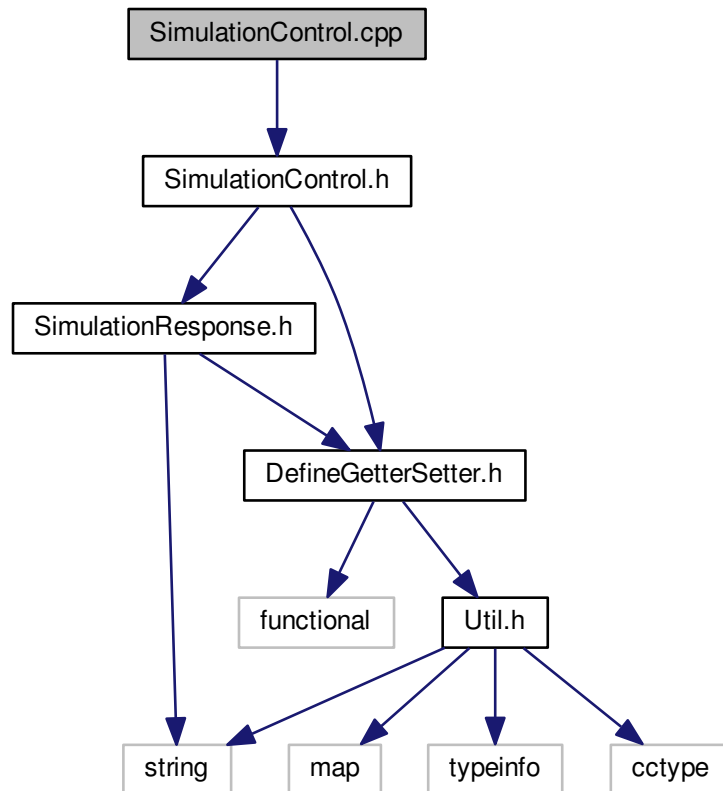
- class [Seize](#)

## 6.102 SimulationControl.cpp File Reference

```
#include "SimulationControl.h"
```



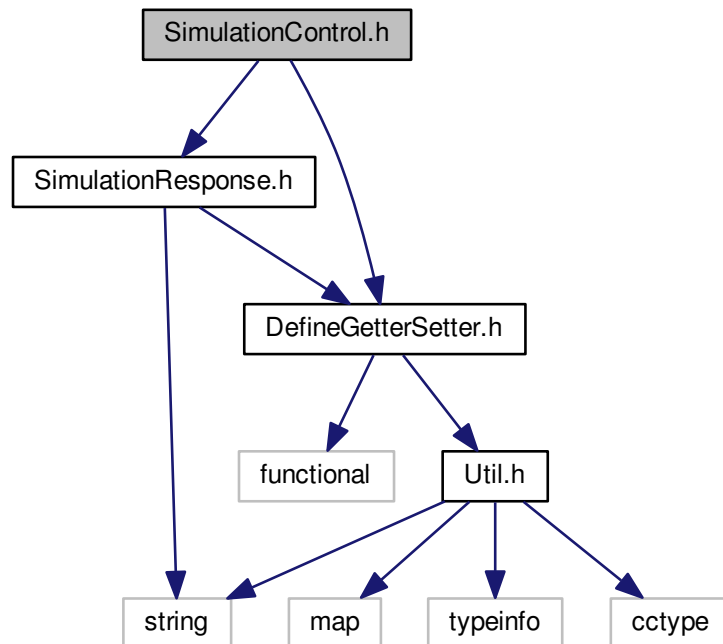
Include dependency graph for SimulationControl.cpp:



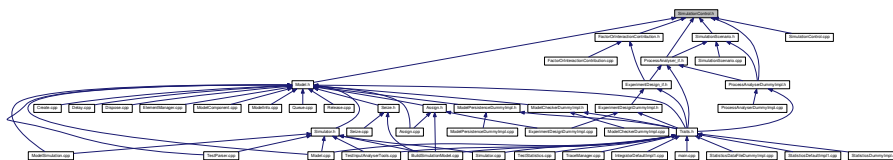
## 6.103 SimulationControl.h File Reference

```
#include "SimulationResponse.h"
#include "DefineGetterSetter.h"
```

Include dependency graph for SimulationControl.h:



This graph shows which files directly or indirectly include this file:



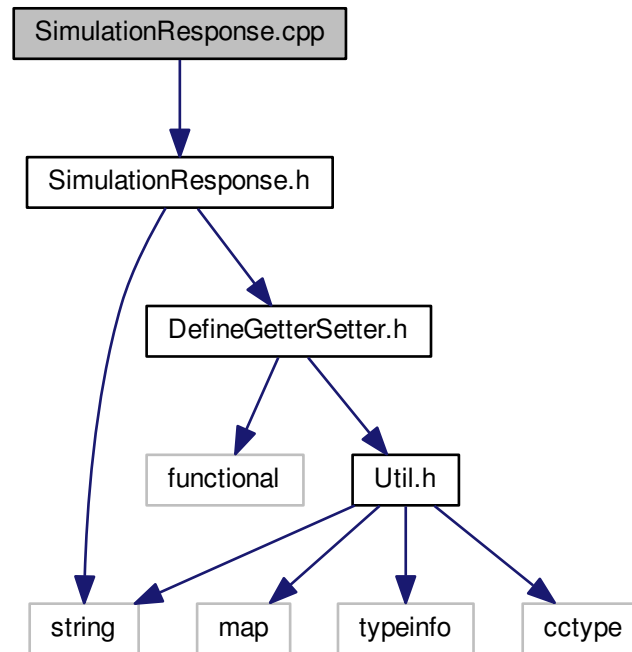
## Classes

- class [SimulationControl](#)

## 6.104 SimulationResponse.cpp File Reference

```
#include "SimulationResponse.h"
```

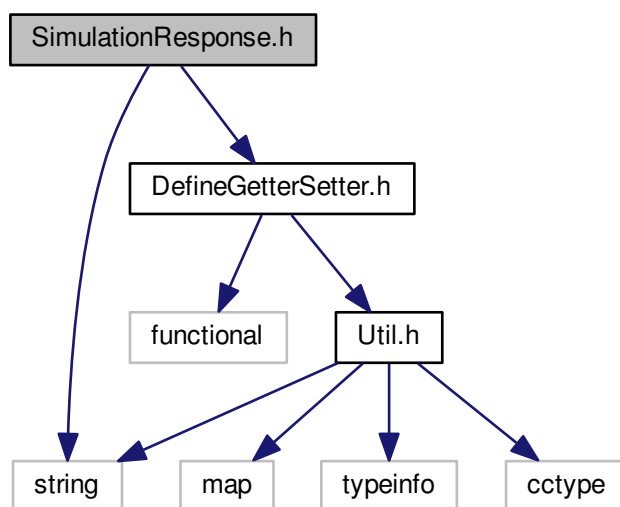
Include dependency graph for SimulationResponse.cpp:



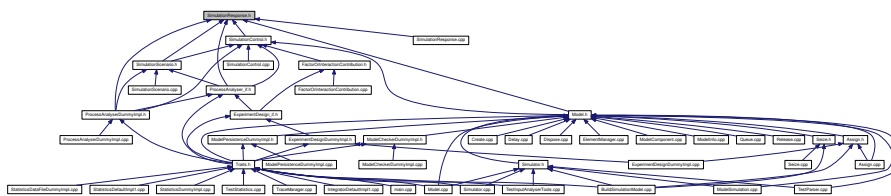
## 6.105 SimulationResponse.h File Reference

```
#include <string>
#include "DefineGetterSetter.h"
```

Include dependency graph for SimulationResponse.h:



This graph shows which files directly or indirectly include this file:



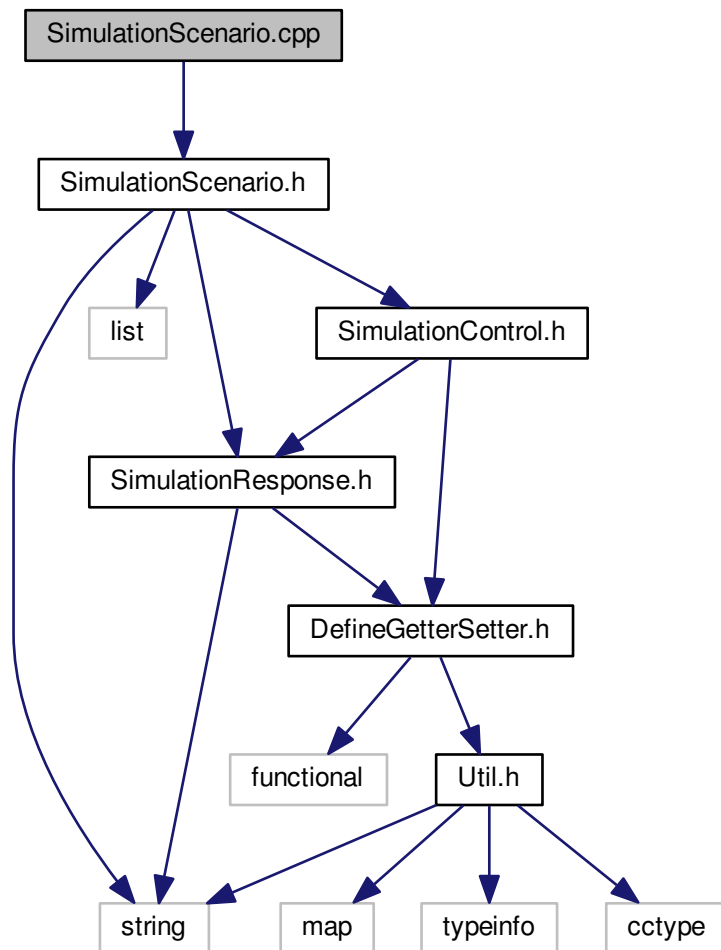
## Classes

- class [SimulationResponse](#)

## 6.106 SimulationScenario.cpp File Reference

```
#include "SimulationScenario.h"
```

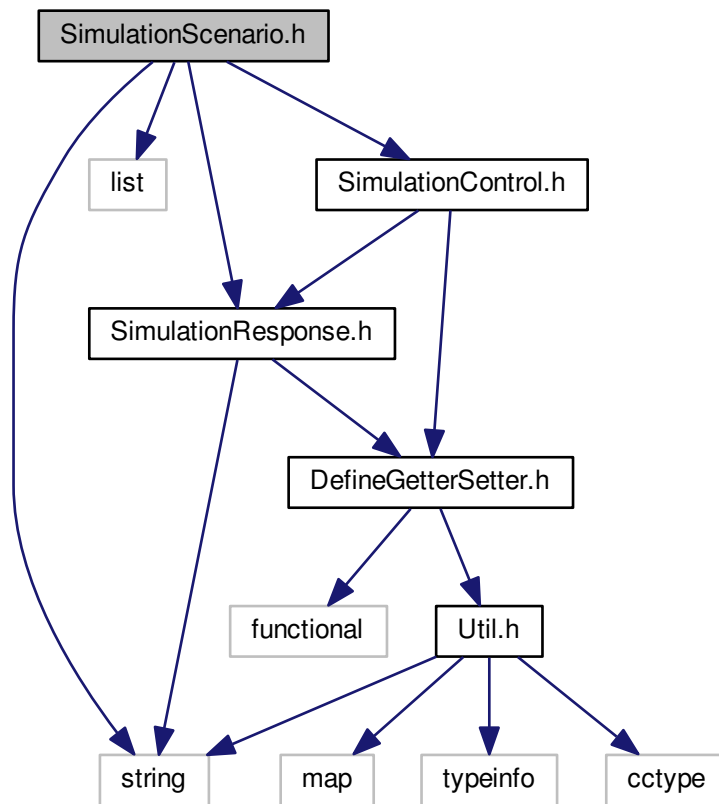
Include dependency graph for SimulationScenario.cpp:



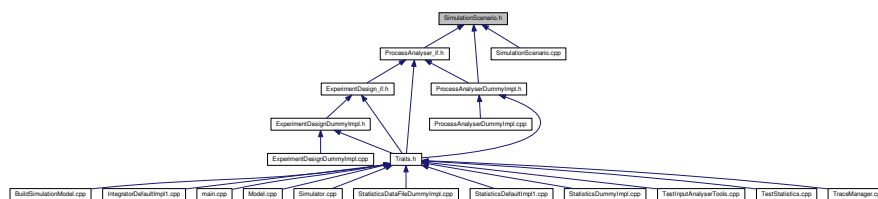
## 6.107 SimulationScenario.h File Reference

```
#include <string>
#include <list>
#include "SimulationResponse.h"
#include "SimulationControl.h"
```

Include dependency graph for SimulationScenario.h:



This graph shows which files directly or indirectly include this file:



## Classes

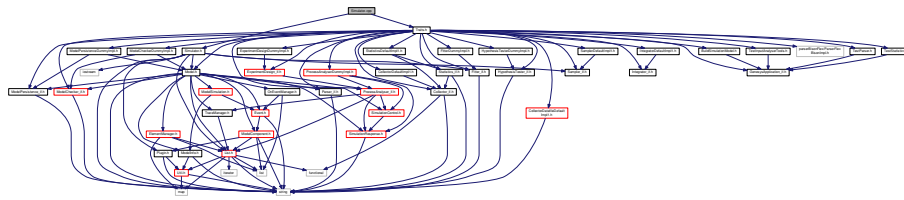
- class [SimulationScenario](#)

## 6.108 Simulator.cpp File Reference

```
#include "Simulator.h"
```

```
#include "Traits.h"
```

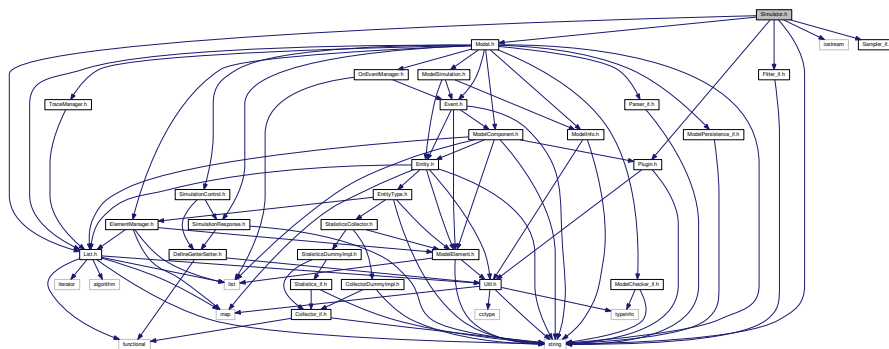
Include dependency graph for Simulator.cpp:



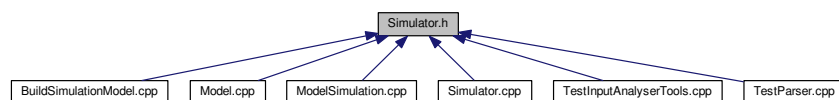
## 6.109 Simulator.h File Reference

```
#include <string>
#include <iostream>
#include "Model.h"
#include "Plugin.h"
#include "List.h"
#include "Fitter_if.h"
#include "Sampler_if.h"
```

Include dependency graph for Simulator.h:



This graph shows which files directly or indirectly include this file:



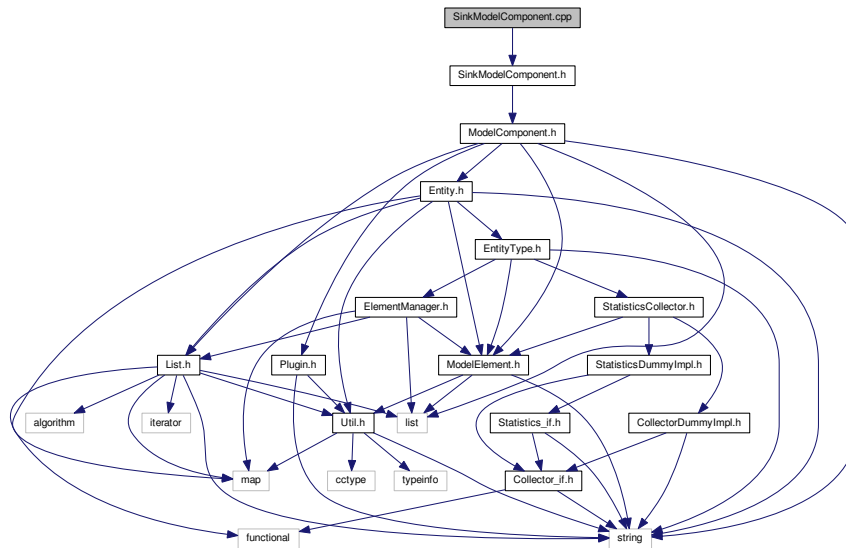
## Classes

- class [Simulator](#)

## 6.110 SinkModelComponent.cpp File Reference

```
#include "SinkModelComponent.h"
```

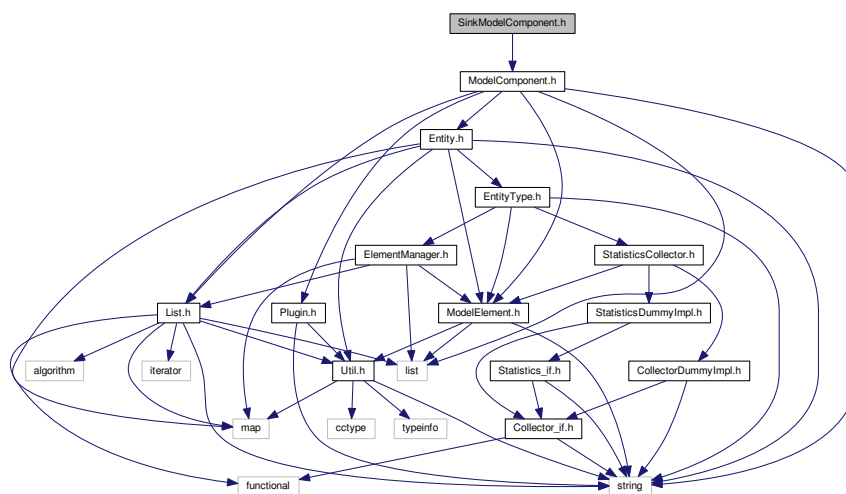
Include dependency graph for SinkModelComponent.cpp:



## 6.111 SinkModelComponent.h File Reference

```
#include "ModelComponent.h"
```

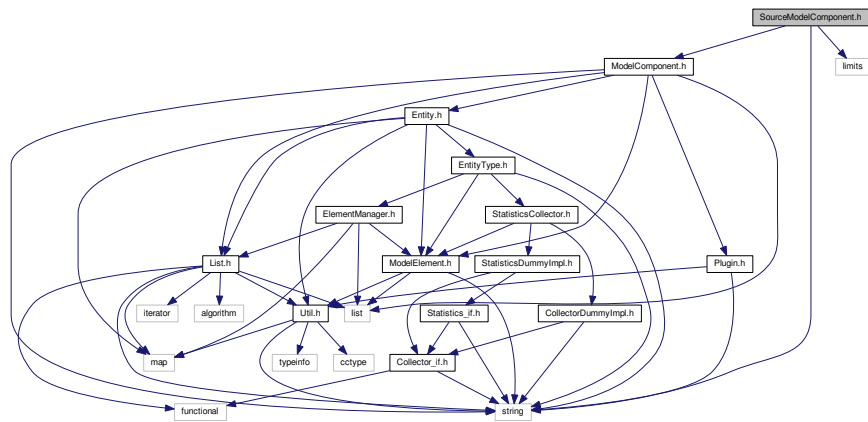
Include dependency graph for SinkModelComponent.h:



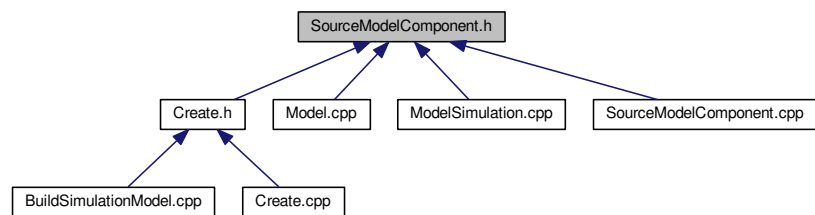




```
#include <string>
#include <limits>
Include dependency graph for SourceModelComponent.h:
```



This graph shows which files directly or indirectly include this file:



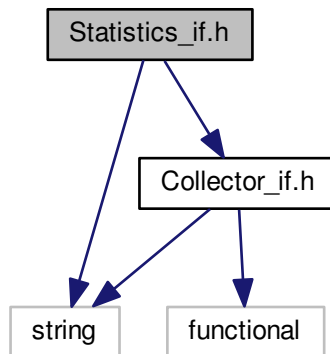
## Classes

- class [SourceModelComponent](#)

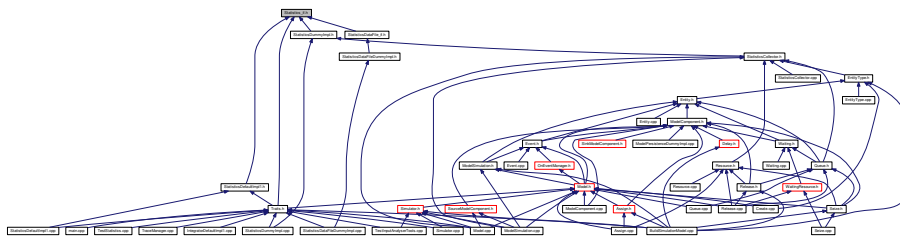
## 6.114 Statistics\_if.h File Reference

```
#include <string>
#include "Collector_if.h"
```

Include dependency graph for Statistics\_if.h:



This graph shows which files directly or indirectly include this file:



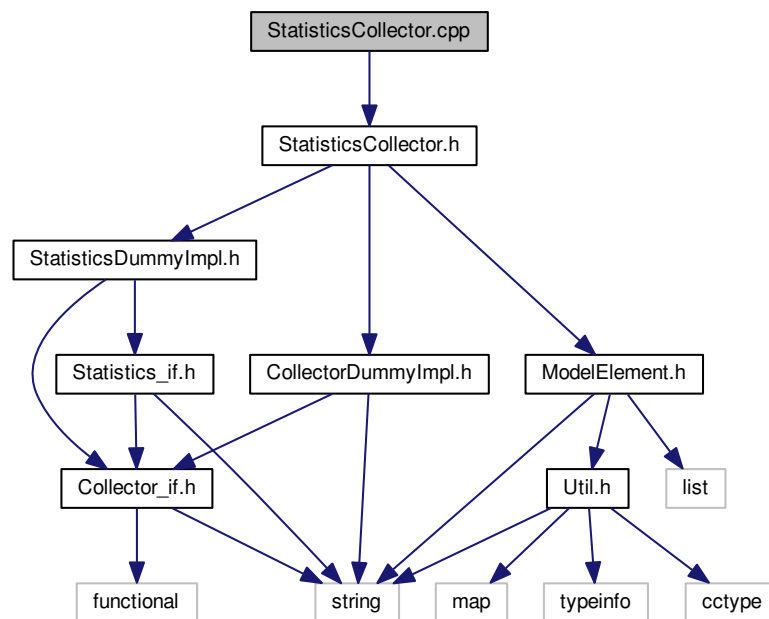
## Classes

- class [Statistics\\_if](#)

## 6.115 StatisticsCollector.cpp File Reference

```
#include "StatisticsCollector.h"
```

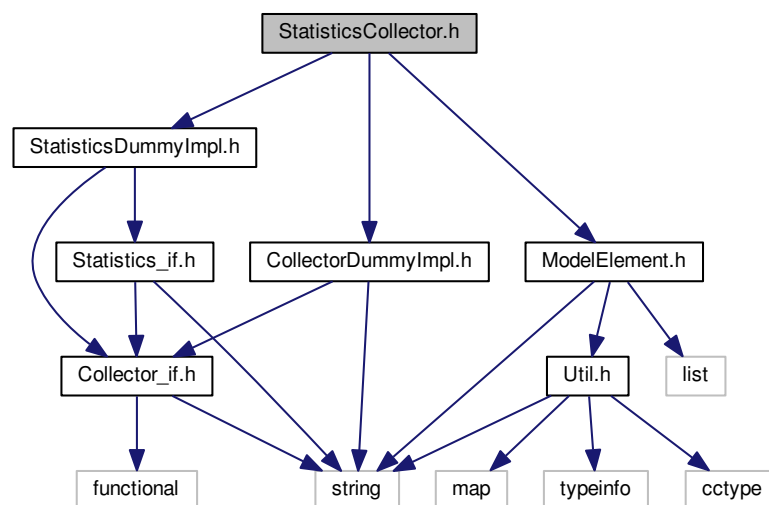
Include dependency graph for StatisticsCollector.cpp:



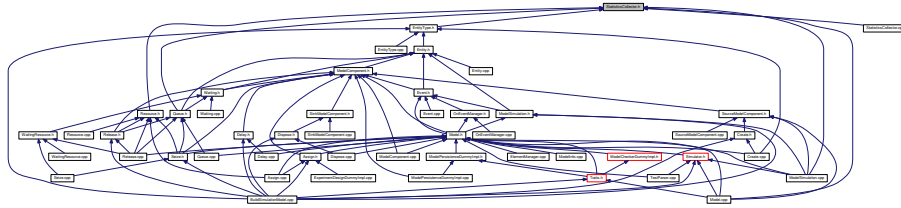
## 6.116 StatisticsCollector.h File Reference

```
#include "StatisticsDummyImpl.h"
#include "CollectorDummyImpl.h"
#include "ModelElement.h"
```

Include dependency graph for StatisticsCollector.h:



This graph shows which files directly or indirectly include this file:



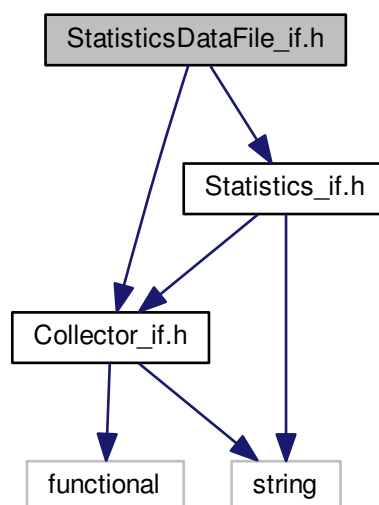
## Classes

- class [StatisticsCollector](#)

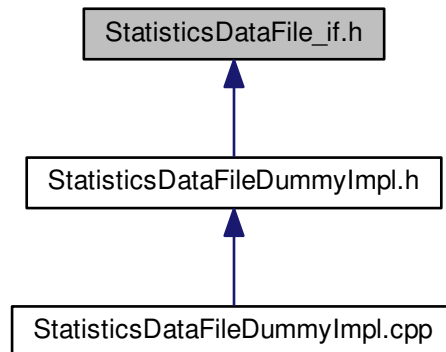
## 6.117 StatisticsDataFile\_if.h File Reference

```
#include "Collector_if.h"  
#include "Statistics_if.h"
```

Include dependency graph for StatisticsDataFile\_if.h:



This graph shows which files directly or indirectly include this file:

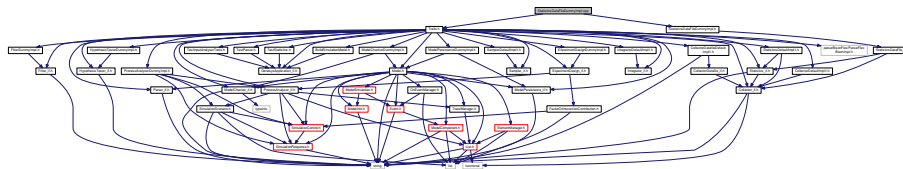


## Classes

- class [StatisticsDatafile\\_if](#)

## 6.118 StatisticsDataFileDummyImpl.cpp File Reference

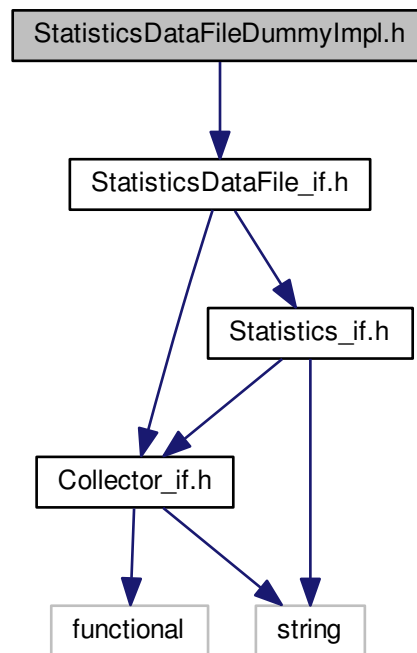
```
#include "StatisticsDataFileDummyImpl.h"
#include "Traits.h"
Include dependency graph for StatisticsDataFileDummyImpl.cpp:
```



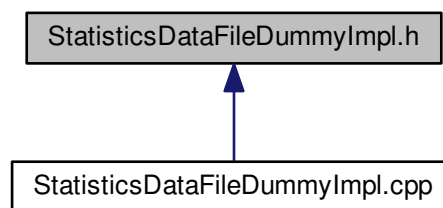
## 6.119 StatisticsDataFileDummyImpl.h File Reference

```
#include "StatisticsDataFile_if.h"
```

Include dependency graph for StatisticsDataFileDummyImpl.h:



This graph shows which files directly or indirectly include this file:



## Classes

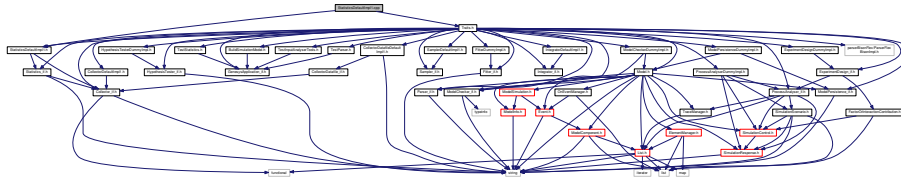
- class [StatisticsDataFileDummyImpl](#)

## 6.120 StatisticsDefaultImpl1.cpp File Reference

```
#include "StatisticsDefaultImpl1.h"
```

```
#include "Traits.h"
```

Include dependency graph for StatisticsDefaultImpl1.cpp:

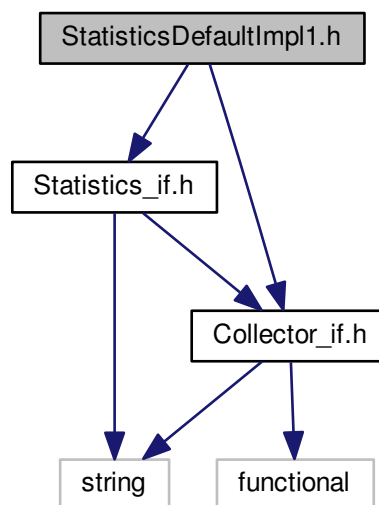


## 6.121 StatisticsDefaultImpl1.h File Reference

```
#include "Statistics_if.h"
```

```
#include "Collector_if.h"
```

Include dependency graph for StatisticsDefaultImpl1.h:



This graph shows which files directly or indirectly include this file:



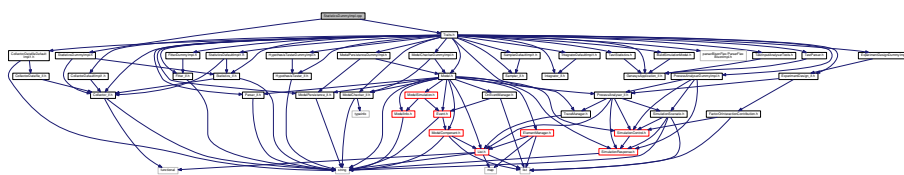


## Classes

- class [StatisticsDefaultImpl1](#)

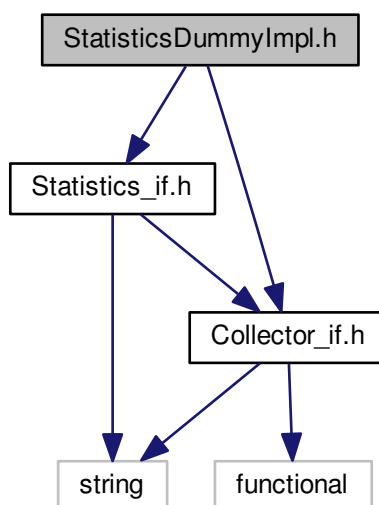
## 6.122 StatisticsDummyImpl.cpp File Reference

```
#include "StatisticsDummyImpl.h"
#include "Traits.h"
Include dependency graph for StatisticsDummyImpl.cpp:
```

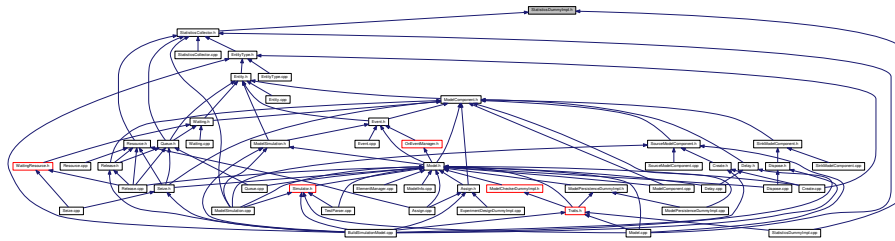


## 6.123 StatisticsDummyImpl.h File Reference

```
#include "Statistics_if.h"
#include "Collector_if.h"
Include dependency graph for StatisticsDummyImpl.h:
```



This graph shows which files directly or indirectly include this file:



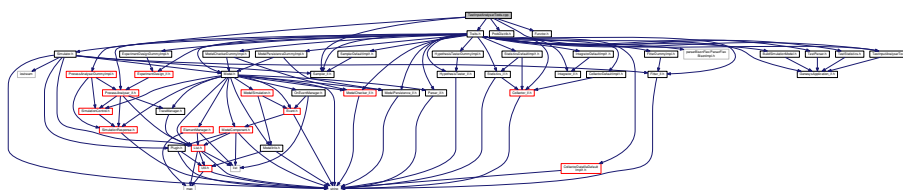
## Classes

- class [StatisticsDummyImpl](#)

## 6.124 TestInputAnalyserTools.cpp File Reference

```
#include "TestInputAnalyserTools.h"
#include "Simulator.h"
#include "Sampler_if.h"
#include "ProbDistrib.h"
#include "Traits.h"
#include "Functor.h"
```

Include dependency graph for TestInputAnalyserTools.cpp:



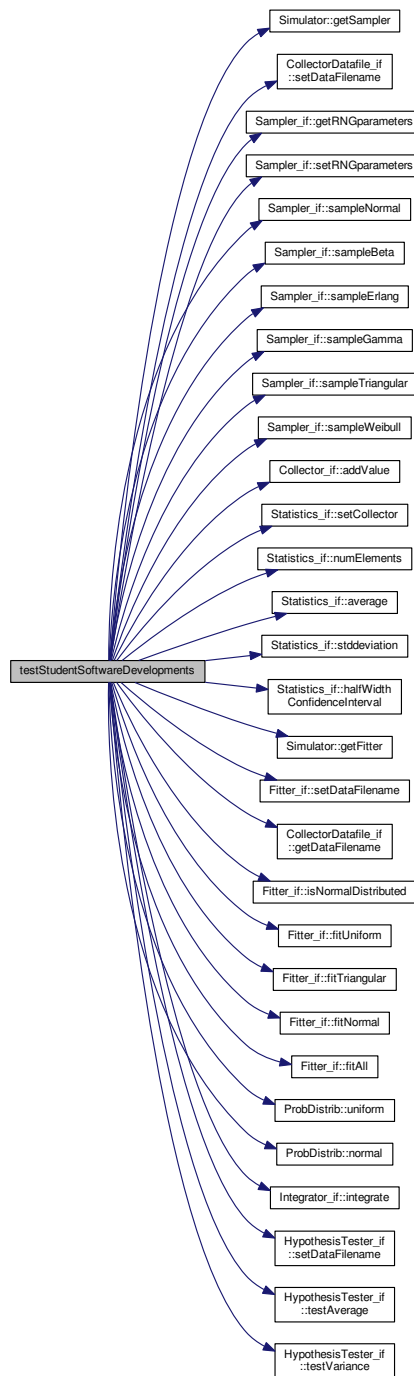
## Functions

- void [testStudentSoftwareDevelopments](#) ()

### 6.124.1 Function Documentation

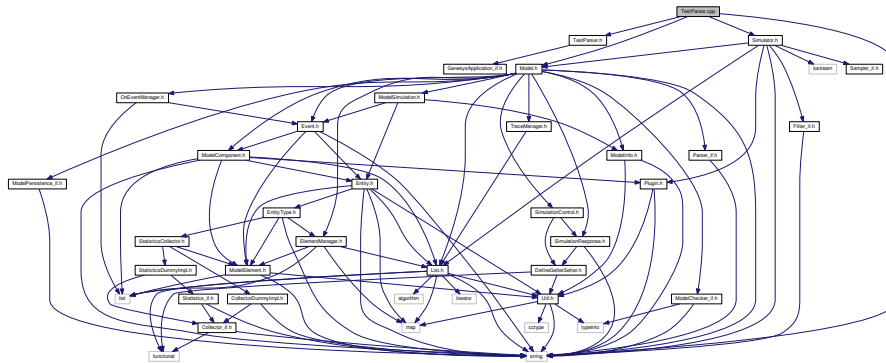
## 6.124.1.1 void testStudentSoftwareDevelopments ( )

Here is the call graph for this function:





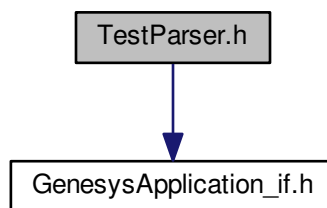
Include dependency graph for TestParser.cpp:



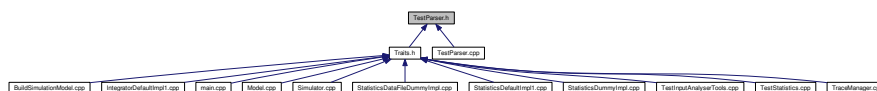
## 6.127 TestParser.h File Reference

```
#include "GenesysApplication_if.h"
```

Include dependency graph for TestParser.h:



This graph shows which files directly or indirectly include this file:

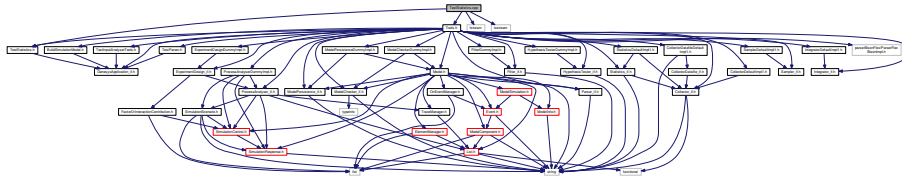


## Classes

- class **TestParser**

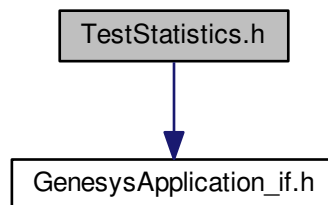
## 6.128 TestStatistics.cpp File Reference

```
#include "TestStatistics.h"
#include <fstream>
#include <iostream>
#include "Traits.h"
Include dependency graph for TestStatistics.cpp:
```

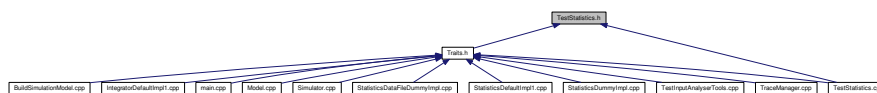


## 6.129 TestStatistics.h File Reference

```
#include "GenesysApplication_if.h"
Include dependency graph for TestStatistics.h:
```



This graph shows which files directly or indirectly include this file:



## Classes

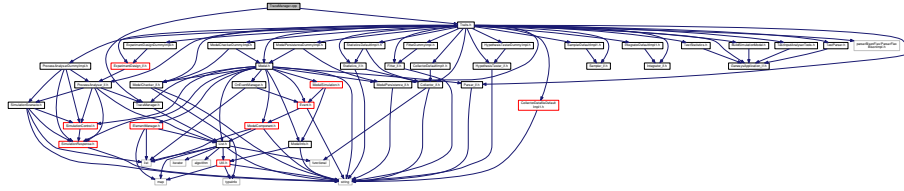
- class [TestStatistics](#)

## 6.130 TraceManager.cpp File Reference

```
#include "TraceManager.h"
```

```
#include "Traits.h"
```

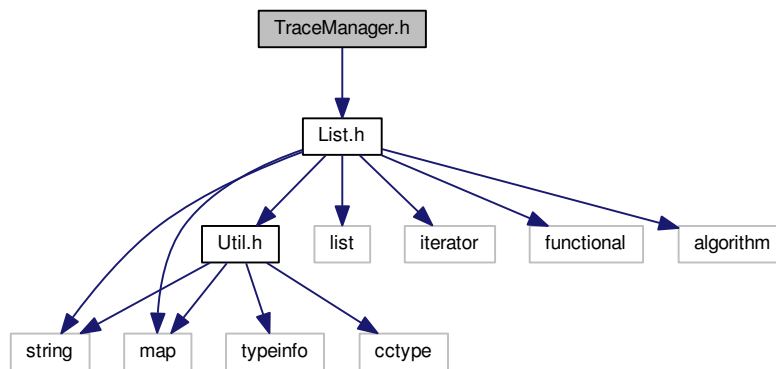
Include dependency graph for TraceManager.cpp:



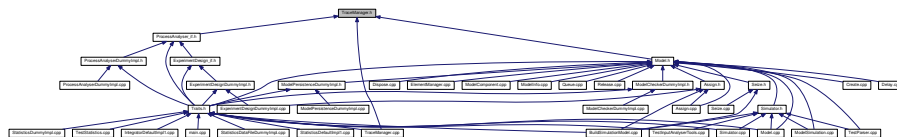
## 6.131 TraceManager.h File Reference

```
#include "List.h"
```

Include dependency graph for TraceManager.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [TraceEvent](#)
- class [TraceErrorEvent](#)
- class [TraceSimulationEvent](#)
- class [TraceSimulationProcess](#)
- class [TraceManager](#)

## Typedefs

- typedef void(\* [traceListener](#)) ([TraceEvent](#))
- typedef void(\* [traceErrorListener](#)) ([TraceErrorEvent](#))
- typedef void(\* [traceSimulationListener](#)) ([TraceSimulationEvent](#))
- typedef void(\* [traceSimulationProcessListener](#)) ([TraceSimulationProcess](#))

### 6.131.1 Typedef Documentation

6.131.1.1 typedef void(\* [traceErrorListener](#)) ([TraceErrorEvent](#))

6.131.1.2 typedef void(\* [traceListener](#)) ([TraceEvent](#))

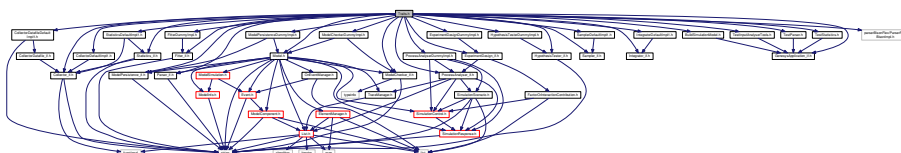
6.131.1.3 typedef void(\* [traceSimulationListener](#)) ([TraceSimulationEvent](#))

6.131.1.4 typedef void(\* [traceSimulationProcessListener](#)) ([TraceSimulationProcess](#))

## 6.132 Traits.h File Reference

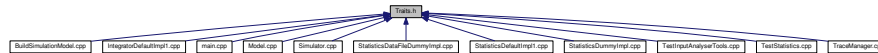
```
#include "Model.h"
#include "Collector_if.h"
#include "Sampler_if.h"
#include "Fitter_if.h"
#include "ModelChecker_if.h"
#include "Parser_if.h"
#include "Statistics_if.h"
#include "Integrator_if.h"
#include "HypothesisTester_if.h"
#include "ModelPersistence_if.h"
#include "GenesysApplication_if.h"
#include "ProcessAnalyser_if.h"
#include "ExperimentDesign_if.h"
#include "BuildSimulationModel.h"
#include "TestInputAnalyserTools.h"
#include "TestParser.h"
#include "TestStatistics.h"
#include "FitterDummyImpl.h"
#include "ModelCheckerDummyImpl.h"
#include "ExperimentDesignDummyImpl.h"
#include "ProcessAnalyserDummyImpl.h"
#include "HypothesisTesterDummyImpl.h"
#include "ModelPersistenceDummyImpl.h"
#include "CollectorDefaultImpl1.h"
#include "CollectorDatafileDefaultImpl1.h"
#include "StatisticsDefaultImpl1.h"
#include "IntegratorDefaultImpl1.h"
#include "SamplerDefaultImpl1.h"
#include "parserBisonFlex/ParserFlexBisonImpl.h"
```

Include dependency graph for Traits.h:





This graph shows which files directly or indirectly include this file:

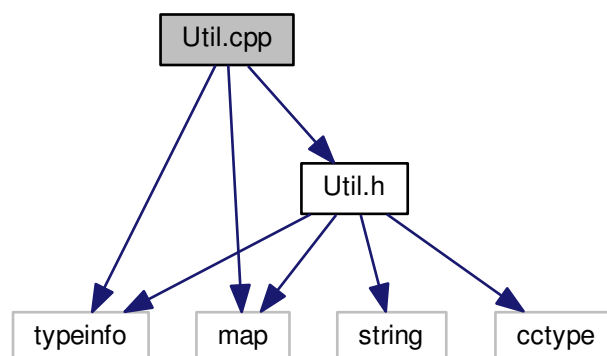


## Classes

- struct [Traits< T >](#)
- struct [Traits< GenesysApplication\\_if >](#)
- struct [Traits< Model >](#)
- struct [Traits< ModelPersistence\\_if >](#)
- struct [Traits< ModelComponent >](#)
- struct [Traits< ModelChecker\\_if >](#)
- struct [Traits< Parser\\_if >](#)
- struct [Traits< Collector\\_if >](#)
- struct [Traits< Statistics\\_if >](#)
- struct [Traits< Integrator\\_if >](#)
- struct [Traits< Sampler\\_if >](#)
- struct [Traits< Fitter\\_if >](#)
- struct [Traits< HypothesisTester\\_if >](#)
- struct [Traits< ExperimentDesign\\_if >](#)
- struct [Traits< ProcessAnalyser\\_if >](#)

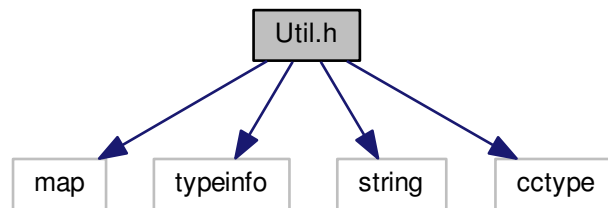
## 6.133 Util.cpp File Reference

```
#include <typeinfo>
#include <map>
#include "Util.h"
Include dependency graph for Util.cpp:
```

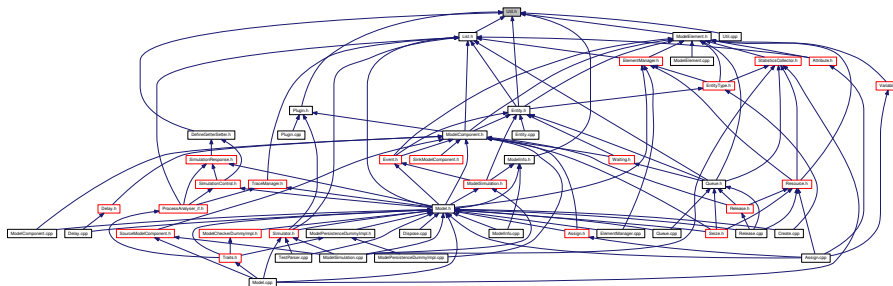


## 6.134 Util.h File Reference

```
#include <map>
#include <typeinfo>
#include <string>
#include <cctype>
Include dependency graph for Util.h:
```



This graph shows which files directly or indirectly include this file:



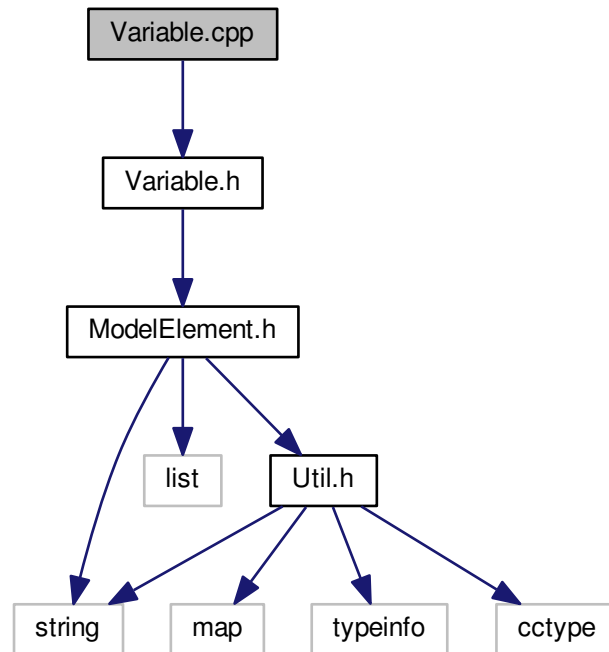
### Classes

- class [Util](#)

## 6.135 Variable.cpp File Reference

```
#include "Variable.h"
```

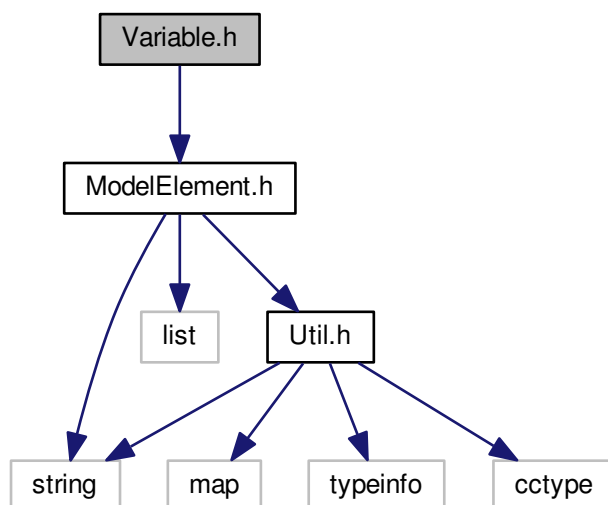
Include dependency graph for Variable.cpp:



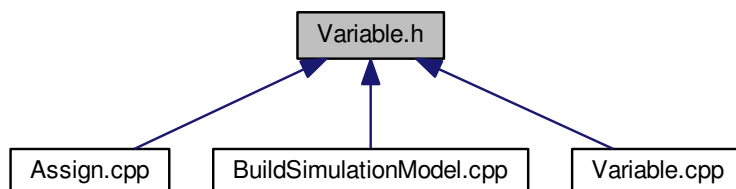
## 6.136 Variable.h File Reference

```
#include "ModelElement.h"
```

Include dependency graph for Variable.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Variable](#)

## 6.137 Waiting.cpp File Reference

```
#include "Waiting.h"
```

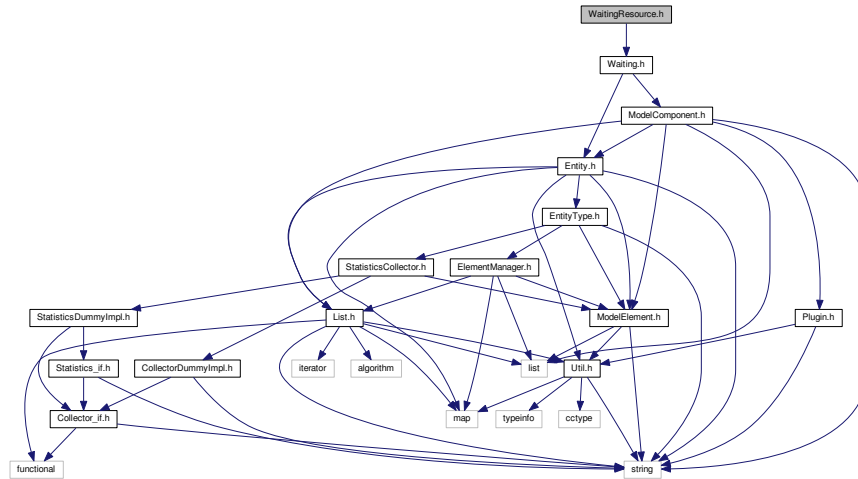




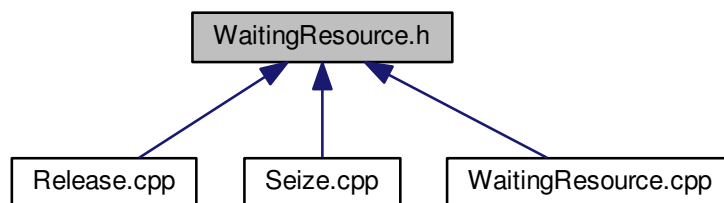
## 6.140 WaitingResource.h File Reference

```
#include "Waiting.h"
```

Include dependency graph for WaitingResource.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class [WaitingResource](#)

